WebSphere® WebSphere Process Server for z/OS

IBM

Version 6.1.0

Product Overview

WebSphere WebSphere Process Server for z/OS

IBM

Version 6.1.0

Product Overview

31 March 2008

This edition applies to version 6, release 1, modification 0 of WebSphere Process Server for z/OS (product number 5655-N53) and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, 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.

Contents

Figures

Chapter 1. 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 an *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® from the WebSphere business process management zone at http://www.ibm.com/developerworks/websphere/zones/bpm/.

Chapter 2. What is new in this release

Version 6.1.0 enhancements focus on nine areas: platform alignment and currency, enhanced ease of use and business flexibility, product installation and configuration improvements, enhanced human workflow capabilities for your applications, enhanced business process capabilities for your applications, Business Process Choreographer Explorer and Business Process Choreographer Observer enhancements, enhanced business rules capabilities for your applications, improved connectivity with new and enhanced service component architecture (SCA) bindings and quality of service, and additional enterprise service bus mediation support.

Note: The information center has been updated for WebSphere Process Server for z/OS®, version 6.1.0.1. For more information about installing fix packs, see Getting fixes.

Welcome to WebSphere Process Server for z/OS, version 6.1.0, which includes the following new features:

- Enhanced ease of use and business flexibility:
  - 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. Those tasks are documented in the WebSphere Application Server documentation.
  - Install Factory enhancements for defining and building WebSphere Process Server installations at a specific fix level make it possible to add additional files to the installation and leave out features that are not required.
  - The Eclipse-based Profile Management Tool, consistent for all WebSphere platforms, enables WebSphere Process Server and WebSphere Enterprise Service Bus profile creation and augmentation within the same tool.
  - Configuration management for deployment is enhanced.
  - Support for pattern-based configurations improves productivity for administrators who configure WebSphere Application Server Network Deployment clusters and all required resources for WebSphere Process Server environments.
  - Necessary database tables are now created when first accessed, eliminating the need to create the database manually during configuration.
- Enhanced human workflow capabilities for your applications:
  - Participant substitution capabilities allow users of the runtime applications to temporarily delegate work while they are unavailable.
  - Bulk APIs allow transferring many tasks in a single operation.
  - Virtual Member Manager, WebSphere Application Server's new component to integrate client-specific people directories, is supported.
  - Auto deletion can be restricted to tasks that completed successfully.
  - Forms created using IBM Lotus® Forms Designer (integrated into WebSphere Integration Developer) can be used as the user interface for human tasks and processes.

– WebSphere Portal Server's My Task portlet can be extended with portlets generated from WebSphere Integration Developer.
- Enhanced business process capabilities for your applications:
  – Generic JMS interface for Business Flow Manager allows for programmatic interaction with business process templates and instances.
  – Extensions to the Generic Web Services Interface for Business Flow Manager add new runtime capabilities.
  – The Web Services Business Process Execution Language (WS-BPEL) **forEach** construct allows processing a dynamic number of multiple branches (either in parallel or serially).
  – Suspend capabilities are extended to allow specifying that process instances resume automatically.
  – Automatic deletion can be restricted to processes that completed successfully, allowing you to keep only the process instances that need further analysis or repair.
  – Additional data-handling option ignores missing data during access instead of producing errors.
  – Back links in single-threaded flows are supported.
- Enhanced business rules capabilities for your applications:
  – Custom business user clients can now administer business rules used in process flows with a new Business Rules Administration API that allows for creating, reading, updating, and deleting rules as an alternative to using the Business Rules Manager.
  – New custom properties can be assigned to Business Rule Groups and accessed from the rule logic of rule sets and decision tables to provide these rules with access to environment information captured in the properties. The properties can also be used for searching on Business Rule Groups through the Business Rules Manager or custom management clients.
- Business Process Choreographer Explorer and Business Process Choreographer Observer enhancements:
  – The Business Process Choreographer Explorer enhanced capabilities allow you to perform the following tasks:
    - Handle absence and substitution of users.
    - Use the "Suspend until" option for processes and tasks.
    - View and edit XML source data.
    - Use custom view improvements to sort and control the amount of data returned to the application users.
    - Combine filter criteria across processes and tasks with their definitions and instances.
    - Navigate between related tasks (subtasks and follow-on), administer and view information about specific tasks.
    - Include human task priority and business category as filter criteria and list columns.
    - Edit custom properties.
    - Improve usability of graphical process view.
  – Reports in the Business Process Choreographer Observer can now be exported for further analysis in tools such as Microsoft® Excel, and can be saved for automatic generation at a later time based on schedule or on demand.
- Improved connectivity:

- – Enhanced support for Web Services Description Language (WSDL) XML and Schema Definition (XSD) enables the use of many industry-standard XML schemas and increases the ability to connect with additional environments.
- – New and enhanced SCA bindings improve connectivity:
  - Enhancements enable integration with any JMS 1.1 Application Server Facilities (ASF) compliant messaging provider and enable automatic setup of generic JMS resources for WebSphere Application Server.
  - New generic HTTP 1.0 and 1.1 bindings enable connections with more applications and services.
  - New data bindings allow easy integration with WebSphere Transformation Extender.
  - Business fault support enhancements enable differentiation between business and runtime exceptions.
  - New data handler framework allows flexible data binding specification.
- – New and enhanced quality of service:
  - A business object instance validator can be invoked for either implicit, interface-qualifier-based validation or explicit, programmatic validation.
  - Enhanced event sequencing support improves quality of service support.
- Additional enterprise service bus mediation support:
  - – A new business object map primitive makes it easy to embed a map inside a mediation flow.
  - – A new **ServiceInvoke** primitive invokes a target service from within a request or response flow.
  - – New splitting and aggregating primitives allow enhanced processing of single or composite messages by splitting parts of the message and invoking multiple services.
  - – Enhanced custom mediation allows for new terminals to be defined.
  - – The message element setter primitive includes minor enhancements.
  - – The **MessageLogger** primitive includes system-wide common database support and a schema qualifier.
- Platform alignment and currency:
  - – Support for WebSphere Application Server version 6.1 allows WebSphere Process Server to use WebSphere Application Server capabilities and inherit the new features and benefits that were introduced in WebSphere Application Server version 6.1.
  - – Support for i5/OS® enables WebSphere Process Server to run on System i™ and to use native i5/OS facilities.
  - – Remote DB2® support for System i systems enables integration with i5/OS applications and enables use of relational database management servers running on i5/OS.
  - – Remote access from a distributed platform to a DB2 installation on a z/OS operating system is supported.
  - – 64-bit Windows® and UNIX® platforms are supported.
  - – Support for Windows Vista as a design, development, and test platform (not for production) enables use of Microsoft's latest version of their desktop operating system.
- Product installation and configuration improvements:
  - – Guided deployment environment installation allows you to easily define a working cluster or set of clusters across installations on multiple machines.

- Non-root installation allows you to install as a non-root user on Linux® or UNIX operating systems, or as a non-administrator on Windows.
- WebSphere Process Server provides a plugin to the Profile Management Tool -- an Eclipse-based tool for profile creation and augmentation that is provided as part of WebSphere Application Server, version 6.1. -- to allow for WebSphere Process Server and WebSphere Enterprise Service Bus profile creation and augmentation within the same tool.
- WebSphere Process Server provides a plugin to IBM Installation Factory, an eclipse-based tool that allows you to create WebSphere Process Server and WebSphere Enterprise Service Bus customized installation packages (CIPs) to include refresh packs, fix packs, and interim fixes, to exclude features, and to include additional files and scripts for both installation and profile creation.
- Trade Up Installation. Customers who purchase and install WebSphere Enterprise Service Bus, version 6.1, and later purchase WebSphere Process Server 6.1 will be able to trade up their WebSphere Enterprise Service Bus installation into a WebSphere Process Server installation by using the WebSphere Process Server 6.1 installer.
- The WebSphere Process Server installer supports remote Windows to i5/OS installation support. The Profile Management Tool can run on a Windows system remotely, connected to an i5/OS system.
- A new installer is available for the IBM User Interface Help System Built on Eclipse, enabling WebSphere Process Server information center documentation to be downloaded and installed from a central repository.

Related tasks

➡ Using Lotus Forms Server API and Lotus Forms Viewer

You can use forms created by IBM Lotus Forms Designer as the user interface for human tasks and processes.

Chapter 3. Product family overview

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

WebSphere Application Server

WebSphere Process Server is based on the robust J2EE infrastructure and associated platform services provided by WebSphere Application Server. WebSphere Process Server for z/OS is built on WebSphere Application Server for z/OS. For more information about WebSphere Application Server Network Deployment, see the WebSphere Application Server Network Deployment 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, Version 6.1 Information Center.

IBM WebSphere Enterprise Service Bus

WebSphere Process Server is powered by the same technology available with WebSphere Enterprise Service Bus. Enterprise service bus capability is part of the underlying functions 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, 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 Partner Gateway

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

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 IBM WebSphere Business Process Management information center.

IBM WebSphere Adapters

IBM 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.

Websphere Adapters are included components with WebSphere Integration Developer.

For more information about WebSphere Adapters, see the WebSphere Integration Developer documentation in the IBM WebSphere Business Process Management information center.

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 WebSphere Business Modeler information center and the WebSphere Business Monitor documentation at IBM WebSphere Business Process Management information center.

IBM Rational® Application Developer and IBM Rational Software Architect

WebSphere Integration Developer can be used in conjunction with IBM Rational Application Developer, or IBM 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

You can extend existing applications for reuse in enterprise processes with an IBM enterprise modernization portfolio that includes CICS Transaction Gateway and WebSphere Host Access Transformation Services.

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

IBM WebSphere Portal

IBM WebSphere Portal provides access to various administrative functions and allows portlets to have access to business processes and other Service Component Architecture services in WebSphere Process Server.

For more information about WebSphere Portal, see the WebSphere Portal product documentation library.

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.

Chapter 4. Architectural overview of WebSphere Process Server

WebSphere Process Server delivers an integration platform with a fully converged, standards-based business process engine, using the full power of WebSphere Application 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.

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

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

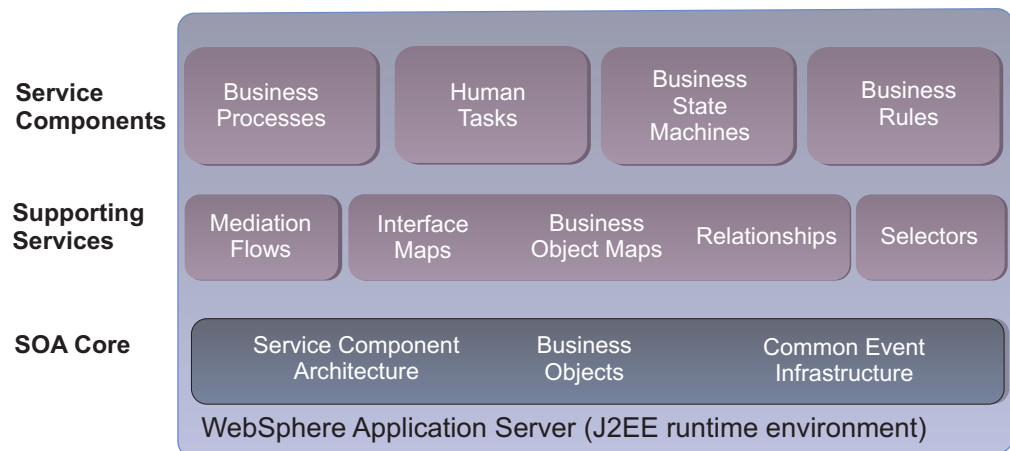*One component-based framework addresses all styles of integration.*

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.

11

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.

One component-based framework addresses all styles of integration.
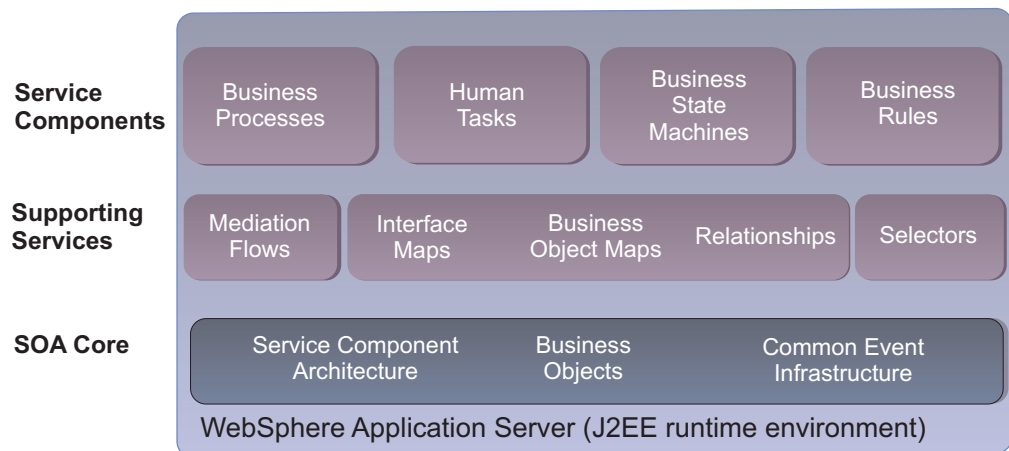
Figure 2. WebSphere Process Server component-based framework

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.

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

Service components can be assembled graphically in the IBM WebSphere Integration Developer tools, and the implementation can be added later. The Service Component Architecture programming model narrows what developers must know about Java™ and J2EE or other implementation in particular scenarios

to a core set of language concepts that are familiar to all who develop business applications in other programming languages today. This allows developers to quickly and easily integrate technologies.

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

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

Service qualifiers govern the interaction between a service client and a service on the WebSphere Process Server runtime environment. Service qualifiers are quality of service specifications that define a set of communication characteristics required by an application for transmission priority, level of route reliability, transaction management, and security level. An application communicates its quality of service needs to a runtime environment by specifying service qualifiers. Quality of service qualifiers can be specified when wiring components in the assembly editor in WebSphere Integration Developer. These specifications, when running on WebSphere Process Server, determine how the clients interact with the target components. Depending on the qualifiers specified, 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 a same application, or to other applications on enterprise information systems (EIS). This allows working with IBM WebSphere Adapters. For more information, see Imports, exports, and adapters.

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

For business processes, WebSphere Process Server offers support for transactions involving multiple resource managers using the two-phase commit process to ensure atomic, consistent, isolated, and durable (ACID) properties. This capability

is available for both short-running flows (single transaction) and long-running flows (multiple transactions). You can group multiple steps in a business process into one transaction by modifying transaction boundaries in WebSphere Integration Developer.

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

Service Data Objects and business objects

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

The *Service Data Object (SDO)* technology is an open standard for enabling applications to handle data from heterogeneous data sources in a uniform way. SDO incorporates J2EE patterns but simplifies the J2EE data programming model.

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 documentation

Related concepts

"Business object maps" on page 17
Business object maps are a way of relating business objects.

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 concepts

Common Event Infrastructure

⮕ Common Base Event model

⮕ 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 IBM WebSphere Process Server.

One component-based framework addresses all styles of integration.
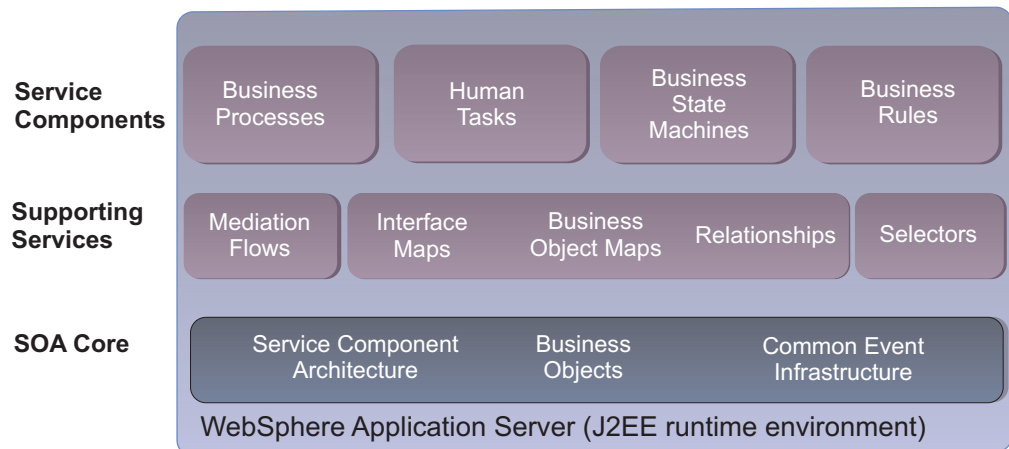
Figure 3. WebSphere Process Server component-based framework

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 as part of a mediation module in WebSphere Process Server.

Related concepts

Chapter 7, "Enterprise service bus messaging infrastructure," on page 27
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" on page 14
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.

Service components

All integration artifacts running on IBM WebSphere Process Server (for example, business processes, business rules, and human tasks) are represented as components with well defined interfaces.

Within the Service Component Architecture (SCA), a *service component*, 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.

One component-based framework addresses all styles of integration.

Service Components	Business Processes	Human Tasks	Business State Machines	Business Rules
Supporting Services	Mediation Flows	Interface Maps	Business Object Maps	Relationships	Selectors
SOA Core	Service Component Architecture	Business Objects	Common Event Infrastructure
	WebSphere Application Server (J2EE runtime environment)		
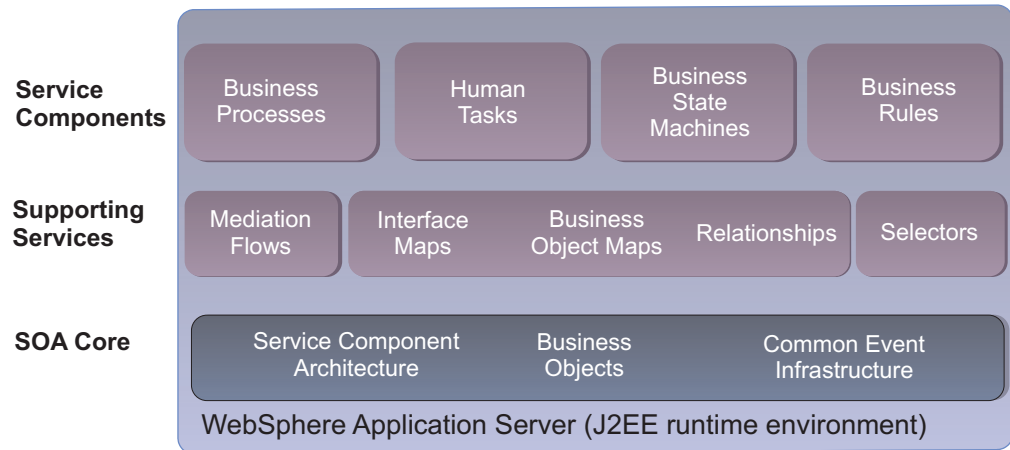
Figure 4. WebSphere Process Server component-based framework

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 actually a series of individual tasks, and each task is executed in a specific order. As an integral part of applications running on IBM WebSphere Process Server, business processes provide the primary means through which enterprise services are integrated.

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

➦ 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 concepts

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 concepts

Business state machine events

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.

Chapter 5. Imports, exports and adapters

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

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

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

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

WebSphere Adapters include the following:
- IBM WebSphere Adapter for E-mail, version 6.1
- IBM WebSphere Adapter for FTP, version 6.1
- IBM WebSphere Adapter for Flat Files, version 6.1
- IBM WebSphere Adapter for JDBC, version 6.1
- IBM WebSphere Adapter for SAP Software, version 6.1
- IBM WebSphere Adapter for Siebel Business Applications, version 6.1
- IBM WebSphere Adapter for Oracle E-Business Suite, version 6.1
- IBM WebSphere Adapter for JD Edwards EnterpriseOne, version 6.1

Websphere Adapters are included components with WebSphere Integration Developer.

For more information about WebSphere Adapters, see the WebSphere Integration Developer documentation in the IBM WebSphere Business Process Management information center.

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

Chapter 6. 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 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 WebSphere Process Server SCA modules including Business Process Choreographer, business rules, mediations, and relationships. This topology consists of one server process running on one computer system, or it may 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 will most likely set up 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.

25

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

Chapter 7. Enterprise service bus messaging infrastructure

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

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

WebSphere Process Server is powered by the same technology available with IBM WebSphere Enterprise Service Bus. This capability is part of the underlying functionality of WebSphere Process Server, and no additional license for WebSphere Enterprise Service Bus is required to take advantage of these capabilities.

However, you can deploy additional stand-alone licenses of WebSphere Enterprise Service Bus around your enterprise to extend the connectivity reach of the process integration solutions powered by WebSphere Process Server. For example, WebSphere Enterprise Service Bus can be installed closer to an SAP application to host an IBM WebSphere Adapter for SAP and to transform SAP messages before sending that information across the network to a business process choreographed by WebSphere Process Server.

Related concepts

"Mediation flows" on page 16
Mediation flows intercept and modify messages that are passed between existing services (providers) and clients (requesters) that want to use those services.

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.

"Mediation modules" on page 28
Mediation modules are Service Component Architecture (SCA) modules that can change the format, content, or target of service requests.

"Mediation primitives" on page 30
Mediation components operate on message flows between service components. The capabilities of a mediation component are implemented by *mediation primitives*, which implement standard service implementation types.

"Service message objects" on page 34
Service message objects (SMOs) provide an abstraction layer for processing and manipulating messages exchanged between services.

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, version 6, including Web services Client, EJB Client, and JMS Client.

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.

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 dynamically, 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 module can contain only one mediation flow component.

  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.
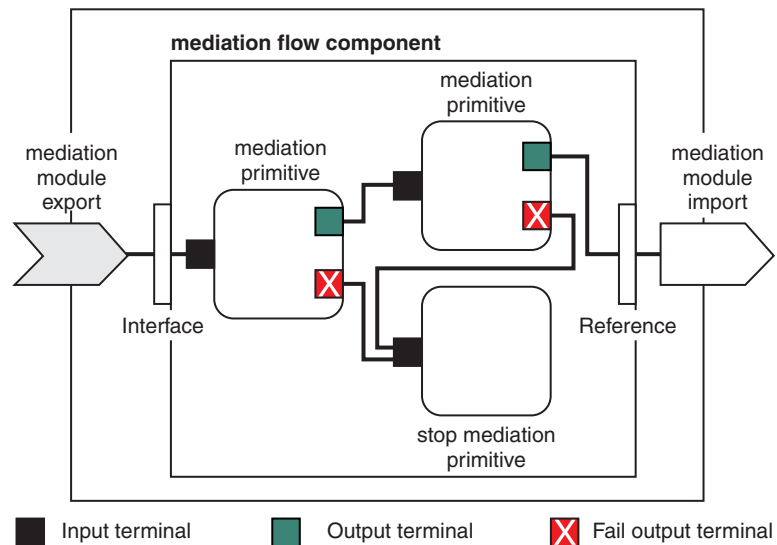
Figure 5. Simplified example of a mediation module. The mediation module contains one mediation flow component, which contains mediation primitives.

- 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 flag the properties as promoted. Certain properties lend themselves to being administratively configured and WebSphere Integration Developer describes these as promotable properties, because they can be promoted from the integration cycle to the administrative cycle. Other properties are not suitable for administrative configuration, 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.

  Newly called mediation flows use property changes immediately, unless the changes occur in a deployment manager cell. If changes occur in a deployment manager cell they take effect on each node as that node is synchronized. Mediation flows that are in-flight continue to use previous values.

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

Deploying mediation modules

Mediation modules are created using WebSphere Integration Developer, and are generally deployed to WebSphere Process Server inside an 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
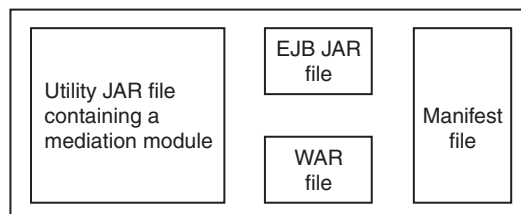
Figure 6. Simplified example of an EAR file containing a mediation module. The EAR file contains JARs. The utility JAR file contains a mediation module.

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.

Mediation primitives

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

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

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

A mediation primitive defines 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.
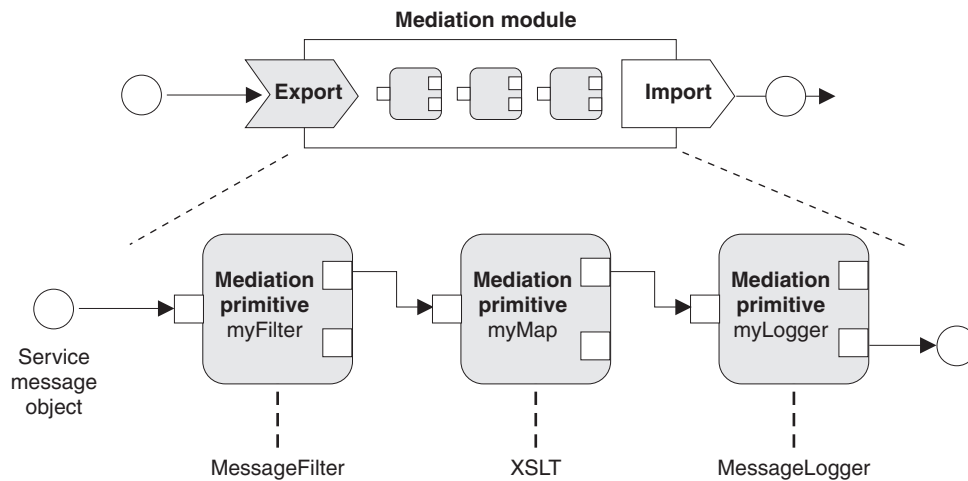
Figure 7. Mediation module containing three mediation primitives

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.

WebSphere Integration Developer also allows you to graphically model and assemble mediation components from mediation primitives, and assemble mediation modules from mediation components.

Supported mediation primitives

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

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.

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 Java primitive or a Java String (or be able to be cast to a Java primitive or a Java String).

Endpoint Lookup

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

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

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.

Message Filter

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

Message Logger

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

- The 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.

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.

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.

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.

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 message payload. The message payload is the application data exchanged
  between service endpoints.
- 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 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 Service Definition Language (WSDL) message that you
specify when you configure a mediation flow.

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.
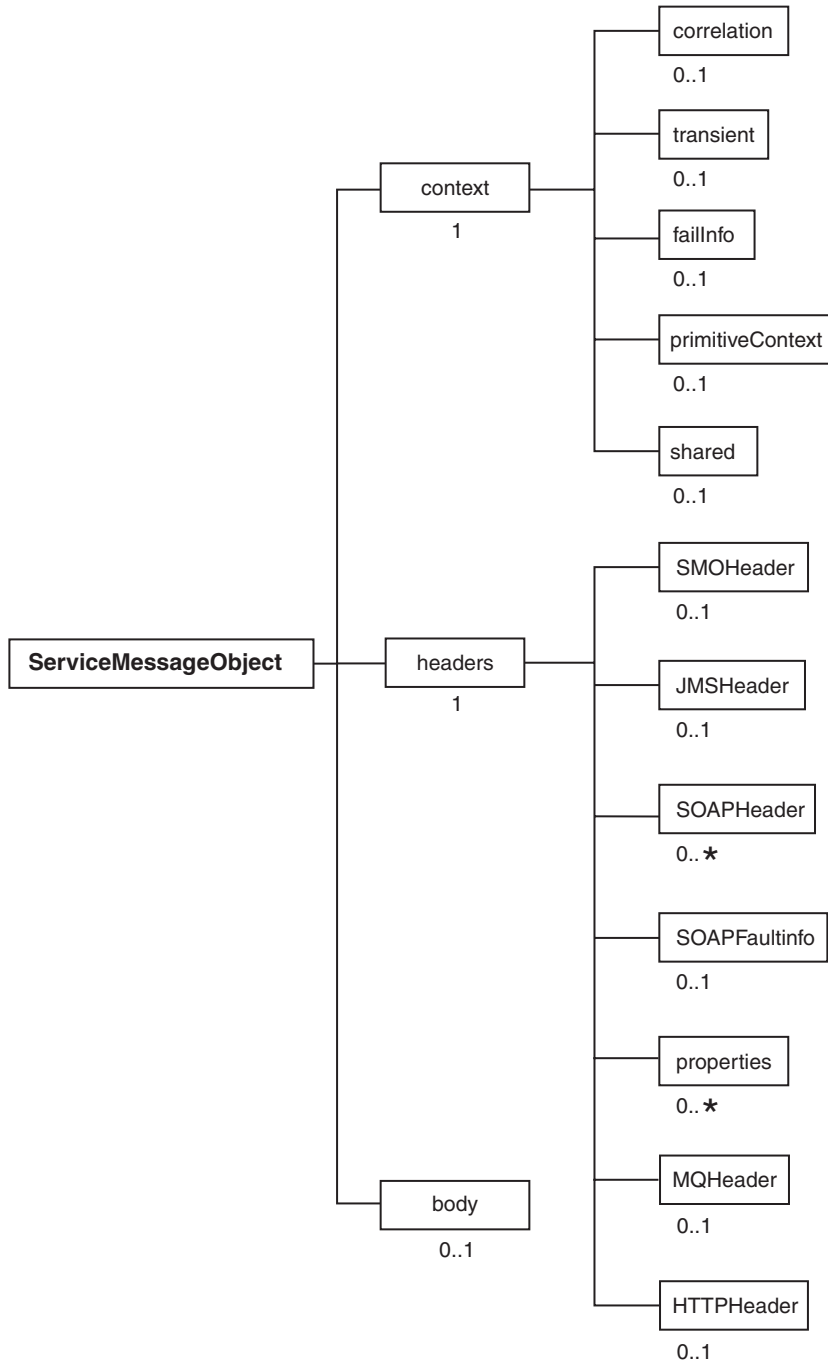
```
                                                            ┌──────────────────┐
                                                            │ correlation      │
                                                            └──────────────────┘
                                                              0..1
                                                            ┌──────────────────┐
                                                            │ transient        │
                                              ┌──────────┐  └──────────────────┘
                                              │ context  │    0..1
                                              └──────────┘  ┌──────────────────┐
                                                 1          │ failInfo         │
                                                            └──────────────────┘
                                                              0..1
                                                            ┌──────────────────┐
                                                            │ primitiveContext │
                                                            └──────────────────┘
                                                              0..1
                                                            ┌──────────────────┐
                                                            │ shared           │
                                                            └──────────────────┘
                                                              0..1

                                                            ┌──────────────────┐
                                                            │ SMOHeader        │
                                                            └──────────────────┘
                                                              0..1
                                                            ┌──────────────────┐
                                                            │ JMSHeader        │
                                                            └──────────────────┘
                                                              0..1
                                                            ┌──────────────────┐
┌─────────────────────┐    ┌──────────┐                     │ SOAPHeader       │
│ ServiceMessageObject │───│ headers  │                     └──────────────────┘
└─────────────────────┘    └──────────┘                       0..*
                              1                             ┌──────────────────┐
                                                            │ SOAPFaultinfo    │
                                                            └──────────────────┘
                                                              0..1
                                                            ┌──────────────────┐
                                                            │ properties       │
                                                            └──────────────────┘
                                                              0..*
                                                            ┌──────────────────┐
                                           ┌──────────┐     │ MQHeader         │
                                           │ body     │     └──────────────────┘
                                           └──────────┘       0..1
                                             0..1           ┌──────────────────┐
                                                            │ HTTPHeader       │
                                                            └──────────────────┘
                                                              0..1
```

Figure 8. Overview of SMO structure. The context, headers and body of a ServiceMessageObject

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

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.

Chapter 8. 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 process and human tasks. You can view information about process templates, process

37

instances, task instances, and their associated objects. You can also act on these objects; for example, you can start new process instances, repair and restart failed activities, manage work items, and delete completed process instances and task instances.

- Business Process Choreographer Observer

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

- 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 for WebSphere Process Server

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

  ↪ Administering WebSphere Process Server

  The topics in this section describe how to administer the WebSphere Process Server runtime environment, including the applications and resources deployed in the environment.

Chapter 9. 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 using WebSphere Process Server.

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

In addition to the WebSphere Integration Developer development environment, Service Component Architecture APIs are published for developers. For more information about Service Component Architecture APIs, see Developing and deploying modules.

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

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

Deploying is the act of enabling your applications – your SCA modules – in either a test or a production environment. Deploying is the same for both environments, there are a few differences between the deployment task in each environment. Because it is best to test any changes to your 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 before using WebSphere Process Server to deploy the modules into a production environment.

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 Installing a module on a production server and Deploying applications using Apache Ant tasks.

Related concepts

Developing and deploying modules
Developing and deploying modules are fundamental tasks.

Installing a module on a production server
This topic describes the steps involved in taking an application from a test server and deploying it into a production environment.

Deploying applications using Apache Ant

This topic describes how to use Apache™ Ant tasks to automate the deployment of applications to WebSphere Process Server. By using Apache Ant tasks, you can define the deployment of multiple applications and have them run unattended on a server.

Chapter 10. Security on WebSphere Process Server

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

For more information about security, see Securing applications and their environment.

Chapter 11. Monitoring on WebSphere Process Server

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

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 concepts

Monitoring

Monitoring the performance and business processes of your service component events is an important feature of IBM WebSphere Process Server.

Chapter 12. Samples and tutorials

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

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

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

Tutorials

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

IBM Education Assistant tutorials

The IBM Education Assistant site provides tutorials that you can use at your convenience. To view this educational content, see the IBM Education Assistant at http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0.

WebSphere Process Server tutorials

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

Tutorial: Relationship manager administration.

Tutorial: Writing a Jacl script that lists security roles.

Tutorial: Using the CEI server for event monitoring.

Tutorial: Logging service component events.

Tutorial: Service component performance monitoring.

Accessing the Samples (Samples Gallery)

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

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

   The samples are installed in the *install_root*/samples directory.
2. Start the server.
3. The Samples Gallery can be accessed directly from a web browser with the link http://xxxxxxxx:9080/WSsamples/en/index.html, where xxxxxxxx represents a host name or IP address of the target z/OS system. The actions of deploying and installing each sample into the server is part of the samples themselves. WebSphere Process Server samples are initially listed as installable samples in the Samples Gallery.
4. Start the Samples Gallery and click **Refresh**. You can deploy any of the installable samples by following the instructions that appear in the browser window. The samples that you previously deployed are listed as installed samples that you can run by selecting an option in the browser window.
5. Run each of the deployed samples.

Chapter 13. 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.

Accessibility features for WebSphere Process Server

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

Accessibility features

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

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

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

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

Keyboard navigation

This product uses standard Web browser navigation keys.

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

Interface information

- Installation

  You install WebSphere Process Server for z/OS using scripts, and you can install the WebSphere Process Server Client silently from the command line. This silent installation meets accessibility needs. For instructions see Installing the server and Installing the WebSphere Process Server Client silently.

- 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 on 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.

Common Criteria

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

WebSphere Process Server derives its security capabilities from WebSphere Application Server, and does not include additional security capabilities. For more information about Common Criteria compliance in WebSphere Application Server, see Common Criteria (EAL4) support in the WebSphere Application Server information center.

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.

Chapter 14. Globalization

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

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

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

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

- Arabic
- Hebrew

For information about the globalizing applications and the internationalization service available through WebSphere Application Server, see WebSphere 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 to be the logical, left-to-right, 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 1. Bidirectional language format string values

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

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 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.

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;
   }
   ```

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

4. Define the function that verifies if the type of a property is String. 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(NMTOKEN_STR_TYPE)) ||
       (property.getType().getName().equalsIgnoreCase(NCNANE_STR_TYPE)) ||
       (property.getType().getName().equalsIgnoreCase(ID_STR_TYPE)) ||
       (property.getType().getName().equalsIgnoreCase(IDREF_STR_TYPE)) ||
       (property.getType().getName().equalsIgnoreCase(IDREFS_STR_TYPE)) ||
       (property.getType().getName().equalsIgnoreCase(ENTITY_STR_TYPE)) ||
   ```

```
      (property.getType().getName().equalsIgnoreCase(ENTITIES_STR_TYPE)))
        rc = true;
     return rc;
}
```

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

   Note: The 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 BiDiDataObjTransformationBO(DataObject boIn, String formatIn, String formatOut){

    Type type;
    Property property;

    if (boIn == null) return null;

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

   a. Skip all non-string properties. 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++){                  BiDiDataObjTransformationBO(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;
}
```

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

This information 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, CICS, DB2, developerWorks, i5/OS, Lotus, Rational, Tivoli,
ViaVoice, WebSphere, z/OS, and zSeries are registered trademarks and System i is
a trademark of International Business Machines Corporation in the United States,
other countries, or both.

Adobe is a registered trademark of Adobe Systems Incorporated in the United
States, and/or other countries.

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

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

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

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

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

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

IBM WebSphere Process Server for z/OS, Version 6.1.0

IBM ®

Printed in USA