



Product Overview



Product Overview

Note

Before using this information, be sure to read the general information in the Notices section at the end of this document.

24 April 2009

This edition applies to version 6, release 2, modification 0 of WebSphere Process Server for Multiplatforms (product number 5724-L01) and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, send an e-mail message to 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.

© **Copyright International Business Machines Corporation 2005, 2009.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

PDF books and the information center

PDF books are provided as a convenience for printing and offline reading. For the latest information, see the online information center.



As a set, the PDF books contain the same content as the information center.

The PDF documentation is available within a quarter after a major release of the information center, such as Version 6.0 or Version 6.1.

The PDF documentation is updated less frequently than the information center, but more frequently than the Redbooks®. In general, PDF books are updated when enough changes are accumulated for the book.

Links to topics outside a PDF book go to the information center on the Web. Links to targets outside a PDF book are marked by icons that indicate whether the target is a PDF book or a Web page.

Table 1. Icons that prefix links to topics outside this book

Icon	Description
	<p>A link to a Web page, including a page in the information center.</p> <p>Links to the information center go through an indirection routing service, so that they continue to work even if target topic is moved to a new location.</p> <p>If you want to find a linked page in a local information center, you can search for the link title. Alternatively, you can search for the topic id. If the search results in several topics for different product variants, you can use the search result Group by controls to identify the topic instance that you want to view. For example:</p> <ol style="list-style-type: none">1. Copy the link URL; for example, right-click the link then select Copy link location. For example: <code>http://www14.software.ibm.com/webapp/wsbroker/redirect?version=wbpm620&product=wesb-dist&topic=tins_apply_service</code>2. Copy the topic id after <code>&topic=</code>. For example: <code>tins_apply_service</code>3. In the search field of your local information center, paste the topic id. If you have the documentation feature installed locally, the search result will list the topic. For example: <div data-bbox="617 1396 1458 1570" style="border: 1px solid black; border-radius: 10px; padding: 10px;"><p>1 result(s) found for</p><p>Group by: None Platform Version Product</p><p>Show Summary</p><p>Installing fix packs and refresh packs with the Update Installer</p></div> <ol style="list-style-type: none">4. Click the link in the search result to display the topic.
	<p>A link to a PDF book.</p>

Contents

PDF books and the information center **iii**

Figures **vii**

Product overview. **1**

Introduction to WebSphere Process Server 1

What is new in this release 2

Product family overview 5

Architectural overview of WebSphere Process Server 9

 Service-oriented architecture core 10

 Supporting services. 17

 Service components 20

Deployment environments in WebSphere Process

Server 23

Business Space powered by WebSphere 24

The enterprise service bus in WebSphere Process

Server 26

 Connecting services through an enterprise service

 bus 26

 Enterprise service bus messaging infrastructure 28

 Service applications and service modules . . . 33

 Message Service clients 45

WebSphere Adapters 45

Development and deployment of applications on

WebSphere Process Server 46

Migration to WebSphere Process Server 48

Administration of applications on WebSphere

Process Server 48

 Administrative control of mediation processing 50

Security on WebSphere Process Server 51

Monitoring on WebSphere Process Server 51

Samples 52

 Installing and accessing the Samples Gallery . . 52

 Business Process Management samples 54

Standards compliance 54

 Accessibility 54

 Federal Information Processing Standards . . . 56

 Internet Protocol Version 6 56

Globalization 56

Notices **63**

Figures

1. WebSphere Process Server component-based framework	10	6. Business Space and Business Process Management products	25
2. WebSphere Process Server component-based framework	11	7. An enterprise service bus	28
3. Overview of SMO structure	15	8. Simplified example of a mediation module	37
4. WebSphere Process Server component-based framework	17	9. Simplified example of an EAR file containing a mediation module	38
5. WebSphere Process Server component-based framework	21	10. Mediation module containing three mediation primitives	39
		11. Detailed schematic of a WebSphere Adapter.	46

Product overview

WebSphere® Process Server is a high-performance business engine to help form processes to meet business goals. It allows the deployment of standards-based business integration applications in a *service-oriented architecture* (SOA), which takes everyday business applications and breaks them down into individual business functions and processes, rendering them as services.

Related information

 PDF documentation

WebSphere Process Server documentation (in PDF format)

 Information roadmaps

Business Process Management information roadmaps on IBM developerWorks organize information about WebSphere Process Server, WebSphere ESB, and the other products in the portfolio.

 IBM Education Assistant

Multimedia educational modules about WebSphere Process Server, provided by IBM Education Assistant.

 Overview

Overview tab, on product library Web page. Use this page to access announcements, data sheets, and other general library documents related to WebSphere ESB.

Introduction to WebSphere Process Server

IBM® WebSphere Process Server is a business process integration server that has evolved from proven business integration concepts, application server technologies, and the latest open standards. WebSphere Process Server is a high-performance business engine to help form processes to meet business goals.

WebSphere Process Server allows the deployment of standards-based business integration applications in a service-oriented architecture (SOA), which takes everyday business applications and breaks them down into individual business functions and processes, rendering them as services. Based on the robust J2EE 1.4 infrastructure and associated platform services provided by WebSphere Application Server, WebSphere Process Server can help you meet current business integration challenges. This includes, but is not limited to, business process automation.

WebSphere Process Server enables the deployment of processes that span people, systems, applications, tasks, rules, and the interactions among them. It supports both long-running and short-running business processes, providing transaction rollback-like functionality for loosely coupled business processes.

Hardware and software requirements

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

Information roadmaps

To help you to navigate through the available information sources, both within and beyond the product information centers, business process management information roadmaps are available online on IBM developerWorks® at www.ibm.com/developerworks/websphere/zones/bpm/roadmaps/bpm_info_resources.html.

What is new in this release

This version includes several new features to enhance business flexibility and enable faster and more effective deployment of applications.

Note: The information center has been updated for IBM WebSphere Process Server, version 6.2.0.1. For more information about installing fix packs, see Getting fixes.

Welcome to WebSphere Process Server, version 6.2.0.1, which includes the following new features:

- Extended capabilities of the Business Space widgets shipped with WebSphere Process Server that support scenarios where they are integrated into a WebSphere Portal environment.
- Usability enhancements to the Managing Tasks and Workflows widgets in Business Space that provide the ability to filter and sort tasks based on business data.
- Support for Microsoft® SQL Server JDBC Driver.
- New application programming interfaces (APIs) that enable manipulation of an individual process instance from the beginning of its lifecycle. The new APIs allow scenarios such as creating a process instance and starting it from a specific activity in the process.
- Support for unreferenced SOAP attachments.

WebSphere Process Server, version 6.2.0 includes the following new features:

- Real-time access to critical process information and the ability to interact with processes to influence runtime process execution in response to changing business conditions:
 - Extends the capabilities for the business user introduced in WebSphere Process Server Version 6.1.2 -- using the common Business Space powered by WebSphere user interface -- for worklist and task management with new features, such as the ability to view the related business process or task history information from the human workflow diagram, and the ability for business users to create, view, modify, verify status and cancel subtasks from within their business space.
 - Enables business users with enhanced flexibility and control over runtime processes through the ability to characterize a collection of process activities and their associated data as a defined unit, enabling them to modify the flow of steps within those inflight process instances by skipping activities, jumping forward and backward between activities, and adding additional activities from within their Business Space.
 - Delivers new Business Calendar Manager widgets in Business Space that enable business users to add, update, and delete timetable and time interval information to reflect available time changes based on ongoing business operations.

- Enables directed deployment from WebSphere Business Modeler in to WebSphere Process Server, so that you can directly deploy models to the runtime environment.
- Simplifies the identification of failed flows by using a graphical tree view that correlates log statements and errors that appear in the system out log, enabling faster problem determination.
- Powerful enhancements that simplify the process of application deployment, grant additional control over the artifacts in the deployment environment, and improve user productivity for ongoing application operation and administration:
 - Supports direct deployment of executable process models from WebSphere Business Modeler.
 - Introduces enhancements that enable module versioning and module-aware service versioning (SCA bindings only).
 - Enables configuration for role-based access to timetable information in the runtime environment with Business Space widgets.
 - Supports the population of relationship tables with instance data through a SQL script or Java™ program, eliminating the need for manual data entry.
 - Enables easier installation and configuration of WebSphere Process Server, which is key to a successful deployment of your SOA infrastructure. New installation and configuration enhancements that increase usability and speed time-to-value include a full installation of WebSphere Process Server Version 6.2, including WebSphere Application Server Network Deployment and the Web Services Feature Pack for WebSphere Application Server, with the creation of a profile that includes the Web Services Feature Pack augmentation; improved installation error determination after a installation failure or partial success; an installation verification tool that validates that the installation produced a successful server configuration; Installation Factory Integrated Install Package (IIP) support for creating custom installation packages; and scripting capabilities for configuration of production environments.
 - Improves installation experience on z/OS® with enhancements that include a reduction in the number of authentication aliases generated for resources within WebSphere Process Server for z/OS, the ability for customers to use the zPMT configuration tool to create augment response files for use in augmenting their WebSphere Process Server for z/OS installation, and the generation of a more consumable Data Definition Language (DDL).
 - Simplifies problem determination with consistent fault support for bindings (including tools in WebSphere Integration Developer) and unified failed event management for all components (including mediations).
 - Empowers the administrator with widgets in Business Space for monitoring the health of the system. They also provide additional information from the administrative console including SCA module details, Enterprise Java Beans (EJB) import binding information, and contextual links throughout the panels, and they provide more control when installing an SCA module through the administrative console or equivalent scripting, and optionally deploying it to a target server or cluster.
 - Simplifies the management of the Service Integration Bus with a new browser view that is tailored to the user who explores existing buses in support of ongoing operations.
 - Delivers simplified user experience with the reporting capabilities of Business Process Choreographer Observer now moved to Business Process Choreographer Explorer to use these reporting capabilities when administrating business processes and human tasks. In addition, includes new

capabilities around custom view definition with time constraints relative to when the view is used and where the selected actions to be displayed match the anticipated scenarios.

- Enhancements that facilitate faster and more effective development, testing, deployment, and execution of business process solutions:
 - Delivers enhancements to more easily support additional use cases, including true support for arrays and discovery enablement for JAX-WS and JAXB2-based Java services.
 - Supports data handlers that are ready to use as-is, for all bindings, as configured in WebSphere Integration Developer.
 - Delivers capabilities to build more flexible and intelligent process solutions through the ability to access business context information and programmatically propagate that context for all binding types. A new context service is responsible for propagating the context (including the protocol headers such as the JMS header and the user context such as account ID) along a Service Component Architecture (SCA) invocation path. The context service offers a set of APIs and configurable settings.
 - Delivers process model extensions for Business Process Modeling Notation support, including processes defined in WebSphere Business Modeler and WebSphere Integration Developer that include generalized flows (previously called "Cyclic Flows") in which the navigation logic has been set using the visual tools and the behavior specified for incoming and outgoing links.
 - Enables the definition of an exit condition on each process step, which specifies the criteria that must be met for an activity to be automatically skipped when reached by navigation, and can be checked on entry, on exit, or on both entry and exit of process step execution.
 - Improves publish response time and messaging engine start-up time.
- Extended reach of process solutions:
 - Delivers new policy-driven connectivity for administrators to configure service mediations through policies.
 - Enhances Web Services standards support.
 - Enhances service mediation capabilities.
- Enhancements designed to ease the effort of migrating from WebSphere Business Integration heritage server solutions:
 - WebSphere Business Integration Server Foundation solutions: Improves integration developer productivity by enabling the import of entire WebSphere Studio Application Developer Integration Edition workspaces into WebSphere Integration Developer, and by aiding in the migration of service interfaces to custom Java code.
 - WebSphere InterChange Server solutions: Improves time to value when migrating solutions by enabling the use of migrated maps with WebSphere Version 6 Adapters on WebSphere Process Server, generating native SCA bindings (MQ, JMS, HTTP, and EJB) for use with migrated maps, supporting running text-based heritage data handlers on WebSphere Process Server, and improving the runtime performance of migrated content.
 - WebSphere Business Integration Server Express™ and WebSphere Business Integration Server Express Plus solutions: Supports the migration of WBI-SX artifacts to new value-add solutions assembled with WebSphere Integration Developer for deployment on WebSphere Process Server. This provides a growth path to an enterprise Business Process Management solution that enables significant value-add capabilities and platform support, and increased workloads.

- WebSphere MQ Workflow solutions: Enhances qualities of service with improved human task performance for WebSphere Process Server workflow solutions and new migration utility options for generating process models optimized for visual recognition and runtime performance.
- Platform alignment and currency:
 - Supports WebSphere Application Server Version 6.1.
 - Supports IBM z/OS and z/OS.e 1.7, or later, including zFS, enabling WebSphere Process Server to run on the latest release of z/OS to use native z/OS facilities.
 - Supports IBM IMS™ Version 10, enabling WebSphere Process Server solutions to use the enhanced database and transaction processing capabilities of the latest IMS release.
 - Supports Microsoft Windows® Vista as a runtime platform in nonproduction environments, enabling WebSphere Process Server to run on the latest version of the Microsoft server operating system in development and test scenarios.
 - Complies with the security settings as defined by the Federal Desktop Core Configuration (FDCC) for the U.S. federal government.

Product family overview

WebSphere Process Server is part of the IBM WebSphere Business Process Management platform and works with many other IBM products.

IBM WebSphere Application Server Network Deployment

WebSphere Process Server is based on the robust J2EE infrastructure and associated platform services provided by WebSphere Application Server. WebSphere Application Server includes a built-in JMS engine, for messaging between J2EE applications, and connectivity for messaging with WebSphere MQ. For more information about WebSphere Application Server Network Deployment offerings, see the WebSphere Application Server documentation.

WebSphere Process Server also works with infrastructure and platform services from WebSphere Application Server. For more information about WebSphere Application Server, see the WebSphere Application Server Information Center.

IBM WebSphere Enterprise Service Bus

WebSphere Process Server provides a fully-converged, standards-based business process engine, using the full power of WebSphere Application Server. It also includes the same technology as WebSphere Enterprise Service Bus, providing the same enterprise service bus capabilities.

No additional license for WebSphere Enterprise Service Bus is required to take advantage of the enterprise service bus capabilities. However, you can deploy additional, purchased 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 Integration Developer

WebSphere Integration Developer is the development environment for WebSphere Process Server. It is a common tool for building service-oriented architecture (SOA)-based integration solutions across WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Adapters. For more information about WebSphere Integration Developer, see the WebSphere Integration Developer documentation.

IBM WebSphere Dynamic Process Edition

WebSphere Dynamic Process Edition is a comprehensive set of software offerings designed for enterprise-wide integration capabilities and service-oriented architecture (SOA). You can optimize business processes with dynamic capabilities for rapid change and customization. WebSphere Dynamic Process Edition is built on SOA-enabled products and capabilities that provide the foundation for integrating end-to-end business processes across an enterprise. It includes three products: WebSphere Business Modeler, WebSphere Business Services Fabric, and WebSphere Business Monitor. For more information about WebSphere Dynamic Process Edition, see the WebSphere Dynamic Process Edition product documentation library.

IBM WebSphere Business Services Fabric

WebSphere Business Services Fabric provides an end-to-end platform for the rapid assembly, delivery, and governance of industry-focused composite business services in an SOA. It adds an industry-specific layer to the IBM SOA Foundation by enabling dynamic business service personalization and delivery based on business context. WebSphere Service Registry and Repository provides the technical service metadata underpinning, serving as a prerequisite for the WebSphere Business Services Fabric solution. For more information about WebSphere Business Services Fabric, see the WebSphere Business Services Fabric product documentation library.

IBM WebSphere Business Modeler and IBM WebSphere Business Monitor

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 and IBM WebSphere Business Monitor.

For more information about these products, see the following documentation:

- [WebSphere Business Modeler documentation](#)
- [WebSphere Business Monitor documentation](#)

IBM WebSphere Service Registry and Repository

WebSphere Service Registry and Repository (WSRR) is a system for storing, accessing and managing information, commonly referred as service metadata, used in the selection, invocation, management, governance and reuse of services in a successful service-oriented architecture (SOA). In other words, it is where you store information about services in your systems, or in other organizations' systems, that you already use, plan to use, or want to be aware of. For example, an application can check the WSSR just before invoking a service to locate the service instance

best satisfying its functionality and performance needs. WSRR also plays a role in other stages of the SOA life cycle. For more information about WebSphere Service Registry and Repository, see the WebSphere Service Registry and Repository documentation

IBM WebSphere MQ

WebSphere MQ, available on more than 80 platform configurations, offers application integration connectivity and integrates many differing platforms, systems, and applications. It delivers heterogeneous messaging, extending your enterprise service bus with reliable message delivery. For more information about WebSphere MQ, see the WebSphere MQ product documentation library.

IBM WebSphere Message Broker

WebSphere Message Broker makes full use of the transport and connectivity options provided by WebSphere MQ and WebSphere Application Server, and allows additional routing and transformation capabilities to implement an integration-based enterprise service bus. For more information about WebSphere Message Broker, see the WebSphere Message Broker product documentation library.

IBM WebSphere DataPower® SOA Appliances

WebSphere DataPower SOA Appliances are easy-to-deploy network devices that simplify, help secure, and accelerate XML and Web services deployments. They extend core SOA infrastructure components such as WebSphere ESB, WebSphere MQ, WebSphere Message Broker, and WebSphere Process Server. For more information about WebSphere DataPower SOA Appliances, see the WebSphere DataPower SOA Appliances product documentation library.

IBM WebSphere Portal

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 product documentation library.

IBM WebSphere Partner Gateway

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 product documentation library.

IBM WebSphere Adapters

WebSphere Adapters allow for integration of existing Enterprise Information System infrastructure and applications that are deployed on WebSphere Process Server. WebSphere Adapters enable you to quickly and easily create integrated processes that exchange information between enterprise resource planning, human resource, customer relationship management, and supply chain systems.

Application adapters extract data and transaction information from cross-industry and industry-specific packaged applications and connect them to a central hub. Technology adapters provide connectivity to access data, technologies and protocols that enhance integration infrastructure. You can use the Adapter Development Toolkit to create custom adapters.

Some WebSphere Adapters are included components with WebSphere Integration Developer.

For more information about WebSphere Adapters, see the WebSphere Integration Developer documentation.

IBM Rational® Application Developer and IBM Rational Software Architect

WebSphere Integration Developer can be used in conjunction with Rational Application Developer, or Rational Software Architect, to create a unique, integrated, and powerful integration development platform.

For more information about these products, see the Rational Application Developer Information Center, and the Rational Software Architect Information Center.

IBM CICS® Transaction Gateway and IBM WebSphere Host Access Transformation Services

An IBM enterprise modernization portfolio that includes CICS Transaction Gateway and WebSphere Host Access Transformation Services allows you to extend existing applications for reuse in enterprise processes.

For more information about these products, see the CICS Transaction Gateway Library and the WebSphere Host Access Transformation Services (HATS) Information Center.

IBM WebSphere Application Toolkit

The WebSphere Application Server Toolkit is a set of tools that help you to assemble, test, and deploy Web services for use in WebSphere Process Server.

For more information, see the WebSphere Application Server Toolkit documentation on the WebSphere Application Server Information Center.

IBM WebSphere Extended Deployment

WebSphere Extended Deployment provides a WebSphere Process Server network deployment environment with the ability to adjust the resources between clusters in the environment to meet processing objectives that you define as policies. Because of the ebb and flow of application volumes, there can be insufficient processing power available to satisfy requests during peak periods, and it can be difficult to optimize resources so that critical applications get needed processing time.

Dynamic reapportioning of processing power at these times can help you meet business needs. WebSphere Extended Deployment dynamically removes resources from clusters with low application volumes and adds them to clusters that are servicing the applications that require the additional resources. The processing priorities are specified in WebSphere Extended Deployment as policies.

For more information about WebSphere Extended Deployment, see the WebSphere Extended Deployment Information Center.

IBM WebSphere Transformation Extender

WebSphere Transformation Extender is a powerful, transaction-oriented, data integration solution that automates the transformation of high-volume, complex transactions without the need for hand-coding. It performs transformation and routing of data from source systems to target systems in batch and real-time environments. The sources can include files, relational databases, message-oriented middleware (MOMs), packaged applications, or other external sources. After retrieving the data from its sources, the WebSphere Transformation Extender product transforms it and routes it to any number of targets where it is needed, providing the appropriate content and format for each target system. For more information about WebSphere Transformation Extender, see WebSphere Transformation Extender product library.

Architectural overview of WebSphere Process Server

WebSphere Process Server is a service-oriented architecture (SOA) integration platform built on a uniform invocation programming model and a uniform data representation model. It provides a fully-converged, standards-based business process engine, using the full power of WebSphere Application Server.

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 the 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.

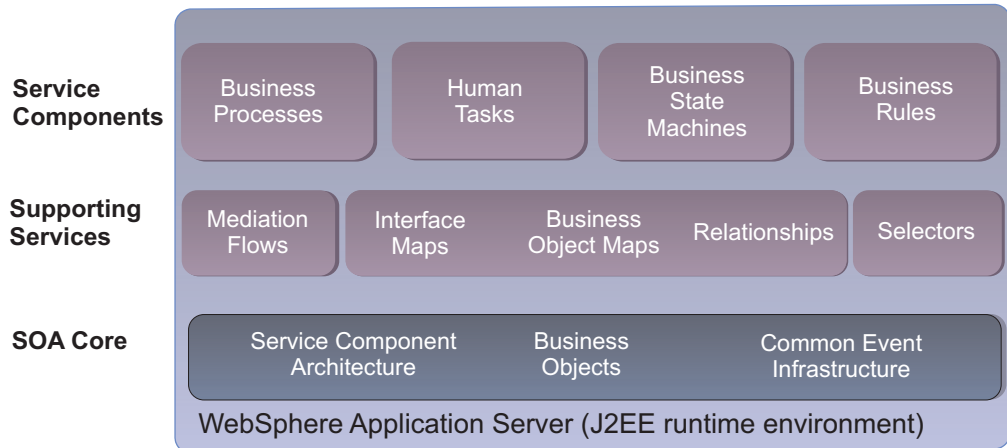


Figure 1. WebSphere Process Server component-based framework

Service-oriented architecture core

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

Service-oriented architecture (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. WebSphere Process Server enables deployment of standards-based process integration solutions in an SOA. 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.

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 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.

The following diagram shows the WebSphere Process Server component-based framework.

One component-based framework addresses all styles of integration.

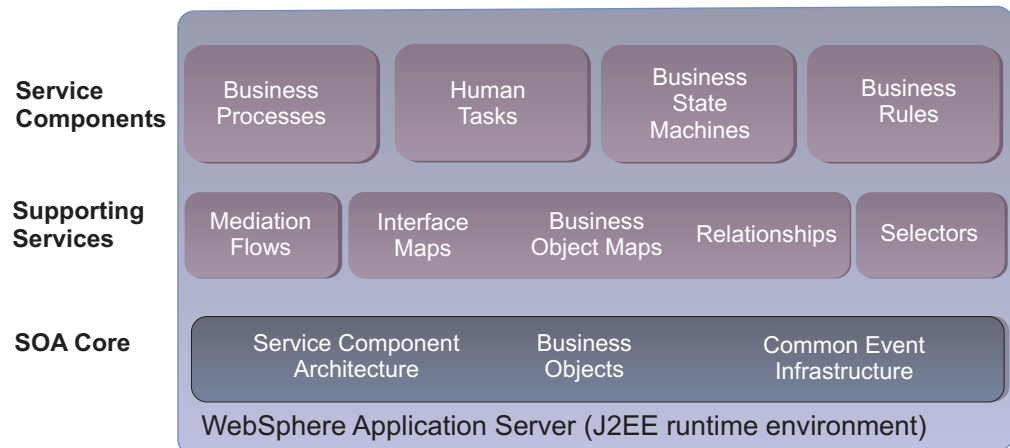


Figure 2. WebSphere Process Server component-based framework

Service Component Architecture

Service Component Architecture presents all elements of business transactions in a service-oriented way in the WebSphere Process Server runtime environment.

Service Component Architecture (SCA) is an architecture in which all elements of a business transaction, such as access to Web services, Enterprise Information System (EIS) service assets, business rules, workflows, databases and so on, are represented in a service-oriented way.

SCA 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 SCA 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. You can specify service qualifiers 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, additional required processing can take place at run time.

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 the same application, or to other applications on enterprise information systems (EIS). This allows working with IBM WebSphere Adapters. For more information about imports and exports, see Service applications.

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.

Part of the IBM WebSphere Application Server capabilities that are built into WebSphere Process Server, SDOs provide a framework for data application development that simplifies the J2EE data programming model.

WebSphere Process Server includes business objects, which are enhanced SDOs, based on the data-access technology. SDOs provide a universal means of describing disparate data (for example, JDBC ResultSet and XML Schema described data). 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, the business object framework is made up of the following 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 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 Network Deployment documentation.

Related concepts

Business object maps

Business object maps are a way of relating business objects.

Service message objects

Service message objects (SMOs) provide an abstraction layer for processing and manipulating messages exchanged between services.

SMO model

Mediation primitives process messages as SMOs. SMOs are enhanced Service Data Objects (SDOs), and the SMO model is a pattern for using SDO DataObjects to represent messages. The SMO contains a representation of the following groups of data:

- 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.
- The body of the message: the message payload. The message payload is the application data exchanged between service endpoints.
- Message attachments.
- 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 WebSphere Integration Developer.

SMO content

All SMOs have the same basic structure. The structure consists of a root data object called a ServiceMessageObject, which contains other data objects representing the header, body, attachments, and context data. The precise structure of the headers, body, and context depends on how you define the mediation flow at integration development. The mediation flow is used at runtime to mediate between services.

The SMO headers contain information that originates from a specific export or import binding (a binding specifies the message format and protocol details). 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 are:

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

Typically, the structure of the SMO body, which holds the application data, is determined by the Web Services Description Language (WSDL) message that you specify when you configure a mediation flow.

If a SOAP message has unreferenced attachments, they are stored in SMO attachments elements. Unreferenced attachments are MIME parts included in a SOAP/HTTP message, the attachments are not defined in the WSDL portType.

Note: You can only send or receive SOAP attachments if the module binding is one of the following types: Web service binding **SOAP 1.1/HTTP** using JAX-WS, Web service binding **SOAP 1.2/HTTP** using JAX-WS, or SCA binding.

SMO context objects are either user-defined or system-defined. You can use user-defined context objects to store a property that mediation primitives can use later in the flow. You define the structure of a user-defined context object in a business object, and use the business object in the input node of the request flow. The correlation context, transient context and shared context are user-defined context objects.

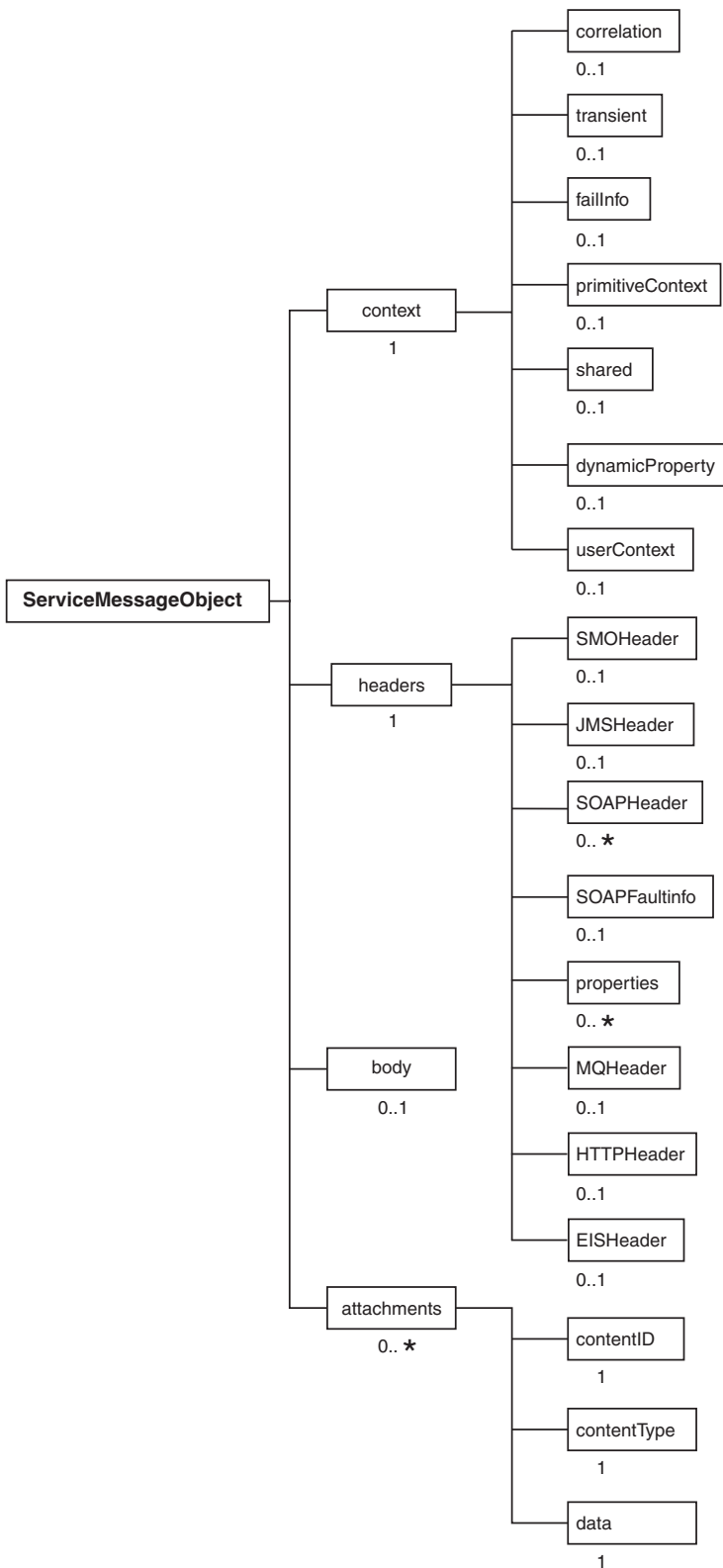


Figure 3. Overview of SMO structure. The context, headers, body and attachments of a ServiceMessageObject

The SMO provides an interface to access and modify message headers, message payloads, message attachments, and message context.

How the runtime uses the SMO

The runtime operates on messages that are in flight between interaction endpoints. The runtime creates SMO objects, which a mediation flow uses to process a message.

When you create 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. The runtime uses this information to convert messages into SMO objects of the specified type.

To provide dynamic routing, the interaction endpoints can be looked up using WebSphere Service Registry and Repository (WSRR), or a database. The result of the WSRR query, or database lookup, can be stored at a particular location in the SMO, from where the runtime will take the dynamic endpoint.

Common Event Infrastructure in WebSphere Process Server

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

The infrastructure portion of the Common Event Infrastructure is included as part of the underlying IBM WebSphere Application Server capabilities in WebSphere Process Server. The event emitting capabilities are additional functions of WebSphere Process Server.

The *Common Event Infrastructure (CEI)* is the implementation of a set of APIs and infrastructure for the creation, transmission, persistence, and distribution of business, system, and network Common Base Events. A *Common Base Event* is a specification based on XML that defines a mechanism for managing events – such as logging, tracing, management, and business events – in business enterprise applications.

CEI provides basic event-management services, including consolidating and persisting raw events from multiple, heterogeneous sources and distributing those events to event consumers. It 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.

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.

Related information

[Administering the Common Event Infrastructure](#)

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 the server. With WebSphere Process Server, you also can use business calendars.

One component-based framework addresses all styles of integration.

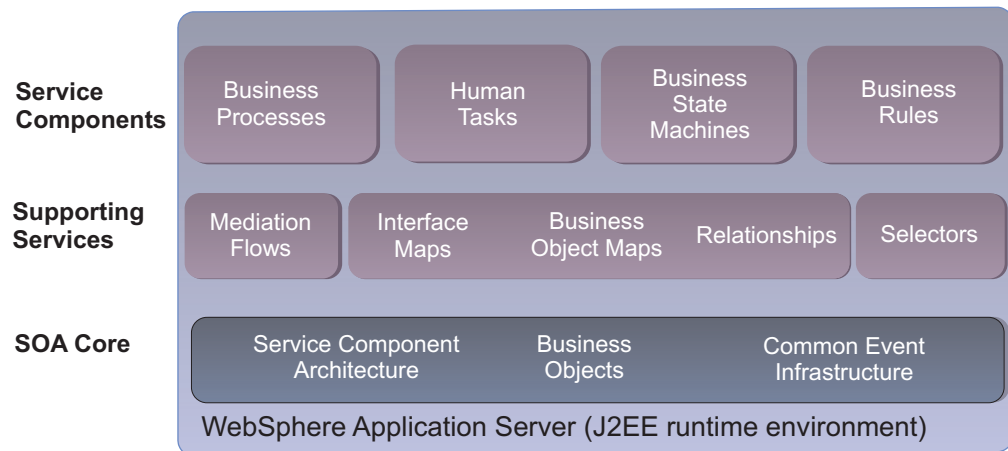


Figure 4. WebSphere Process Server component-based framework

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 to WebSphere Process Server in either a module or a mediation module.

Related concepts

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.

Interface maps

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

Interface maps are supporting service components in 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 are a way of relating 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. One business object becomes the source and another becomes the target. The business object map maps the source and target. Business object maps support 1-to-n, many-to-1 and many-to-n mappings among business objects. This includes mapping the business data and the aspects associated with the business object, such as verb.

Developers create and edit the business object maps in IBM WebSphere Integration Developer. During run time, the maps resolve how data is represented between the source and target business objects. You can monitor map events during run time in WebSphere Process Server.

Related concepts

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.

Relationships

Relationships are services used to model and maintain associations between business objects and other data.

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

A *relationship* is an association between two or more data entities in the business integration system. Most often, these entities are business objects. Relationships are used to transform data that is equivalent across business objects but is represented differently.

In WebSphere Process Server, relationship manager is a tool for manually manipulating relationship data to correct errors found in automated relationship management or to provide more complete relationship information. In particular, it provides a facility for retrieving and 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.

Related concepts



Administering relationships

The relationship manager is a tool for manually controlling and 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.

Selectors

Selectors provide flexibility at points in the processing of service components during run time.

Selectors, also called selector components, are supporting services in IBM WebSphere Process Server that take one invocation and allow different targets to be called based on the selection criteria.

A *selector component* is a component that provides a means of interposing a dynamic selection mechanism between the client application and a set of target implementations.

Selectors allow additional flexibility beyond business rules. *Business rules*, a fundamental part of a businesses, 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.

Related concepts

 Overview of selector components

As businesses change, the business processes that drive them must change, too. Some of those changes may require that certain processes return different results than as originally designed without changing the design of the process. The selector component provides the framework for that flexibility.

Business calendars and timetables

Timetables, also called business calendars, define available time for a business year in an organization, allowing for office hours and holidays. Human tasks and business processes use the available time defined in timetables to schedule timeouts, when a task expires, when it is deleted, or when it is overdue. The Business Calendar Manager, available with Business Space, provides an environment for you and all runtime users of the application to view and modify the time intervals, depending on security roles.

A timetable is a schedule of times that indicates availability (such as Monday to Friday). For example, an organization can define its business days and holidays in a timetable that includes the following criteria: working hours of 9:00 a.m. – 5:00 p.m. Monday through Friday, with New Year’s Day, Memorial Day, Labor Day, Thanksgiving Day and Christmas Day as holidays.

Timetables are created in WebSphere Business Modeler, and business calendars are created in WebSphere Integration Developer. Both are deployed to WebSphere Process Server as XML artifacts in Service Component Architecture (SCA) modules. Modules containing business calendars and timetables are deployed the same way as other modules – as an enterprise archive (EAR) file – either from WebSphere Integration Developer using the unit test environment or from the administrative console.

The timetable, or business calendar, capability can be used by business processes and human tasks. Business processes use it to schedule timeouts. Human tasks use it to schedule when a task expires or when it will be deleted or is overdue.

At run time, a specified business calendar, or a timetable, is loaded for components that are clients of the service. It uses the current time and the delta to calculate the time for the components. For example, if a claim is identified as overdue if not completed by 3 working days, and the claim is assigned to an employee on a Friday, May 16, before a public holiday, the process will not be overdue until 6 days later, on Thursday, May 22 – taking into account when the office is closed for Saturday, Sunday, and the public holiday.

During runtime, if you have configured Business Space, you and all users of the applications can use the Business Calendar Manager widget to view and edit timetables. This includes business calendars that were created in WebSphere Integration Developer and timetables created in WebSphere Business Modeler that were deployed to WebSphere Process Server version 6.2. For business calendars developed in WebSphere Integration Developer version 6.1.2, if you want them to be available in Business Calendar Manager, you must import modules into WebSphere Integration Developer version 6.2 and then deploy them to WebSphere Process Server version 6.2.

Each timetable has security roles associated with it: owner, reader, and writer. Users who have the owner role can modify timetables they own in Business Calendar Manager, and they can grant writer and reader roles to other users by using Security Manager widget in Business Space. Users who have writer roles can modify timetables by creating and modifying time intervals in Business Calendar Manager. Users who have reader roles can view timetables and time intervals but cannot modify them.

The business calendar schema is flexible enough to allow multiple types of timetables. In the flat model, all of the metadata is in one timetable file. In a hierarchical model, you can build small timetables that are complete on their own and then build a top level timetable that references other timetables.

Timetables have dates with offsets based on Greenwich Mean Time (GMT). For example, if a timetable is designated for working hours of 9 a.m. to 5 p.m. in New York, it is set with a GMT offset of GMT-5, which keeps the working hours the same, even if the module is moved to a server in California. In a flat timetable, all the dates use the same offset. In a hierarchical timetable, which references other timetables, the individual timetables can have different GMT offsets.

The scope of a business calendar, or a timetable, is the module into which it is deployed.

For more information about creating business calendars in WebSphere Integration Developer, see "Working with business calendars" in the WebSphere Integration Developer documentation.

For more information about using Business Calendar Manager, see the online help for Business Calendar Manager in Business Space.

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.

In the Service Component Architecture (SCA), a *service component*, also called an SCA component, defines a service implementation. Service components each have an interface and can be wired together to form a module deployed to WebSphere Process Server.

This creates a flexible runtime environment and enables changing any part of an application without affecting the other parts. For example, it is possible to replace a human task representing an approval with a business rule representing automatic approval – simply by replacing the service components in the assembly diagram – without changing either a business process or the caller of the business process.

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

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

In addition, service components can interact with other applications on enterprise information systems (EIS) with IBM WebSphere Adapters.

On top of the runtime infrastructure of supporting services and the service-oriented architecture core, WebSphere Process Server offers a variety of ready-to-use SCA components that can be used in integration applications. Mediation flows are implemented in an SCA component (a mediation flow component) but for WebSphere Process Server modules they provide a supporting service role.

One component-based framework addresses all styles of integration.

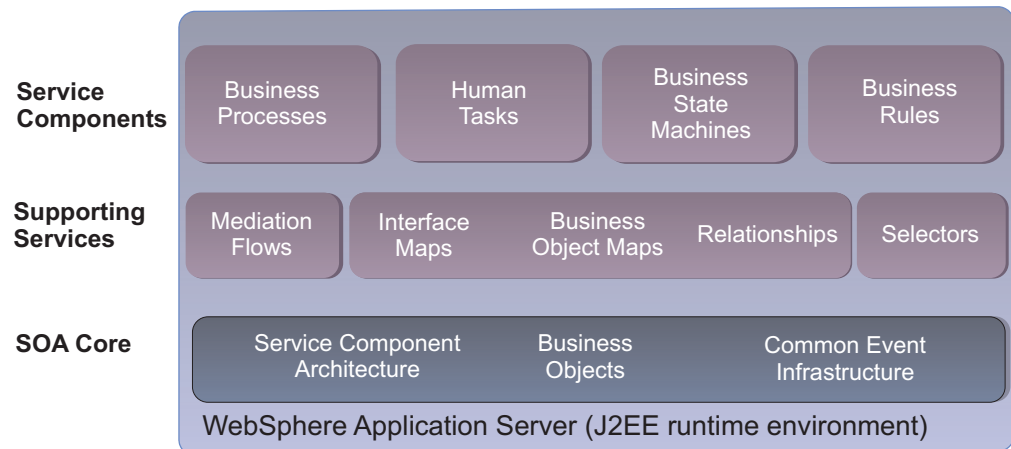


Figure 5. WebSphere Process Server component-based framework

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 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, or import them from a business model you created in WebSphere Business Modeler.

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.

Related information

 [About business processes](#)

Human tasks

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

A human task is a unit of work done by a human that often involves interaction with other services, and thus becomes a task within a larger business goal.

The Human Task Manager, available in WebSphere Process Server, supports creation and tracking of tasks during run time. 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 service components allow role-based task assignment, invocation and escalation.

Related information

 [Human tasks](#)

Business state machines

Business state machines are service components that allow you to represent business processes based on states and events instead of a sequential business process model.

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

You create and edit business state machines in IBM WebSphere Integration Developer, and you monitor them during run time in IBM WebSphere Process Server.

Related reference

 [Business state machine events](#)

The event types available for the business state machine component are listed.

Business rules


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

A *business rule* is a representation of how business policies or practices apply to a business activity. It is anything that controls the behavior of, or imposes a structure on a business practice. A rule can enforce business policy, establish common guidelines within an organization, or control access in a business environment.

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.

Related concepts

 [Overview of business rules](#)

Use business rules to control the behavior of a business practice.

Deployment environments in WebSphere Process Server

WebSphere Process Server allows you to manage the deployment environment for your Service Component Architecture (SCA) modules as one collection of servers. WebSphere Application Server Network Deployment capabilities included with WebSphere Process Server provide elements for this collection of servers.

The WebSphere Process Server environment includes a layout of interconnected servers, or topology, which supports SCA modules of your service applications. This topology consists of one server process running on one computer system, or it can consist of multiple server processes running on multiple computer systems. A *server process* is a runtime environment for components that are deployed as SCA modules. In WebSphere products, including WebSphere Process Server, a server process is a Java Virtual Machine (JVM).

If your environment consists of one server process on one system, the server process that is set up is called a *stand-alone server*. A stand-alone server does not have interconnections with other server processes, it has a capacity that is limited to the resources on that one computer system, and it does not include failover support. It is also the easiest environment to set up.

If your environment consists of multiple server processes, you most likely set up those processes as a *clustered* environment in a *cell*. A cell is a management domain of a distributed computing environment consisting of SCA modules and the resources needed to support them. A *deployment environment* is an environment in

which server processes, typically on different physical computer systems, are managed together. One deployment manager can manage multiple deployment environments.

Using a deployment environment with clusters provides the following benefits:

- **Ease of management:** You can have one view for configuring SCA modules, one view of the server processes that support the SCA modules, and one point of control for runtime actions for the SCA modules such as starting, stopping, creating, and deleting.
- **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.

Related tasks



Implementing a deployment environment

After designing a deployment environment, you will perform specific tasks to make that design a reality. Regardless which method you use to implement the deployment environment, you will perform the same general steps.

Business Space powered by WebSphere

WebSphere Process Server includes Business Space powered by WebSphere, which provides a common interface for application users to create, manage and integrate Web interfaces across the IBM WebSphere Business Process Management portfolio.

Business Space is a browser-based graphical user interface that lets application users customize content from products in the WebSphere Business Process Management portfolio. Application users (business users) are the users of the applications deployed on WebSphere Process Server.

Business Space provides an Asynchronous JavaScript™ and XML (AJAX) interface using mashup technology to enable business users to create and customize human task-centric user experiences. Mashups are Web pages created by combining Web applications (widgets), which mix together Web content to create novel interfaces. The widgets communicate with the WebSphere Process Server runtime using Representational State Transfer (REST) interactions with common Web formats, such as JavaScript Object Notation (JSON) and XML data.

Business users can customize Business Space widgets to view the runtime business data according to their preferences. Business Space administrators can create new spaces (collections of pre-configured pages), and mashup page content, in addition to the predefined scenarios that are shipped with Business Space. For example, these mashups act on specific business needs for the enterprise, such as assigning people to various tasks or adjusting business rules for different outcomes.

Business Space is shipped with WebSphere Process Server, Enterprise Service Bus, WebSphere Business Monitor, and WebSphere Business Modeler Publishing Server. Templates for predefined scenarios are enabled in Business Space when each product is installed. Business Space also includes information from WebSphere Business Services Fabric that business users can view and modify.

The following diagram shows the Business Space framework and products in the WebSphere Business Process Management portfolio. The top layer represents Business Space. The middle layer contains products that directly contribute content for Business Space: WebSphere Business Monitor, WebSphere Process Server, WebSphere Enterprise Service Bus, WebSphere Business Modeler Publishing Server, and WebSphere Business Services Fabric. The bottom layer represents products that indirectly contribute content through one of the middle-layer products: WebSphere Integration Developer and WebSphere Business Modeler.

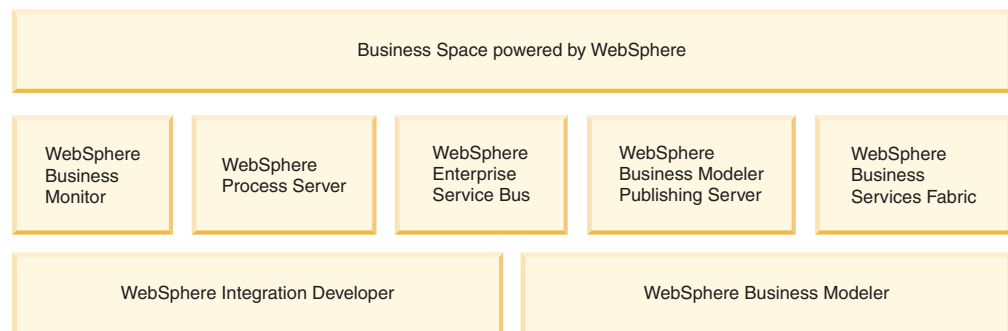


Figure 6. Business Space and Business Process Management products

For WebSphere Process Server applications, Business Space provides pages for your business users to work with runtime artifacts. Business users can create mashups with data that they can view and modify.

Business Space provided with WebSphere Process Server includes the Solution Management space, which is a collection of widgets for administrators to work with runtime artifacts. The space includes Health Monitor, for viewing a snapshot of the overall system health of your business solution; Business Calendar Manager, for viewing and modifying timetables that define available time for your business solution; and Security Manager, for setting owner, reader, and writer security roles.

Business Space provided with WebSphere Process Server also includes Managing Tasks and Workflows widgets for business users to view their own tasks that they created, tasks for members of their team, available tasks, and task information. They can create tasks and view human workflow diagrams that show the status of the tasks belonging to the same business process. Business Space also includes a Business Configuration page with a widget to work with business rules.

In addition to the widgets related to WebSphere Process Server applications, Business Space also includes Google Tools widgets and other widgets for viewing presentations, documents, spreadsheets, Web sites, and RSS feeds.

The Business Space framework is installed with WebSphere Process Server. For stand-alone server profiles, you can configure Business Space with either the Profile Management Tool or the administrative console. If you use deployment

environments or a remote database, you must configure Business Space using the administrative console Business Space Configuration page, or the Deployment Environment Configuration wizard.




To use the Managing Tasks and Workflows widgets, you must configure Business Process Choreographer. For more information, see "Configuring Business Process Choreographer" in the WebSphere Process Server information center. To work with Solution Management widgets, you must configure System Representational State Transfer (REST) services. For more information, see "Enabling Business Space widget endpoints on the administrative console" in the WebSphere Process Server information center.

After you have installed and configured Business Space for use with WebSphere Process Server, the Solution Management space is set up automatically in Business Space. You also can create your own space with the Solution Management template in Business Space Manager. See "Business Space concepts" in the Business Space documentation.

After you have installed and configured Business Space, users of your runtime environment can open it from the following URL: `http://host:port/BusinessSpace`, where *host* is the name of the host where your server is running and *port* is the port number for your server.

If your team is also working with a WebSphere Portal environment, you can configure Business Space widgets to be available in WebSphere Portal. For more information about this configuration, see the related task "Configuring widgets for WebSphere Portal."

Related tasks

-  [Configuring Business Space](#)
-  [Enabling Business Space widget endpoints on the administrative console](#)
-  [Configuring widgets for WebSphere Portal](#)

The enterprise service bus in WebSphere Process Server

WebSphere Process Server supports service applications with an integrated enterprise service bus.

Connecting services through an enterprise service bus

With an enterprise service bus (ESB), you can maximize the flexibility of an SOA. Participants in a service interaction are connected to the ESB, rather than directly to one another.

When the service requester connects to the ESB, the ESB takes responsibility for delivering its requests, using messages, to a service provider offering the required function and quality of service. The ESB facilitates requester-provider interactions and addresses mismatched protocols, interaction patterns, or service capabilities. An ESB can also enable or enhance monitoring and management. The ESB provides virtualization and management features that implement and extend the core capabilities of SOA.

The ESB abstracts the following features:

Location and identity

Participants need not know the location or identity of other participants. For example, requesters need not be aware that a request could be serviced by any of several providers; service providers can be added or removed without disruption.

Interaction protocol

Participants need not share the same communication protocol or interaction style. For example, a request expressed as SOAP over HTTP can be serviced by a provider that only understands SOAP over Java Message Service (JMS).

Interface

Requesters and providers need not agree on a common interface. An ESB reconciles differences by transforming request and response messages into a form expected by the provider.

Requesters and providers need not agree on a common interface

An ESB reconciles differences by transforming request messages into a form expected by the provider.

Qualities of (interaction) service

Participants, or systems administrators, declare their quality-of-service requirements, including authorization of requests, encryption and decryption of message contents, automatic auditing of service interactions, and how their requests should be routed (for example, optimizing for speed or cost).

Interposing the ESB between participants enables you to modulate their interaction through a logical construct called a *mediation*. Mediations operate on messages in-flight between requesters and providers. For example, mediations can be used to find services with specific characteristics that a requester is asking for, or to resolve interface differences between requesters and providers. For complex interactions, mediations can be chained sequentially.

An enterprise service bus, with mediations, performs the following actions between requester and service:

- *Routing* messages between services. An enterprise service bus offers a common communication infrastructure that can be used to connect services, and thereby the business functions they represent, without the need for programmers to write and maintain complex connectivity logic.
- *Converting* transport protocols between requester and service. An enterprise service bus provides a consistent, standards-based way to integrate business functions that use different IT standards. This enables integration of business functions that could not normally communicate, such as to connect applications in departmental silos or to enable applications in different companies to participate in service interactions.
- *Transforming* message formats between requester and service. An enterprise service bus enables business functions to exchange information in different formats, with the bus ensuring that the information delivered to a business function is in the format required by that application.
- *Handling* business events from disparate sources. An enterprise service bus supports event-based interactions in addition to the message exchanges to handle service requests.

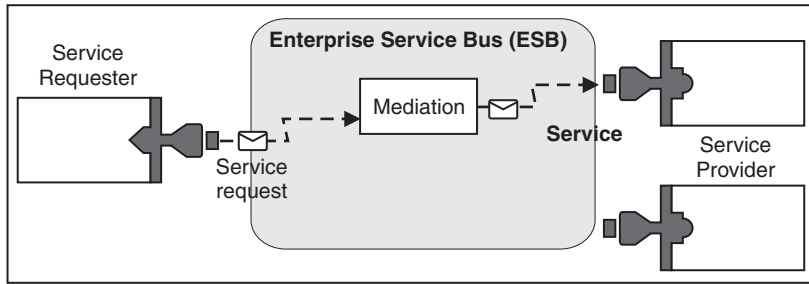


Figure 7. An enterprise service bus. The enterprise service bus is routing messages between applications, which are requesters or providers of services. The bus is converting transport protocols and transforming message formats between requesters and providers. In this figure, each application uses a different protocol (represented by the different geometric shapes of their connectors) and uses different message formats.

By using the enterprise service bus you can concentrate focus on your core business rather than your computer systems. You can change or add to the services when you need to; for example, to respond to changes in the business requirement, to add extra service capacity, or to add new capabilities. You can make the required changes by reconfiguring the bus, with little or no impact to existing services and applications that use the bus.

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.

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.

You can deploy WebSphere Enterprise Service Bus around your enterprise to extend the connectivity reach of the process integration solutions powered by separate installations of WebSphere Process Server or other integration solutions as part of a federated ESB. 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

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

Messaging or queue destination hosts

A messaging or queue destination host provides the messaging function within a server. A server becomes the messaging destination host when you configure it as the messaging target.

A messaging engine runs within a server. The messaging engine provides messaging functions and a connection point for applications to connect to the bus. Service Component Architecture (SCA) asynchronous communication, JMS imports and exports, asynchronous internal processing use message queues on the messaging engine.

The deployment environment connects the message source to the message target through the bus when the application modules are deployed. Knowing the message source and message target helps you determine what type of deployment environment you need.

Applications can store persistent data in a data store, which is a set of tables in a database or schema, or in a file store. The messaging engine uses an instance of a JDBC data source to interact with that database.

Configure the messaging destination host when you define your deployment environment by using **Server** from the administrative console or designate the server as the destination host during software installation.


Data stores:

Every messaging engine can use a data store, which is a set of tables in a database or schema that store persistent data.

All of the tables in the data store are held in the same database schema. You can create each data store in a separate database. Alternatively, you can create multiple data stores in the same database, with each data store using a different schema.

A messaging engine uses an instance of a JDBC data source to interact with the database that contains the data store for that messaging engine.

Related concepts

 Database configurations

WebSphere Process Server uses a number of database tables to hold, store and track information. Creating these database tables is part of the process of configuring the WebSphere Process Server. You can create these database tables during profile creation or you can choose to create them separately using scripts.

Data sources

Data sources provide the link between applications and relational databases.

Applications use a data source to obtain connections to a relational database. A data source is analogous to the J2EE Connector Architecture (JCA) connection factory, which provides connectivity to other types of enterprise information systems (EIS).

A data source is associated with a JDBC provider, which supplies the driver implementation classes that are required for JDBC connectivity with a specific type of database. Application components transact directly with the data source to obtain connection instances to your database. The connection pool that corresponds to each data source provides connection management.

You can create multiple data sources with different settings, and associate them with the same JDBC provider. For example, you might use multiple data sources to access different databases within the same database application. WebSphere Process Server requires JDBC providers to implement one or both of the following data source interfaces, which are defined by Sun Microsystems. These interfaces enable the application to run in a single-phase or two-phase transaction protocol.

- **ConnectionPoolDataSource** - a data source that supports application participation in local and global transactions, except two-phase commit transactions. When a connection pool data source is involved in a global transaction, transaction recovery is not provided by the transaction manager. The application is responsible for providing the backup recovery process if multiple resource managers are involved.
- **XADataSource** - a data source that supports application participation in any single-phase or two-phase transaction environment. When this data source is involved in a global transaction, the WebSphere Application Server transaction manager provides transaction recovery.

The following tables provide examples of typical stand-alone and typical deployment environment setups:

Table 2. Typical stand-alone environment setup

Datasource	Component	Scope	JNDI Name
WBI DataSource	CommonDB	Node	jdbc/WPSDB
SCA Application Bus ME data source	SCA ME	Server	jdbc/com.ibm.ws.sib/nlNode01.server1-SCA.APPLICATION.localHostNode01Cell.Bus
Business Process Choreographer data source	BPC	Server	jdbc/BPEDB
Business Process Choreographer ME data source	BPC ME	Server	jdbc/com.ibm.ws.sib/nlNode01.server1-BPC.localHostNode01Cell.Bus
event	CEI	Server	jdbc/cei
CEI ME data source	CEI ME	Server	jdbc/com.ibm.ws.sib/nlNode01.server1-CommonEventInfrastructure_Bus

Table 3. Typical deployment environment setup

Datasource	Component	Scope	JNDI Name
WBI DataSource	CommonDB	Cell	jdbc/WPSDB
SCA Application Bus ME data source	SCA ME	Cluster	jdbc/com.ibm.ws.sib/clusterone-SCA.APPLICATION.enduranceTestCell01.Bus
Business Process Choreographer data source	BPC	Cluster	jdbc/BPEDB

Table 3. Typical deployment environment setup (continued)

Datasource	Component	Scope	JNDI Name
Business Process Choreographer ME data source	BPC ME	Cluster	jdbc/com.ibm.ws.sib/clusterone-BPC.enduranceTestCell01.Bus
event	CEI	Cluster	jdbc/cei
CEI ME data source	CEI ME	Cluster	jdbc/com.ibm.ws.sib/clusterone-CommonEventInfrastructure_Bus

For more information on data sources, see “Data sources” in the WebSphere Application Server information center.

JDBC providers:

JDBC providers allow applications to interact with relational databases.

Applications use JDBC providers to interact with relational databases. The JDBC provider supplies the specific JDBC driver implementation class for access to a specific type of database. To create a pool of connections to that database, you associate a data source with the JDBC provider. Together, the JDBC provider and the data source objects are functionally equivalent to the J2EE Connector Architecture (JCA) connection factory, which provides connectivity with a non-relational database.

Refer to the examples of both Typical stand-alone environment setup and Typical deployment environment setup in the previous topic.

For more information on JDBC providers, see “JDBC providers” in the WebSphere Application Server information center.

Service integration buses for WebSphere Process Server

A service integration bus is a managed communication mechanism that supports service integration through synchronous and asynchronous messaging. A bus consists of interconnecting messaging engines that manage bus resources. It is one of the WebSphere Application Server technologies on which WebSphere Process Server is based.

Some buses are automatically created for use by the system, the Service Component Architecture (SCA) applications that you deploy, and by other components. You can also create buses to support service integration logic or other applications, for example, to support applications that act as service requesters and providers within WebSphere Process Server, or to link to WebSphere MQ.

A bus destination is a logical address to which applications can attach as a producer, consumer, or both. A queue destination is a bus destination that is used for point-to-point messaging.

Each bus can have one or more bus members, each of which is either a server or a cluster.

The *bus topology* is the physical arrangement of application servers, messaging engines, and WebSphere MQ queue managers and the pattern of bus connections and links between them that makes up your enterprise service bus.

Some service integration buses are created automatically to support WebSphere Process Server. Up to four buses are created when you create your deployment environment or configure a server or cluster to support SCA applications. These buses each have three authentication aliases that you must configure.

SCA system bus:

The *SCA system bus* is a service integration bus that is used to host queue destinations for Service Component Architecture (SCA) modules. The SCA run time, which supports mediation modules, uses queue destinations on the system bus as an infrastructure to support asynchronous interactions between components and modules.

The system bus is automatically created when you create a deployment environment or when you configure a server or cluster to support SCA applications. The system bus provides a scope within which resources, such as queue destinations, are configured for mediation modules and interaction endpoints. The bus enables message routing between endpoints. You can specify the quality of service for the bus, including priority and reliability.

The bus name is `SCA.SYSTEM.busID.Bus`. The authentication alias used for securing this bus is `SCA_Auth_Alias`.

SCA application bus:

The application bus destinations support the asynchronous communication of WebSphere Business Integration Adapters and other System Component Architecture components.

The application bus is automatically created when you create a deployment environment or when you configure a server or cluster to support SCA applications. The application bus is similar to service integration buses you might create to support service integration logic or other applications.

The bus name is `SCA.APPLICATION.busID.Bus`. The authentication alias used for securing this bus is `SCA_Auth_Alias`.

The Common Event Infrastructure bus:

The Common Event Infrastructure bus is used for transmitting common base events, asynchronously, to the configured Common Event Infrastructure server.

The bus name is `CommonEventInfrastructure_Bus`. The authentication alias used for securing this bus is `CommonEventInfrastructureJMSAuthAlias`

The Business Process Choreographer bus:

Use the Business Process Choreographer bus name and authentication for internal message transmission.

The Business Process Choreographer bus is used for transmitting messages internally and for business flow manager's Java Messaging Service (JMS) API.

The bus name is `BPC.cellName.Bus`. The authentication alias is `BPC_Auth_Alias`

Service applications and service modules

A service module is a Service Component Architecture (SCA) module that provides services in the run time. When you deploy a service module to WebSphere Process Server, you build an associated service application that is packaged as an Enterprise ARchive (EAR) file.

Service modules are the basic units of deployment and can contain components, libraries, and staging modules used by the associated service application. Service modules have exports and, optionally, imports to define the relationships between modules and service requesters and providers. WebSphere Process Server supports modules for business services and mediation modules. Both modules and mediation modules are types of SCA modules. A mediation module allows communication between applications by transforming the service invocation to a format understood by the target, passing the request to the target and returning the result to the originator. A module for a business service implements the logic of a business process. However, a module can also include the same mediation logic that can be packaged in a mediation module.

Deploying a service application

The process of deploying an EAR file containing a service application is the same as the process of deploying any EAR file. You can modify values for mediation parameters at deployment time. After you have deployed an EAR file containing an SCA module, you can view details about the service application and its associated module. You can see how a service module is connected to service requesters (through exports) and service providers (through imports).

Viewing SCA module details

The service module details that you can view depend upon the SCA module. They can include the following attributes.

- SCA module name
- SCA module description
- Associated application name
- SCA module version information, if the module is versioned
- SCA module imports:
 - Import interfaces are abstract definitions that describe how an SCA module accesses a service.
 - Import bindings are concrete definitions that specify the physical mechanism by which an SCA module accesses a service. For example, using SOAP/HTTP.
- SCA module exports:
 - Export interfaces are abstract definitions that describe how service requesters access an SCA module.
 - Export bindings are concrete definitions that specify the physical mechanism by which a service requester accesses an SCA module, and indirectly, a service.
- SCA module properties

Imports and import bindings

Imports define interactions between Service Component Architecture (SCA) modules and service providers. SCA modules use imports to permit components to

access external services (services that are outside the SCA module) using a local representation. Import bindings define the specific way that an external service is accessed.

If SCA modules do not need to access external services they are not required to have imports. Mediation modules usually have one or more imports that are used to pass messages or requests on to their intended targets.

Interfaces and bindings

An SCA module import needs at least one interface, and an SCA module import has a single binding.

- Import interfaces are abstract definitions that define a set of operations using Web Services Description Language (WSDL), an XML language for describing Web services. An SCA module can have many import interfaces.
- Import bindings are concrete definitions that specify the physical mechanism that SCA modules use to access an external service.

Supported import bindings

WebSphere Process Server supports the following import bindings:

- Web Service bindings permit components to invoke Web services. The supported protocols are SOAP/HTTP and SOAP/JMS. A Web services binding that uses SOAP/JMS, supports JMS using the default WebSphere Application Server default messaging provider in a point-to-point configuration. The SOAP/JMS binding does not support: generic JMS, MQ JMS, or the JMS broadcast mode.
- SCA Bindings connect SCA modules to other SCA modules. SCA bindings are also referred to as default bindings.
- Java Message Service (JMS) 1.1 Bindings permit interoperability with the WebSphere Application Server default messaging provider. JMS can exploit various transport types, including TCP/IP and HTTP or HTTPS. The JMS Message class and its five subtypes (Text, Bytes, Object, Stream, and Map) are automatically supported.
- WebSphere MQ JMS bindings permit interoperability with WebSphere MQ based JMS providers. The JMS Message class and its five subtypes (Text, Bytes, Object, Stream, and Map) are automatically supported. If you want to use WebSphere MQ as a JMS provider you might have WebSphere MQ JMS bindings.
- WebSphere MQ Bindings permit interoperability with WebSphere MQ. You can use WebSphere MQ bindings only with remote queue managers via a WebSphere MQ client connection; you cannot use them with local queue managers. You might have WebSphere MQ bindings if you want to communicate with native WebSphere MQ applications.
- Generic JMS bindings permit interoperability with third-party JMS providers that integrate with the WebSphere Application Server using the JMS Application Server Facility (ASF).
- WebSphere Adapters bindings permit interaction with Enterprise Information Systems (EIS).
- HTTP bindings permit you to access applications using the HTTP protocol.

Dynamic invocation of services

Services can be invoked through any supported import binding. A service is normally found at an endpoint specified in the import. This endpoint is called a

static endpoint. It is possible to invoke a different service by overriding the static endpoint. Dynamic override of static endpoints lets you invoke a service at another endpoint, through any supported import binding. Dynamic invocation of services also permits you to invoke a service, where the supported import binding does not have a static endpoint.

A specific configuration is used to control how dynamic invocation of services works. The configuration can be defined using a model import binding, or at invocation time.

The invocation target type is identified from the endpoint URL. An `sca` URL indicates an SCA module. An `http` or a `jms` URL indicates a web service. Using `http` in the URL does not mean that the endpoint is an HTTP service. Similarly, using `jms` in the URL does not mean that the endpoint is a JMS service.

Exports and export bindings

Exports define interactions between Service Component Architecture (SCA) modules and service requesters. SCA modules use exports to offer services to others. Export bindings define the specific way that an SCA module is accessed by service requesters.

Interfaces and bindings

An SCA module export needs at least one interface.

- Export interfaces are abstract definitions that define a set of operations using Web Services Description Language (WSDL), an XML language for describing Web services. An SCA module can have many export interfaces.
- Export bindings are concrete definitions that specify the physical mechanism that service requesters use to access a service. Usually, an SCA module export has one binding specified. An export with no binding specified is interpreted by the run time as an export with an SCA binding.

Supported export bindings

WebSphere Process Server supports the following export bindings:

- Web Service bindings permit exports to be invoked as Web services. The supported protocols are SOAP/HTTP and SOAP/JMS. A Web services binding that uses SOAP/JMS, supports JMS using the default WebSphere Application Server default messaging provider in a point-to-point configuration. The SOAP/JMS binding does not support: generic JMS, MQ JMS, or the JMS broadcast mode.
- SCA bindings connect SCA modules to other SCA modules. SCA bindings are also referred to as default bindings.
- Java Message Service (JMS) 1.1 bindings permit interoperability with the WebSphere Application Server default messaging provider. JMS can exploit various transport types, including TCP/IP and HTTP(S). The JMS Message class and its five subtypes (Text, Bytes, Object, Stream, and Map) are automatically supported.
- WebSphere MQ JMS bindings permit interoperability with WebSphere MQ based JMS providers. The JMS Message class and its five subtypes (Text, Bytes, Object, Stream, and Map) are automatically supported. If you want to use WebSphere MQ as a JMS provider you might have WebSphere MQ JMS bindings.
- WebSphere MQ bindings permit interoperability with WebSphere MQ. A remote (or client) connection is the type of connection needed to connect to an MQ

queue manager on a remote machine. A local (or bindings) connection is a direct connection to WebSphere MQ. This can only be used for a connection to a MQ queue manager on the same machine. WebSphere MQ will permit both types of connection, but MQ bindings only support the "remote" (or "client") connection.

- Generic JMS bindings permit interoperability with third-party JMS providers that integrate with the WebSphere Application Server using the JMS Application Server Facility (ASF).
- WebSphere Adapters bindings permit interaction with Enterprise Information Systems (EIS).
- HTTP bindings permit exports to be accessed using the HTTP protocol.

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. You are able to route messages to different service providers and to amend message content or form. 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, from the WebSphere Process Server administrative console, without having to redeploy the module.

Components of mediation modules

Mediation modules contain the following items:

- Imports, which define interactions between SCA modules and service providers. They allow SCA modules to call external services as if they were local. You can view mediation module imports from WebSphere Process Server and modify the binding.
- Exports, which define interactions between SCA modules and service requesters. They allow an SCA module to offer a service and define the external interfaces (access points) of an SCA module. You can view mediation module exports from WebSphere Process Server.

- SCA components, which are building blocks for SCA modules such as mediation modules. You can create and customize SCA modules and components graphically, using WebSphere Integration Developer. After you deploy a mediation module you can customize certain aspects of it from the WebSphere Process Server administrative console, without having to redeploy the module.

Usually, mediation modules contain a specific type of SCA component called a *mediation flow component*. Mediation flow components define mediation flows.

A mediation flow component can contain none, one, or a number of mediation primitives. WebSphere Process Server supports a supplied set of mediation primitives that provide functionality for message routing and transformation. If you need additional mediation primitive flexibility, you can use the Custom Mediation primitive to call custom logic.

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 might need transforming to SOAP/HTTP before sending on.

Note: You can view and make certain changes to mediation modules from WebSphere Process Server. However, you cannot view or change the SCA

components inside a WebSphere Process Server module. Use WebSphere Integration Developer to customize SCA components.

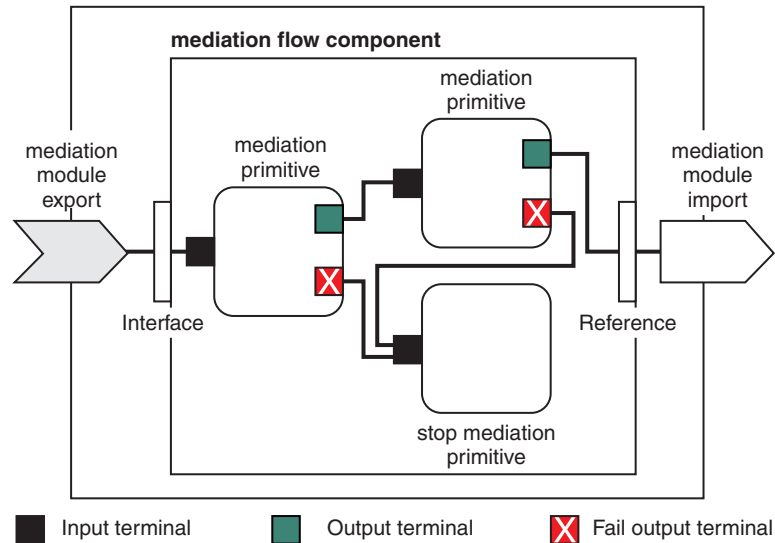


Figure 8. Simplified example of a mediation module. The mediation module contains one mediation flow component, which contains mediation primitives.

- Properties

Mediation primitives have properties, some of which can be displayed in the administrative console as additional properties of an SCA module.

For mediation primitive properties to be visible from the WebSphere Process Server administrative console, the integration developer must promote the properties. 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, because modifying them can affect the mediation flow in such a way that the mediation module needs to be redeployed. 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.

Generally, mediation flows use property changes immediately. However, if property changes occur in a deployment manager cell, they take effect on each node as that node is synchronized. Also, mediation flows that are in-flight continue to use previous values.

Note: From the administrative console, you can only change property values, not property groups, names or types. If you want to change property groups, names or types, you must use WebSphere Integration Developer.

- A mediation module or dependent library may also define subflows. A subflow encapsulates a set of mediation primitives wire together as a reusable piece of integration logic. A primitive can be added to a mediation flow to invoke a subflow.

Deploying mediation modules

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

Example of EAR file, containing a mediation module

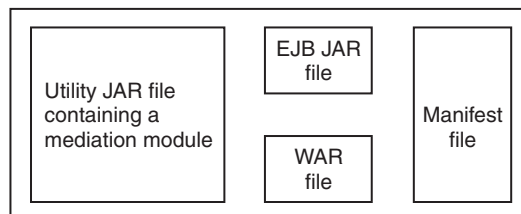


Figure 9. Simplified example of an EAR file containing a mediation module. The EAR file contains JARs. The utility JAR file contains a mediation module.

Mediation primitives

Mediation flow 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 flow 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 or modules deployed into WebSphere Process Server. If you need special mediation capabilities, you can develop your own custom mediation primitives.

A mediation primitive defines an “in” operation that processes or handles messages that are represented by service message objects (SMOs). A mediation primitive can also define “out” operations that send messages to another component or module.

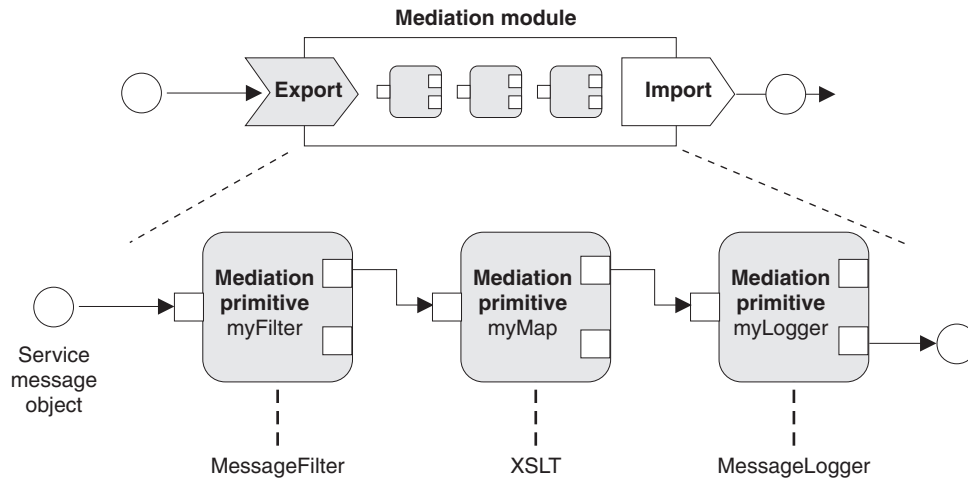


Figure 10. Mediation module containing three mediation primitives

You can use WebSphere Integration Developer to configure mediation primitives and set their properties. Some of these properties can be made visible to the runtime administrator by promoting them. Any mediation primitive property that can be promoted can also be a dynamic property. A dynamic property can be overridden, at run time, using a policy file.

WebSphere Integration Developer also allows you to graphically model and assemble mediation flow components from mediation primitives, and assemble mediation modules or modules from mediation flow components. The administrative console refers to mediation modules and modules as SCA modules.

WebSphere Integration Developer also allows the definition of subflows in modules or their dependent libraries. A subflow can contain any mediation primitive except for the Policy Resolution mediation primitive. A subflow is invoked from a request or response flow, or from another subflow using the Subflow mediation primitive. Properties promoted from mediation primitives in a subflow are exposed as properties on the Subflow mediation primitives. These may then be promoted again until they reach the module level at which point they can then be modified by the runtime administrator.

Supported mediation primitives

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

Business Object Map

Transforms messages.

- Defines message transformations using a business object map, which can be reused.
- Allows you to define message transformations graphically, using the business object map editor.
- Can alter the content of a message.
- Can transform an input message type to a different output message type.

Custom Mediation

Allows you to implement your own mediation logic in Java code. The

Custom Mediation primitive combines the flexibility of a user-defined mediation primitive, with the simplicity of a pre-defined mediation primitive. You can create complex transformations and routing patterns by:

- Creating Java code.
- Creating your own properties.
- Adding new terminals.

You can call a service from a Custom Mediation primitive, but the Service Invoke mediation primitive is designed to call services and provides additional functionality, such as retry.

Data Handler

Allows you to transform a part of a message. It is used to convert an element of a message from a physical format to a logical structure or a logical structure to a physical format. The primary usage of the primitive is to convert a physical format, such as a Text string within a JMS Text Message object, into a logical Business Object structure and back again. This mediation is commonly used to:

- Transform a section of the input message from a defined structure to another - an example of this would be were the SMO includes a string value that is comma delimited and you want to parse this into a specific Business Object.
- Alter the message type – an example would be when a JMS export has been configured to use a JMS basic typed data binding and within the mediation module the integration developer decides that the content should be inflated to a specific BO structure.

Database Lookup

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

- You must set up a database, data source, and any server authentication settings for the Database Lookup mediation primitive to use. Use the administrative console to help you do this.
- 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 simple XML schema type, or an XML schema type that extends a simple XML schema type.

Endpoint Lookup

Allows for the 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). The WSRR registry can be local or remote.
- You make registry changes from the WSRR administrative console.
- WebSphere Process Server needs to know which registry to use, therefore, you must create WSRR access definitions using the WebSphere Process Server administrative console.

Event Emitter

Enhances monitoring by letting you send events from inside a mediation flow component.

- You can suspend the mediate action by deselecting the checkbox.
- You can view Event Emitter events using the Common Base Events (CBE) browser on WebSphere Process Server.

- You should only send events at a significant point in a mediation flow, for performance reasons.
- You can define the parts of the message that the event contains.
- The events are sent in the form of Common Base Events and are sent to a Common Event Infrastructure server.
- To fully use the Event Emitter information, event consumers need to understand the structure of the Common Base Events. The Common Base Events has an overall schema, but this does not model the application specific data, which is contained in the extended data elements. To model the extended data elements, the WebSphere Integration Developer tools generate a Common Event Infrastructure 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.

Fail Stops a particular path in the flow, and generates an exception.

Fan In Helps aggregate (combine) messages.

- Can only be used in combination with the Fan Out mediation primitive.
- Together, the Fan Out and Fan In mediation primitives allow aggregation of data into one output message.
- The Fan In mediation primitive receives messages until a decision point is reached, then one message is output.
- The shared context should be used to hold aggregation data.

Fan Out

Helps split and aggregate (combine) messages.

- Together, the Fan Out and Fan In mediation primitives allow aggregation of data into one output message.
- In iterate mode, the Fan Out mediation primitive lets you iterate through a single input message that contains a repeating element. For each occurrence of the repeating element, a message is sent.
- The shared context should be used to hold aggregation data.

HTTP Header Setter

Provides a mechanism for managing headers in HTTP messages.

- Can create, set, copy, or delete HTTP message headers.
- Can set multiple actions to change multiple HTTP headers.

MQ Header Setter

Provides a mechanism for managing headers in MQ messages.

- Can create, set, copy, or delete MQ message headers.
- Can set multiple actions to change multiple MQ headers.

SOAP Header Setter

Provides a mechanism for managing headers in SOAP messages.

- Can create, set, copy, or delete SOAP message headers.
- Can set multiple actions to change multiple SOAP headers.

Message Element Setter

Provides a simple mechanism for setting the content of messages.

- Can change, add or delete message elements.
- Does not change the type of the message.
- The data in the value columns must be either a simple XML schema type, or an XML schema type that extends a simple XML schema type.

Message Filter

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

- You can suspend the mediate action by deselecting the checkbox.

Message Logger

Logs messages in a relational database or through your own custom logger. The messages are stored as XML, therefore, data can be post-processed by XML-aware applications.

- You can suspend the mediate action by deselecting the checkbox.
- The relational database schema (table structure) is defined by IBM.
- By default, the Message Logger mediation primitive uses the Common database. The runtime maps the data source at `jdbc/mediation/messageLog` to the Common database.
- You can set Handler implementation classes to customize the behavior of the custom logger. Optionally, you can provide Formatter implementation classes, Filter implementation classes, or both to customize the behavior of the custom logger.

Policy Resolution

Allows for the dynamic configuration of requests, by searching for service endpoints, and associated policy files, in a repository.

- You can use a policy file to dynamically override the promoted properties of other mediation primitives.
- Service endpoint information and policy information is retrieved from a WebSphere Service Registry and Repository (WSRR). The WSRR registry can be local or remote.
- You make registry changes from the WSRR administrative console.
- WebSphere Process Server needs to know which registry to use, therefore, you must create WSRR access definitions using the WebSphere Process Server administrative console.

Service Invoke

Calls a service from inside a mediation flow, rather than waiting until the end of the mediation flow and using the callout mechanism.

- If the service returns a fault, you can retry the same service or call another service.
- The Service Invoke mediation primitive is a powerful mediation primitive that can be used on its own for simple service calls, or in combination with other mediation primitives for complex mediations.

Set Message Type

During integration development, lets you treat weakly-typed message fields as though they are strongly-typed. A field is weakly-typed if it can contain more than one type of data. A field is strongly-typed if its type and internal structure are known.

- At runtime, the Set Message Type mediation primitive lets you check that the content of a message matches the data types you expect.

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

Type Filter

Allows you to direct messages down a different path of a flow, based on their type.

XSL Transformation

Transforms messages.

- Allows you to perform Extensible Stylesheet Language (XSL) transformations.
- You transform messages using an XSLT 1.0 transformation. The transformation operates on an XML serialization of the message.

Dynamic routing

You can route messages in various ways using endpoints defined at integration time or endpoints determined, dynamically, at run time.

Dynamic routing covers message routing where the flow is dynamic but all possible endpoints are predefined in a Service Component Architecture (SCA) module, and message routing where the flow is dynamic and the endpoint selection is also dynamic. In the latter case, the service endpoints are selected from an external source, at run time.

Dynamic endpoint selection

The run time has the capability to route messages to an endpoint address identified by a message header element. This message header element can be updated by mediation primitives, in a mediation flow. The endpoint address could be updated with information from a registry, a database, or with information from the message itself.

In order for the run time to implement dynamic routing on a request, the SCA module must have the Use dynamic endpoint if set in the message header property set. Integration developers can set the Use dynamic endpoint if set in the message header property or they can promote it (make it visible at run time), so that the runtime administrator can set it. You can view module properties in the Module Properties window. To see the window, click **Applications** → **SCA Modules** → **Module Properties**. The integration developer gives promoted properties alias names, and these are the names displayed on the administrative console.

Registry

You can use IBM WebSphere Service Registry and Repository (WSRR) to store service endpoint information, and then create SCA modules to retrieve endpoints from the WSRR registry.

When you develop SCA modules, you use the Endpoint Lookup mediation primitive to allow a mediation flow to query a WSRR registry for a service endpoint, or a set of service endpoints. If an SCA module retrieves a set of endpoints then it must use another mediation primitive to select the preferred one.

Mediation policy control of service requests

You can use mediation policies to control mediation flows between service requesters and service providers.

You can control mediation flows using mediation policies stored in IBM WebSphere Service Registry and Repository (WSRR). The implementation of service policy management in WSRR is based on the Web Services Policy Framework (WS-Policy).

In order to control service requests using mediation policies, you need to have suitable Service Component Architecture (SCA) modules and mediation policy documents in your WSRR registry.

How to attach a mediation policy to a service request

When you develop an SCA module that needs to make use of a mediation policy, you must include a Policy Resolution mediation primitive in the mediation flow. At run time, the Policy Resolution mediation primitive obtains mediation policy information from the registry.

Note: Therefore, an SCA module must contain a mediation flow component in order to support mediation policy control of service requests.

In the registry, you can attach one or more mediation policies to an SCA module. Attached mediation policies could be used (are in scope) for all service messages processed by that SCA module. The mediation policies can have policy attachments that define conditions. Mediation policy conditions allow different mediation policies to apply in different contexts. In addition, mediation policies can have classifications, which can be used to specify a governance state.

WebSphere Service Registry and Repository

The WebSphere Service Registry and Repository (WSRR) product allows you to store, access, and manage information about service endpoints and mediation policies. You can use WSRR to make your service applications more dynamic, and more adaptable to changing business conditions.

Introduction

Mediation flows can use WSRR as a dynamic lookup mechanism, providing information about service endpoints or mediation policies.

To configure access to WSRR, you create WSRR definition documents using the administrative console. Alternatively, you can use the WSRR administration commands from the wsadmin scripting client. WSRR definitions and their connection properties are the mechanism used to connect to a registry instance, and retrieve a service endpoint or mediation policy.

Service endpoints

You can use WSRR to store information about services that you already use, that you plan to use, or that you want to be aware of. These services might be in your systems, or in other systems. For example, a service application can use WSRR to locate the most appropriate service to satisfy its functional and performance needs.

When you develop an SCA module that needs to access service endpoints from WSRR, you must include an Endpoint Lookup mediation primitive in the mediation flow. At run time, the Endpoint Lookup mediation primitive obtains service endpoints from the registry.

Mediation policies

You can also use WSRR to store mediation policy information. Mediation policies can help you to control service requests, by dynamically overriding module properties. If WSRR contains an SCA module and attached mediation policies, the mediation policies have the potential to override module properties. If you want different mediation policies to apply in different contexts, you can create mediation policy conditions.

Note: Mediation policies are concerned with the control of mediation flows, and not with security.

When you develop an SCA module that needs to make use of a mediation policy, you must include a Policy Resolution mediation primitive in the mediation flow. At run time, the Policy Resolution mediation primitive obtains mediation policy information from the registry.

Message Service clients

WebSphere Process Server provides Message Service clients for C/C++ and .NET that enable non-Java applications to connect to the enterprise service bus.

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.

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

WebSphere Adapters

WebSphere Adapters provide a service-oriented approach to integration with Enterprise Information Systems (EIS).

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. They are then exported as an enterprise archive (EAR) file and deployed on WebSphere Process Server.

WebSphere Adapters include the following:

- IBM WebSphere Adapter For Email
- IBM WebSphere Adapter For Flat Files
- IBM WebSphere Adapter For FTP
- IBM WebSphere Adapter for JDBC
- IBM WebSphere Adapter for JD Edwards EnterpriseOne
- IBM WebSphere Adapter for Oracle E-Business Suite
- IBM WebSphere Adapter for Siebel Business Applications

- IBM WebSphere Adapter for SAP Software

Figure 11 shows a WebSphere Adapter managing the connectivity between a J2EE component supported by WebSphere Process Server and the EIS. The WebSphere Adapter resides inside WebSphere Process Server.

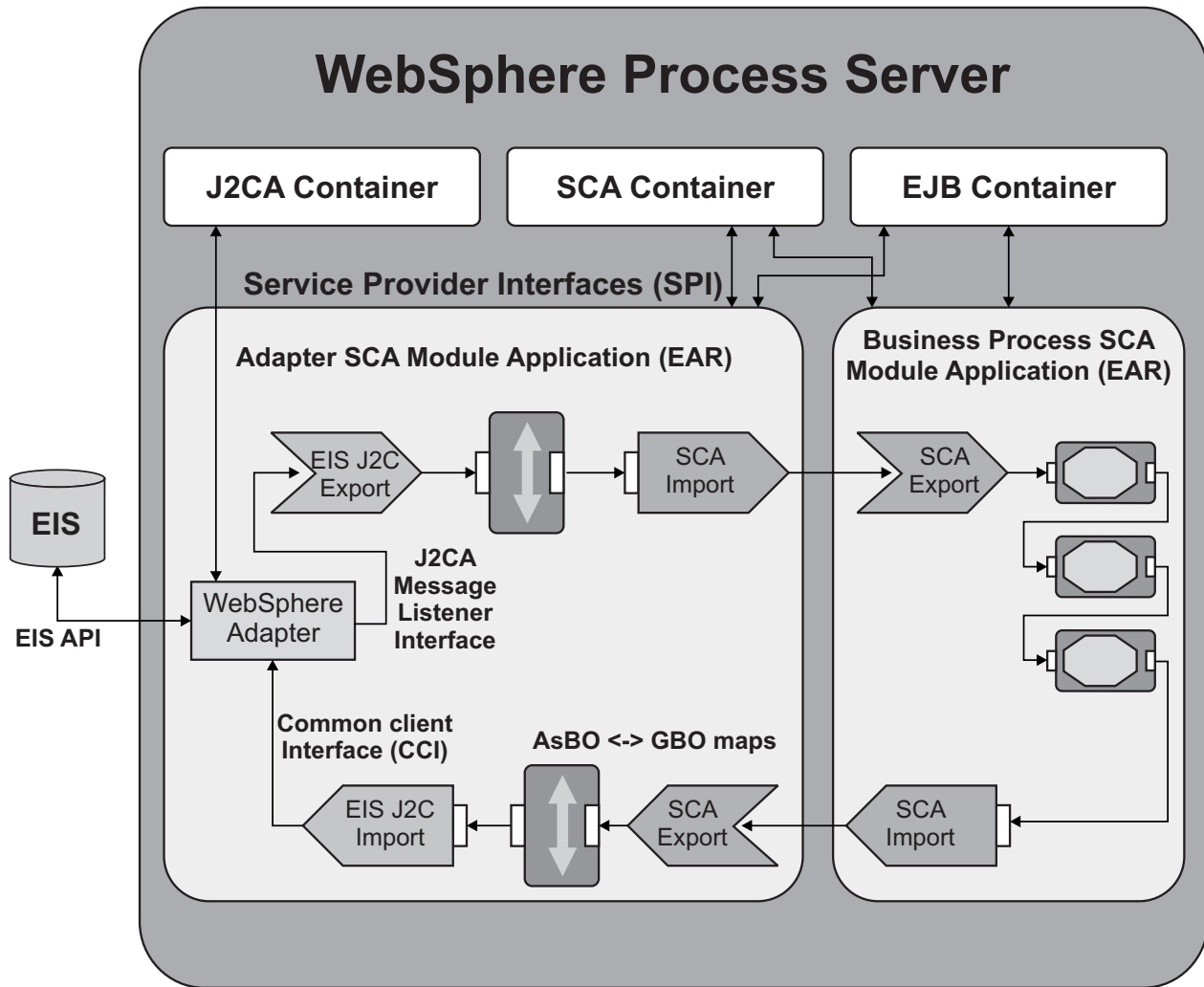


Figure 11. Detailed schematic of a WebSphere Adapter.

Development and deployment of applications on WebSphere Process Server

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.

Developing service applications

IBM WebSphere Integration Developer is the separate development environment for WebSphere Process Server. In addition to the WebSphere Integration Developer development environment, Service Component Architecture APIs are published for

developers. You can also 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.

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. You 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 if the interface of the changed module stays the same.

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.

For more information about developing modules for use with WebSphere Process Server, see *Developing for WebSphere Process Server*.

For more information about using WebSphere Integration Developer to develop integration applications, see the *WebSphere Integration Developer* documentation.

Deploying service applications

Deploying is the act of enabling your applications – your SCA modules – in either a test or a production environment. While the concept of deployment 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 SCA modules on a test server before committing them to the production environment, use WebSphere Integration Developer to deploy the modules into a test environment, and to package them 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 to deploy and manage the components of service integration packages. For more information about deploying applications into WebSphere Process Server, see *Deploying modules*.

If you need to deploy many application files, which means installing many SCA modules, you may want to use a batch file. For more information about batch files, see “Deploying applications using Apache Ant tasks”.

Related tasks

Deploying a module

You can deploy a module or a mediation module, as generated by WebSphere® Integration Developer, into a production WebSphere Process Server environment using these steps.

Deploying applications using Apache Ant tasks

ANT tasks allow you to define the deployment of multiple applications to WebSphere Process Server and have them run unattended on a server.

Related information

Developing modules

Migration to WebSphere Process Server

In this release, you can migrate your installed applications and profile configurations from prior versions of IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus to WebSphere Process Server version 6.2 using version-to-version migration tools. Version-to-version migration requires that you install the new version of the product alongside the older product, then run migration tools to migrate the existing applications and configurations to the new product. You cannot use updates (in-place upgrades) to migrate from previous releases to WebSphere Process Server version 6.2.

In addition, you can migrate applications and configuration data from certain IBM products that existed before WebSphere Process Server, such as WebSphere InterChange Server, WebSphere Business Integration Server Express, WebSphere Studio Application Developer Integration Edition, and WebSphere MQ Workflow.

Related information

Migrating to WebSphere Process Server

Migrating refers to the process of moving from one product to another or one version of a product to another while preserving product configuration information and user applications, thus enabling the existing applications and configuration data to be used in the new environment. You can migrate to WebSphere Process Server from certain other IBM products or from an earlier version of WebSphere Process Server to a later version such as version 6.2.

Administration of applications on WebSphere Process Server

Administering IBM WebSphere Process Server involves preparing, monitoring, and modifying the environment into which Service Component Architecture (SCA) modules are deployed as applications and resources, and working with the applications and resources themselves.

For more information about administering applications, see the *Administering WebSphere Process Server* 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. The WebSphere Process Server administrative console is part of the Integrated Solutions Console framework in general, and the WebSphere Application Server administrative console in particular. As a result, many administrative tasks (for example, setting security, viewing logs, and installing applications) are the same for both WebSphere Process Server and WebSphere Application Server.

- 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.

- WebSphere administrative (`wsadmin`) scripting program

The `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.

- 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.

- 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 processes 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 Explorer also includes an optional reporting function. You can use Process Choreographer Explorer reporting function to create reports on processes that have been completed. You can then use these reports to evaluate the effectiveness and reliability of your processes and activities. You can also use the reporting function to view the status of running processes.

- Business rules manager

The business rules manager is a Web-based tool that assists the business analyst in browsing and modifying business rule values. The tool is an option of WebSphere Process Server that you can select to install at profile creation time or after the initial installation of the server.

Related concepts

 The administrative console

The administrative console is a browser-based interface used to administer applications, services, and other resources at a cell, node, server, or cluster scope. You can use the console with stand-alone servers and with deployment managers that manage all servers in a cell in a networked environment.

Related information

 Administering WebSphere Process Server

Administering involves preparing, monitoring, and modifying the runtime environment into which applications, their modules, and resources are deployed, and managing those applications, modules, and resources within the runtime environment.

Administrative control of mediation processing

You can control mediation flows between service requesters and service providers, administratively.

You can control mediation flows by changing module properties. The module properties set the values of mediation primitive properties.

Module properties

You can change the properties of Service Component Architecture (SCA) modules that contain mediation flows. You can make changes in the following ways:

- When you install an application.
 - From the administrative console
 - Using an administrative command
- When you administer an application.
 - From the administrative console
 - Using an administrative command

The properties that you can change are those that have been promoted from WebSphere Integration Developer. Any property that you promote is also a dynamic property (which means that it can be overridden at run time using a mediation policy).

The module properties displayed on the administrative console, can do the following:

- Change the values of properties in a mediation flow.
- Provide default values for mediation flows that use mediation policies. (Promoted property values are used when there are no suitable mediation policy values. For further information, see the mediation policy model.)

Promoted properties always have a name, a type, and a value; you can change the value, administratively.

In addition, promoted properties can belong to a group (property groups are introduced in Version 6.2). Property groups can do the following:

- Separate multiple properties that have the same name. An administrator sets property values inside groups. You might have one group for properties on the request flow, and another group for properties on the response flow.

- Set multiple properties (of the same type) under one name. If the integration developer promotes two properties with the same alias name and group, then the administrator can set their values together. You might have logging for the request flow and the response flow, and want to set both at the same time.
- Map to a namespace in a mediation policy.

Security on WebSphere Process Server

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

Securing the WebSphere Process Server environment involves enabling administrative security, enabling application security, creating profiles with security, and restricting access to critical functions to selected users.

Related information

 [Securing applications and their environment](#)

Securing the WebSphere Process Server environment involves enabling administrative security, enabling application security, creating profiles with security, and restricting access to critical functions to selected users.

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.

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.

Related information



Monitoring

Monitoring enables you to assess performance, to troubleshoot problems, and evaluate the overall processing progress of service components that make up the applications deployed on your system.

Samples

Samples help you learn how to accomplish your goals with WebSphere Process Server.

WebSphere Process Server samples are available from the Samples Gallery, which you can install with the product.

WebSphere Process Server samples are also included in the Business Process Management samples at <http://publib.boulder.ibm.com/bpcsamp/index.html>.

Installing and accessing the Samples Gallery

Samples of integration application artifacts are available in the Samples Gallery, an option to install when you install this product.

About this task

The Samples Gallery contains samples of simple artifacts such as those generated by IBM WebSphere Integration Developer and deployed on IBM WebSphere Process Server. Other Business Process Management samples are available at <http://publib.boulder.ibm.com/bpcsamp/index.html>.

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

Procedure

1. Install WebSphere Process Server and select the samples package on the feature selection panel, and create a profile as part of the product installation.

Note: If you are installing WebSphere Process Server on top of WebSphere Application Server, the base WebSphere Application Server Samples Gallery must be installed in order for you to use the samples.

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

2. Start the server.
3. Start the Samples Gallery by selecting **Samples Gallery** on the First Steps console. WebSphere Process Server samples are initially listed as installable samples in the Samples Gallery. You can expand **Installable Samples** and look for samples under **Business Integration** that you want to deploy and run.

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_nameDeployed.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.

4. If the WebSphere Process Server installable samples were not installed automatically on the Samples Gallery, install and deploy them manually.
 - To install and deploy samples in a distributed WebSphere Process Server deployment 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 the following directory:
 - **Linux** **UNIX** **i5/OS** **On UNIX®, Linux® and i5/OS® platforms:** *install_root/samples/lib/SamplesGallery*
 - **Windows** **On Windows platforms:** *install_root\samples\lib\SamplesGallery*
 - 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 WBIExamplesGallery.ear file in the SamplesGallery directory.
 - 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 deploy and run each sample, but use **Install New Application** on the administrative console instead of the `installwbi` command, which does not support clusters. You can locate the deployed EAR files in the following directory for each sample:
 - **Linux** **UNIX** **i5/OS** **On UNIX, Linux and i5/OS platforms:** *install_root/samples/lib/sample_name*
 - **Windows** **On Windows platforms:** *install_root\samples\lib\sample_name*
 - To install and deploy samples in a distributed WebSphere Process Server deployment environment without clustering, perform the following steps.
 - a. On the workstation with the deployment manager node, run the following command:
 - **Linux** **UNIX** **i5/OS** **On UNIX, Linux and i5/OS platforms:**
install_root/samples/bin/installwbi -node node_name -server server_name -samples SamplesGallery WBIExamplesGallery
 - **Windows** **On Windows platforms:** *install_root\samples\bin\installwbi -node node_name -server server_name -samples SamplesGallery WBIExamplesGallery*

Note: If administrative security is enabled on the WebSphere Process Server profile, you must also type the `-samplepw` parameter and type the password that you created when creating the profile.

- b. In the administrative console, expand **Applications**, click **Enterprise Applications**, and start the SamplesGallery and WBIExamplesGallery.

- 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 deploy and run each sample, making sure to use the `-node node_name -server server_name` parameters with the `installwbi` command.

Related concepts

Options on the First steps console

After installing WebSphere Process Server, use the First steps console to start product tooling, access product documentation, or direct elements such as servers and administrative consoles related to individual profiles. A generic version of the console, plus a version for each profile in your installation are available. Options on each console are displayed dynamically, depending on features you install and the availability of certain elements on particular operating systems. Options include verifying your installation, starting or stopping the server or deployment manager, accessing the administrative console, starting the Profile Management Tool, accessing the Samples gallery, accessing the product documentation, or starting the migration wizard.

Business Process Management samples

Business Process Management samples demonstrate features developed in IBM WebSphere Integration Developer and deployed on IBM WebSphere Process Server. They help you work with various product features to develop your own applications.

Business Process Management samples are available at <http://publib.boulder.ibm.com/bpcsamp/index.html>.

Standards compliance

WebSphere Process Server 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, except in Business Space powered by WebSphere.
- 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 can install WebSphere Process Server either in graphical or silent form. The silent installation program is recommended for users with accessibility needs.

For instructions, see *Installing the product silently*.

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

This product includes certain third-party software not covered under the IBM license agreement. IBM makes no representation about status of these products regarding the Section 508 of the U.S. Federal Rehabilitation. Contact the vendor for information about the Section 508 status of its products. You can request a U.S. Section 508 Voluntary Product Accessibility Template (VPAT) on the IBM Product accessibility information Web page at www.ibm.com/able/product_accessibility.

Related accessibility information

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

Federal Information Processing Standards

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 about these standards, see the National Institute of Standards and Technology at <http://www.nist.gov/>.

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.

For more information, see "Configuring Federal Information Processing Standard Java Secure Socket Extension files" in the WebSphere Application Server information center. 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.

Related information

 [Configuring Federal Information Processing Standard Java Secure Socket Extension files](#)

Internet Protocol Version 6

WebSphere Process Server relies on WebSphere Application Server for all Internet Protocol Version 6 compatibility.

IBM WebSphere Application Server Version 6.1 and its JavaMail component support Internet Protocol Version 6 (IPv6).

For more information about this compatibility in WebSphere Application Server, see "IPv6 support" in the WebSphere Application Server Network Deployment documentation.

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

Related information

 [IPv6 support](#)

 www.ipv6.org

Globalization

WebSphere Process Server is globalized: It has multicultural support, and the user interface and documentation are translated into multiple languages.

Multicultural support means that WebSphere Process Server supports the cultural conventions of multiple languages and geographic regions. These conventions include the use of various writing systems and sort orders; various formats for date, time, numbers, and currency; and various keyboard layouts.

Translations are provided for the following national languages:

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

WebSphere Process Server provides partial translations (Human Task Manager and Business Process Choreographer Explorer) for the following national languages:

- Arabic
- Hebrew

Because WebSphere Process Server is built on WebSphere Application Server, you can get information about developing and assembling globalized applications in the WebSphere Application Server information center. See "Learn about WebSphere programming extensions" in the WebSphere Application Server Network Deployment documentation.

Bidirectional language support

WebSphere Process Server supports bidirectional languages, through bidirectional enablement. Bidirectional enablement is a mechanism for accurately displaying and processing bidirectional script data inside components either bundled with WebSphere Process Server (for example, Web-based tools such as the Common Base Event Browser or the business rules manager) or supported by it (for example, service components).

WebSphere Process Server processes all bidirectional language data in the logical, left-to-right format, which is the Windows standard bidirectional language format. It processes data passed to internal components, stores data, and outputs the data in that format. WebSphere Adapters and other Enterprise Information Systems (EIS), must convert the data into this format before sending the data to be processed by WebSphere Process Server. Because the data output by WebSphere Process Server is also in the logical, left-to-right format, the receiving application must convert it to the correct bidirectional format required by the external EIS.

The following table shows the attributes and settings that must match the Windows standard bidirectional format.

Table 4. 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

For data coming from an external component that does not enforce bidirectional support, such as Web services or connectors that are not enabled for processing bidirectional data, you can use example bidirectional application programming interfaces (APIs), based on the IBM Java Development Kit (JDK) to create APIs that transform the data from an external source into the supported bidirectional language format and that transform data sent from WebSphere Process Server to an external EIS into the bidirectional format used by that specific EIS.

To create APIs that transform string objects, see "Transforming string objects from one bidirectional language format to another."

To create APIs that transform data objects, see "Transforming data objects from one bidirectional language format to another."

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.

Related tasks

Transforming string objects from one bidirectional language format to another
For data coming from an external Enterprise Information System (EIS), you can create APIs that transform string data into the supported bidirectional language format and that transform data sent from WebSphere Process Server to an external EIS into the bidirectional format used by that specific EIS.

Transforming data objects from one bidirectional language format to another
For data coming from an external Enterprise Information System (EIS), you can create APIs that transform Service Data Objects into the supported bidirectional language format and that transform data sent from WebSphere Process Server to an external EIS into the bidirectional format used by that specific EIS.

Related information

 Learn about WebSphere programming extensions

 www.ibm.com/developerworks/websphere/library/techarticles/bidi/bidigen.html

Transforming string objects from one bidirectional language format to another

For data coming from an external Enterprise Information System (EIS), you can create APIs that transform string data into the supported bidirectional language format and that transform data sent from WebSphere Process Server to an external EIS into the bidirectional format used by that specific EIS.

Before you begin

For more information about bidirectional language support, see Globalization. Use the table in Globalization to determine the correct value for either the input string or output string to use when transforming string data from one format to another.

To create an API to transform the bidirectional language format of string objects, perform the following steps.

Procedure

1. Include all bidirectional classes that contain the bidirectional engine implementation. For example:

```
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. For example:

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

3. Call the `BidiStringTransformation` function. For example:

```
String strOut = BidiStringTransformation(strIn, formatIn, formatOut);  
String BidiStringTransformation(String strIn, String formatIn, String formatOut) {  
  a. Test if the input string is null. For example:  


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

  
  b. Perform the transformation. For example:
```

```

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;
}

```

Related concepts

Globalization

WebSphere Process Server is globalized: It has multicultural support, and the user interface and documentation are translated into multiple languages.

Transforming data objects from one bidirectional language format to another

For data coming from an external Enterprise Information System (EIS), you can create APIs that transform Service Data Objects into the supported bidirectional language format and that transform data sent from WebSphere Process Server to an external EIS into the bidirectional format used by that specific EIS.

Before you begin

For more information about bidirectional language support, see Globalization. Use the table in Globalization to determine the correct value for either the input string or output string to use when transforming DataObject-type data from one format to another.

To create an API to transform the bidirectional language format of data objects, perform the following steps.

Procedure

1. Include all bidirectional classes that contain the bidirectional engine implementation. For example:

```
import com.ibm.bidiTools.bdlayout.*;
```
2. Include all the classes you need to manipulate the DataObject-type object. For example:

```
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 filters the attributes of type String while recursively transversing the DataObject. For example:

```
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 NM_TOKEN_STR_TYPE = "NM_TOKEN";
String NC_NAME_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. For example:

```
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(NM_TOKEN_STR_TYPE)) ||
        (property.getType().getName().equalsIgnoreCase(NC_NAME_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 code logic includes the following assumptions:

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

For example:

```
DataObject BiDiDataObjTransformationB0(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. For example:

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

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

- b. Recursively call the transformation to handle child objects. For example:

```
for (int childNumber = 0; childNumber < childsList.size();
    childNumber++){
    BiDiDataObjTransformationB0(connectionContext,
    ((DataObject)childsList.get(childNumber)),formatIn, formatOut);
}
} else {
```

- c. Recursively call the transformation to handle child objects of any contained business objects. For example:

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

d. Transform the simple string attributes. For example:

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

Related concepts

Globalization

WebSphere Process Server is globalized: It has multicultural support, and the user interface and documentation are translated into multiple languages.

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 character set (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
1001 Hillsdale Blvd., Suite 400
Foster City, CA 94404
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 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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (^R or TM), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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

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

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

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

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 Multiplatforms, Version 6.2



Printed in USA