WebSphere® Process Server

Version 6.0

Product overview

Note
Before using this information, be sure to read the general information in "Notices" on page 27.

September 29 2005

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

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

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

Contents

Chapter 1. Product overview

You can use links to overview and introduction materials to gain a high-level understanding of IBM WebSphere Process Server.

WebSphere Process Server documentation (PDF files)

Hardware and software requirements for WebSphere(R) Process Server are available on the WebSphere Process Server system requirements Web site.

Information roadmaps for WebSphere Process Server are available on the IBM(R) developerWorks Web site under WebSphere Business Integration information roadmaps.

Overview materials include an introduction to the product, new features, information about other IBM products that work with IBM WebSphere Process Server, a technical introduction to the architecture and components, and information about samples, standards compliance, and internationalization.

Introduction to WebSphere Process Server

IBM WebSphere Process Server is the next generation business process integration server that has evolved from proven business integration concepts, application server technologies, and the latest open standards. It provides a simplified programming model, an invocation model, and a deployment model that are based on open standards.

On demand businesses, those whose business processes are integrated end-to-end across the company and with key partners, suppliers and customers, are gaining sustainable advantage over their competitors. Your IT infrastructure must be capable of instantiating new business processes and adapting quickly to any future changes in business processes.

IBM(R) WebSphere(R) Process Server addresses the common business need for a simplified and flexible IT infrastructure by including the following capabilities.

- Model-driven business process management, which helps companies build continuous and sustained improvement in predictability and performance.
- Standards-based componentization, which enables composite application development, improving reuse and time to market.
- Service-oriented architecture, which provides standards-based architecture to support both of the above.

WebSphere Process Server fully supports model-driven business process development and seamlessly integrates with WebSphere portfolio including IBM WebSphere Business Modeler and IBM WebSphere Business Monitor to provide an end-to-end business process for continuous and sustained business process improvement. In this way, WebSphere Process Server enables an on demand operational environment with improved business flexibility and increased business responsiveness.

Componentization is a transformation from a monolithic, product-oriented development model to one based on software offerings composed from the sharing

of reusable software components, a key part of the IBM On Demand strategy. WebSphere Process Server runs composite applications that are authored in WebSphere Integration Developer, a tool that makes it easy to decompose a business model, map process flows to service components, and assemble solutions by wiring service components. These capabilities are easy to use and do not require J2EE or programming skills.

WebSphere Process Server enables deployment of standards-based integration applications in a service-oriented architecture (SOA). 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. Loosely coupled integration applications that are based on SOA provide flexibility and agility. You can implement integration solutions independent of platform, protocols and products. For more information about SOA, refer to the Service-Oriented Architecture (SOA) from IBM Web site.

Hardware and software requirements

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

Information Roadmaps

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

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

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

What is new in this release

IBM WebSphere Process Server, version 6.0, includes several new features.

Welcome to IBM(R) WebSphere(R) Process Server, version 6.0, which includes the following new features:

- Service Component Architecture - one simplified integration framework that leverages existing IT.
- A deployment environment for applications developed in IBM WebSphere Integration Developer, version 6.0, a new simplified development tool with visual editors for component development, assembly, integrated testing, and deployment.
- Support for all styles of integration - including human tasks, role-based task assignments, and multilevel escalation.

- Ability to change business processes with minimal programming skills, without redeploying the application.
- Business rules, business state machines, and selectors to dynamically choose interfaces based on business scenarios.
- Broad reach in integration - built on Enterprise Service Bus (ESB) technologies and support for IBM WebSphere Adapters.
- Support for business-to-business (B2B) applications through a limited license of IBM WebSphere Partner Gateway included with WebSphere Process Server.

Product family overview

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

IBM(R) WebSphere(R) Process Server is based on the robust J2EE 1.4 infrastructure and associated platform services provided by WebSphere Application Server, version 6.0. WebSphere Process Server is built on WebSphere Application Server, Network Deployment, version 6.0. WebSphere Process Server also works with infrastructure and platform services from WebSphere Application Server, version 6.0. For more information about WebSphere Application Server, see the WebSphere Application Server Network Deployment information center.

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

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

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

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

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

IBM WebSphere Partner Gateway used with WebSphere Process Server supports business-to-business (B2B) 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 information center.

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 information center.

Technical overview of WebSphere Process Server

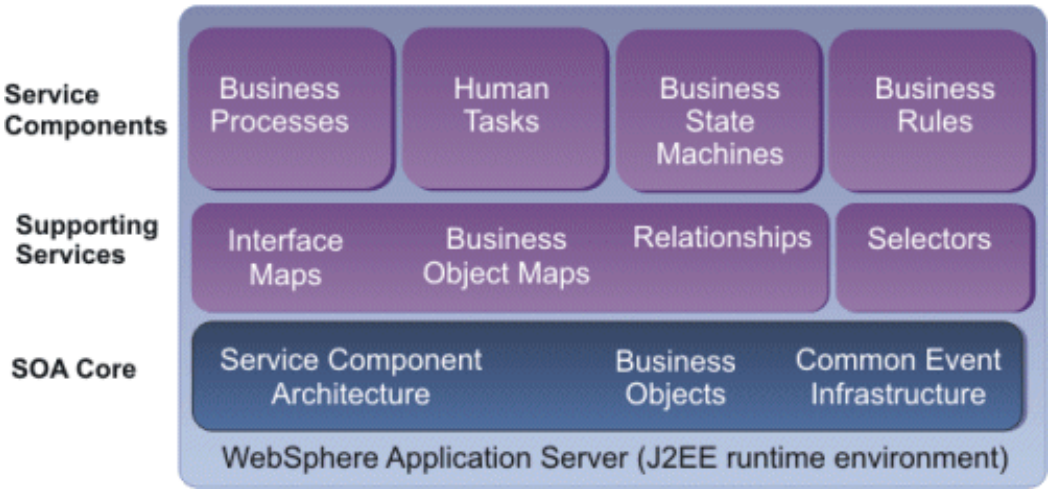
IBM WebSphere Process Server combines integration capabilities with a composite application platform to deliver a integration platform with a fully converged, standards-based business process engine, using the full power of WebSphere Application Server.

IBM(R) WebSphere(R) 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 WebSphere Process Server. Supporting services provide the foundational business object and transformation framework for WebSphere Process Server. Service components represent the functional components required to build 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.

One component-based framework addresses all styles of integration.



Everything in WebSphere Process Server is a component. These components have an interface and can be wired together to form a module. This enables changing any part of an application without affecting the other parts. For example, a human task can be replaced with a business rule without the need to touch the business process.

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

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

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

WebSphere Process Server includes the following features:

- 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.
- Business process integration semantics include compensation, and recovery.
- Human task services allow role-based task assignment, invocation and escalation.
- Business rules make business processes more flexible.
- Business process transformation services include mapping, mediation and assembly.
- Business state machines enrich business transactions.
- A simplified programming model eliminates programming complexities.
- Support for Common Event Infrastructure and Common Base Events enhances tracking, auditing and monitoring of business processes.
- A common administration console simplifies administration.

- Applications developed with easy-to-use tools in WebSphere Integration Developer can be deployed to a runtime environment.
- Business models can be transferred from WebSphere Business Integration Modeler to WebSphere Integration Developer.

Process integration

Process integration means integrating applications, data, and processes within an enterprise or among a set of enterprises. You can use WebSphere Integration Developer and WebSphere Process Server capabilities to integrate applications, data and processes.

Business units and enterprises that are collaborating have a need to closely integrate applications that at one time worked well separately.

Common challenges to integrating applications, data, and process include the following:

- Applications "re-invent the wheel" instead of reusing work others have built.
- When applications need a service from either within their own application or another integrated application, programmers build the interfaces between the service requester and service provider in a tightly coupled way to manage their dependencies and maintenance risk if the service changes.
- Programmers spend time creating and maintaining infrastructure code within one or more of your business applications, taking away precious time and budget which would otherwise be spent implementing business logic.

With WebSphere Integration Developer and WebSphere Process Server, you can develop, deploy and maintain integrated applications to address these business challenges. Everything in WebSphere Process Server is a component. These components have an interface and can be wired together to form a module (with an export). This enables changing any part of an application without affecting the other parts.

Process integration simplifies and enhances the value of service-oriented architecture (SOA). SOA is an application framework that takes everyday business applications and breaks them down into individual business functions and processes, called services. SOA lets you build, deploy, integration and manage these services independently of the applications and the computing platforms on whey they are running. Process integration simplifies integration, revolutionizes business flexibility and provides broad-reaching integration in the following ways:

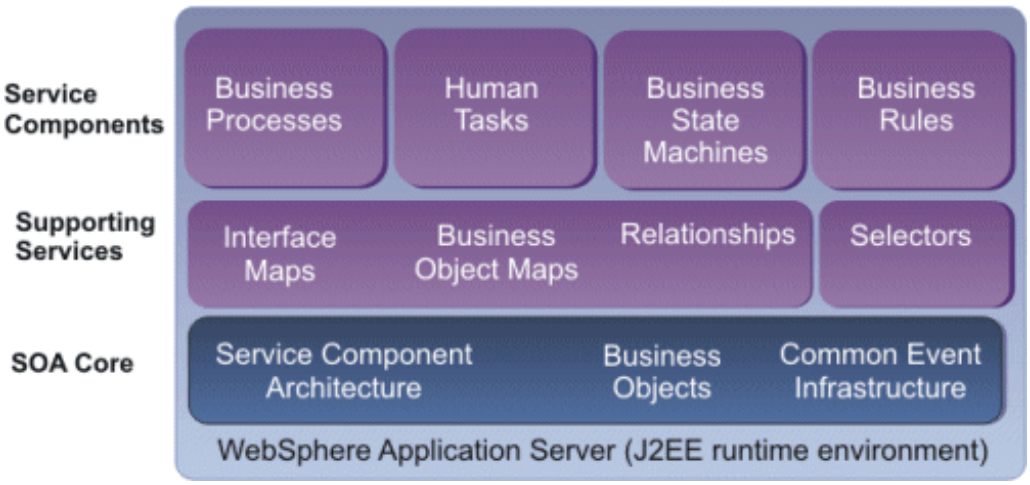
- WebSphere Process Server provides one runtime environment with simple deployment and one high-performance process engine for all process integration solutions.
- WebSphere Process Server provides business flexibility by enabling you to change processes while a solution is deployed to the runtime environment. Business rules provide agility and responsiveness, and programming knowledge is not required.
- WebSphere Process Server provides integration that is designed for wide range, powered by IBM Enterprise Service Bus (ESB) technologies and supporting IBM WebSphere Adapters, version 6.0. and IBM WebSphere Business Integration Adapters (based on WebSphere Business Integration Framework, version 2.6).

Service Component Architecture

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

Service Component Architecture separates business logic from implementation, so that you can focus on assembling an integrated application without knowing implementation details. The implementation of business processes is contained in service components. The result is an architecture of three layers, as shown in the following diagram.

One component-based framework addresses all styles of integration.



Service components can be assembled graphically in the 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 components

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

Because all integration artifacts are represented as SCA components, WebSphere Process Server creates an environment with unparalleled flexibility. It is possible, for example, to replace a human task for an approval with a business rule for automatic approval simply by replacing the components in the assembly diagram without changing either a business process or the caller of the business process.

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

- *Business processes*. The business process component implements a fully supported Web Services Business Process Execution Language (BPEL) engine. It represents the fourth release of a business process choreography engine on top of the WebSphere Application Server. You can develop and deploy complex business processes in a simple development model with sophisticated support for long and short running business processes in a highly scalable infrastructure. You can either create BPEL models in WebSphere Integration Developer, version 6.0, or import them from a business model you created in WebSphere Business Modeler, version 6.0.
- *Human tasks*. In WebSphere Process Server, human tasks are stand-alone service components that can be used to either assign work to employees or to invoke other services. The Human Task Manager tool supports ad-hoc creation and tracking of tasks. Existing LDAP directories (as well as operating system repositories and the WebSphere user registry) can be used to access user and group information. WebSphere Process Server supports multi-level escalation for human tasks including e-mail notification. It also includes a Web client to manage human tasks, and a set of Java Server Faces (JSF) components that can be used to create custom clients or to embed human task functionality into other Web applications.
- *Business state machines*. A business state machine, which specifies the sequences of states, responses, and actions that an object or an interaction goes through during its life in response to events, provides another way of modeling a business process. This gives you the choice of representing business processes based on states and events rather then a sequential business process model.
- *Business rules*. Business rules are declarations of policy or conditions that must be satisfied within your business that can be captured in models. 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 WebSphere Integration Developer. WebSphere Process Server also includes Web-based runtime tools for business analysts so that business rules can be updated as business needs dictate, without affecting other SCA services.

Supporting services in WebSphere Process Server, which include maps, relationships, interface mediation components, and selector components, address a number of transformation challenges developers face when connecting components and external artifacts.

Service Data Objects and business objects

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

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

WebSphere Process Server includes business objects, which are enhanced SDOs. Business objects are extensions of SDOs that provide an abstraction layer for data access. SDOs provide a universal means of describing disparate data (like JDBC ResultSet, XML Schema described data, for example). Business objects include

some extensions that are important for integration solutions and are used to further describe the data that is being exchanged between Service Component Architecture services.

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

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

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

- Business object definition
- Business graph definition
- Business object metadata definition
- Business object services

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

For more information about WebSphere Application Server Service Data Objects, see the WebSphere Application Server information center.

Service qualifiers

Service qualifiers govern the interaction between a service client and a service on the WebSphere Process Server runtime environment.

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

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 modules

Modules work with Service Component Architecture to group components together for deployment on WebSphere Process Server and provide further specification and encapsulation of services.

A module determines what artifacts are packaged in enterprise archive (EAR) file that is deployed to WebSphere Process Server.

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

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

Service Component Architecture enables services to be invoked through both synchronous and asynchronous programming styles. Assemble modules into solutions with asynchronous calls to services when you desire an increase the overall throughput of the system (for example, when using long-running services). Within the asynchronous style, there are mechanisms to request without a response, to call back the requester when the service completes, and to request and wait for a response.

For more information about service components, see Service components.

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.

A *module level* import or export lets modules access other modules. A *system level* import or export lets your application access applications on EIS systems as if they were local components, which allows working with WebSphere Adapters and WebSphere Business Integration Adapters.

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

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

WebSphere Adapters include the following:

- IBM WebSphere Adapter For Flat Files, version 6.0
- IBM WebSphere Adapter for JDBC, version 6.0
- IBM WebSphere Adapter for PeopleSoft Enterprise, version 6.0
- IBM WebSphere Adapter for Siebel Business Applications, version 6.0
- IBM WebSphere Adapter for SAP Applications, version 6.0

For more information about these products, see the WebSphere Adapters information center.

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

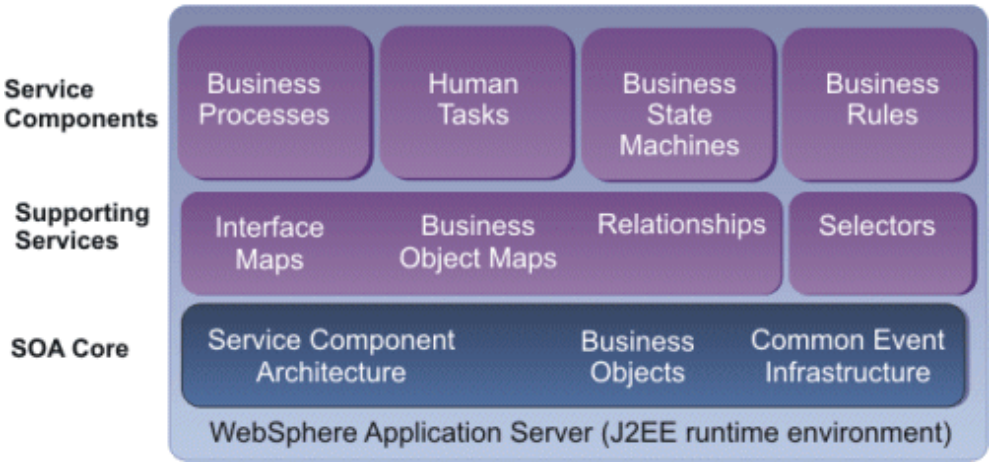
Imports and exports require binding information, which specifies the means of transporting the data from the modules. The assembly editor in WebSphere Integration Developer sets up imports and exports, lists the bindings supported and simplifies the creation of them. A properties view displays the binding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

Selectors

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

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

One component-based framework addresses all styles of integration.



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

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

For more information about service components, see “Service components” on page 7.

Service implementation types

Integration applications running on WebSphere Process Server can support several standard service implementations, including business processes, Enterprise Information Systems (EIS) systems, and human tasks.

Service Component Architecture supports several standard service implementation types:

- *Java objects*. A Java component implements a Java class. As in the Java programming language, instances of Java components at run time are referred to as Java objects.
- *Business processes*. A process component implements a business process. Its implementation language is the Business Process Execution Language (BPEL) and its IBM extensions.
- *Human tasks*. A human task component represents and implements a task typically performed by a person in a business process or an integration application.
- *Business state machines*. Applications sometimes work with artifacts that have a set of states. A state machine defines what the artifact can do at a point in time.
- *Mediation*. Mediation components facilitate the interaction between service components. They are implemented in a variety of ways, including with Java classes or with short BPEL processes.

- *Business rules*. Business rules 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.

Development and deployment of integrated applications

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

WebSphere Integration Developer is the development environment for WebSphere Process server. For more information about developing integrated applications in WebSphere Integration Developer, refer to the WebSphere Integration Developer 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, refer to the WebSphere Process Server *Developing and Deploying Modules* PDF file.

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

If you need to deploy many application files, you may want to use a batch file.

For more information about batch files, refer to the WebSphere Process Server *Developing and Deploying Modules* PDF file.

Administration of applications on WebSphere Process Server

Administering applications on WebSphere Process Server is made easier through a variety of tools that it provides.

For more information about administering applications, refer to the WebSphere Process Server *Administration Guide*.

- Administrative console

  The administrative console is a graphical interface that guides you through deployment and systems administration tasks. From the administrative console, you can monitor, update, stop, and start a wide variety of applications, services, and resources. The administrative console can also be used to work with relationships and to locate and resolve failed WebSphere Process Server events. There are specific panels for administering WebSphere Process Server. Other panels provide administration options for WebSphere Application Server, the basis for WebSphere Process Server.
- Scripting program

The WebSphere administrative scripting program (the wsadmin tool) is a powerful, non-graphical command interpreter environment that enables you to run administrative operations in a scripting language. You can also submit scripting language programs to run. The wsadmin tool is intended for production environments and unattended operations. For more information, refer to Wsadmin.

- Command-line tools

  Command-line tools are simple programs that you run from an operating system command-line prompt to perform specific tasks. Using the tools, you can start and stop servers, check server status, add or remove nodes, and complete similar tasks. For more information, refer to Administrative commands.

Security on WebSphere Process Server

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

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

Performance monitoring on WebSphere Process Server

You can monitor applications on WebSphere Process Server to understand response times, basic health of the systems running on the server and resource usage of applications.

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

Samples and tutorials

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

Samples and tutorials are available from the IBM Education Assistant Web site, and tutorials for administrative tasks are available in the IBM(R) WebSphere(R) Process Server information center.

Tutorials

You can find tutorials and their accompanying Samples using the IBM Education Assistant.

IBM Education Assistant tutorials

The IBM Education Assistant site provides tutorials that you can use at your convenience. For more information about the IBM Education Assistant, see the IBM Education Assistant Web page.

WebSphere Process Server administration tutorials

The Administering section of the WebSphere Process Server information center contains some tutorial topics to assist you with some administrative tasks.

For tutorials on administrative tasks, refer to the *Administering WebSphere Process Server* PDF file.

Accessing the Samples (Samples Gallery)

Samples that complement the WebSphere Process Server tutorials are available in the WebSphere Application Server Samples Gallery, an option to install during WebSphere Process Server installation. The Samples Gallery contains samples of simple artifacts such as those that would be generated by WebSphere Integration Developer and deployed on WebSphere Process Server.

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

1. Install WebSphere Process Server. The samples package is selected for installation by default. (If you are installing WebSphere Process Server on top of WebSphere Application Server, the base WebSphere Application Server Samples Gallery must be installed in order for you to use the samples.)

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

   Applications that are run on WebSphere Process Server have XML artifacts, such as business objects, relationship definitions, and business rules, which must be deployed before installing the application. WebSphere Process Server provides a utility named serviceDeploy to build and deploy these artifacts. The sample EAR file in *install_root*/samples/lib for each sample application contains these artifacts. The sampleDeploy utility invokes serviceDeploy with specific parameters required for the samples. Running sampleDeploy creates a second EAR file named *sample_ name*Deployed.ear in the same directory as the original EAR file. This new EAR contains the WAR files that were in the original EAR file plus the additional JAR and WAR files that contain the deployed artifacts. The deployed EAR file may be installed as an Enterprise Application in WebSphere Process Server.

   For more information, see Creating an installable EAR file using serviceDeploy in the WebSphere Process Server *Developing and Deploying Modules* PDF file.
2. Start the server.
3. Start the Samples Gallery by selecting **Samples Gallery** on the First Steps panel. WebSphere Process Server samples are initially listed as installable samples in the Samples Gallery.
4. Click each sample that you want to install and follow the instructions that appear in the browser window for installation of each sample.
5. Click **Refresh** in the Samples Gallery and the samples are listed as installed samples. For each installed sample, you can select the sample name to open a browser window with additional information an option to run the sample.

Standards compliance

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

Accessibility features

Accessibility features enable users with physical disabilities, such as restricted mobility or limited vision, to operate software products successfully. These features are built into WebSphere Process Server installation and administration features.

- Installation

  You can install WebSphere Process Server either in graphical or silent form. The silent installation program is recommended for users with accessibility needs.

For instructions, see Installing the product silently in the WebSphere Process Server *Installing* PDF file.

- 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 or Netscape Navigator, 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.

Federal Information Processing Standard

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

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

To enable FIPS for WebSphere Application Server, see Configuring Federal Information Processing Standard Java Secure Socket Extension files. When you enable FIPS, several components of the Application 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

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

The National Institute of Science and Technology (NIST) has developed Common Criteria to insure you have a safe option for downloading software to use on your systems. Information held by IT products or systems is a critical resource that enables organizations to succeed in their mission. Additionally, individuals have a

reasonable expectation that their personal information contained in IT products or systems remain private, be available to them as needed, and not be subject to unauthorized modification. IT products or systems should perform their functions while exercising proper control of the information to ensure it is protected against hazards such as unwanted or unwarranted dissemination, alteration, or loss. The term IT security is used to cover prevention and mitigation of these and similar hazards.

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

More information about Common Criteria can be found on the Support site, including Recommended Updates for WebSphere Application Server and WebSphere Process Server.

Internationalization

An application that can present information to users according to regional cultural conventions is said to be internationalized: The application can be configured to interact with users from different localities in culturally appropriate ways.

If a product has been translated to the chosen language, users of an internationalized application see error messages and interface elements in the chosen language. Date and time formats, as well as currencies, are presented appropriately for users in the specified region.

IBM(R) WebSphere(R) Process Server, version 6.0, is available only in English but provides basic enablement support for all locales (data can be entered in other languages, but messages and interface elements are in English).

Historically, the creation of internationalized applications has been restricted to large corporations writing complex systems. However, given the rise in distributed computing and in the use of the World Wide Web, application developers are pressured to internationalize a much wider variety of applications. This trend requires making internationalization techniques much more accessible to application developers.

Internationalization of an application is driven by two variables, the time zone and the locale. The time zone indicates how to compute the local time as an offset from a standard time like Greenwich Mean Time. The locale is a collection of information about language, currency, and the conventions for presenting information like dates. A time zone can cover many locales, and a single locale can span time zones. With both time zone and locale, the date, time, currency, and language for users in a specific region can be determined.

Localization of interface strings

You can localize the application for each locale that must be supported by the application.

In a localized application, the locale determines the message catalog from which the application retrieves message strings. Instead of printing an error message, the

application represents the error message with some language-neutral information; in the simplest case, each error condition corresponds to a key. To print a usable error message, the application looks up the key in a message catalog. Each message catalog is a list of keys with associated strings. Different message catalogs provide strings for the different languages that are supported. The application looks up the key in the appropriate catalog, retrieves the corresponding error message in the requested language, and prints the string for the user.

IBM(R) WebSphere(R) Process Server, version 6.0, is available only in English but provides basic enablement support for all locales (data can be entered in other languages, but messages and interface elements are in English).

Localization of text can be used for far more than translating error messages. For example, by using keys to represent each element in a graphical user interface (GUI) and by providing the appropriate message catalogs, the GUI (buttons, menus, and so on) can support multiple languages. Extending support to additional languages requires that you provide message catalogs for those languages; in many cases, the application needs no further modification.

The localizable-text package is a set of Java classes and interfaces that can be used to localize the strings in distributed applications easily. Language-specific string catalogs can be stored centrally so that they can be maintained efficiently.

Internationalization challenges in distributed applications

With the advent of Internet-based business computational models, applications increasingly consist of clients and servers that operate in different geographical regions.

These differences introduce the following challenges to the task of designing a solid client-server infrastructure.

Clients and servers can run on computers that have different endian architectures or code sets

Clients and servers can reside in computers that have different endian architectures: A client can reside in a little-endian CPU, while the server code runs in a big-endian one. A client might want to call a business method on a server running in a code set different from that of the client.

A client-server infrastructure must define precise endian and code-set tracking and conversion rules. The Java platform has nearly eliminated these problems in a unique way by relying on its Java virtual machine (JVM), which encodes all of the string data in UCS-2 format and externalizes everything in big-endian format. The JVM uses a set of platform-specific programs for interfacing with the native platform. These programs perform any necessary code set conversions between UCS-2 and the native code set of a platform.

Clients and servers can run on computers with different locale settings

Client and server processes can use different locale settings. For example, a Spanish client might call a business method upon an object that resides on an American English server. Some business methods are locale-sensitive in nature; for example, given a business method that returns a sorted list of strings, the Spanish client expects that list to be sorted according to the Spanish collating sequence, not in the English collating sequence of the server. Because data retrieval and sorting procedures run on the server, the locale of the client must be available to perform a legitimate sort.

A similar consideration applies in instances where the server has to return strings containing date, time, currency, exception messages, and so on, that are formatted according to the cultural expectations of the client.

Clients and servers can reside in different time zones

Client and server processes can run in different time zones. To date, all internationalization literature and resources concentrate mainly on code set and locale-related issues. They have generally ignored the time zone issue, even though business methods can be sensitive to time zone as well as to locale.

For example, suppose that a vendor makes the claim that orders received before 2:00 PM are processed by 5:00 PM the same day. The times given, of course, are in the time zone of the server that is processing the order. It is important to know the time zone of the client to give customers in other time zones the correct times for same-day processing.

Other time zone-sensitive operations include time stamping messages logged to a server, and accessing file or database resources. The concept of Daylight Savings Time further complicates the time zone issue.

Java 2 Platform, Enterprise Edition (J2EE) provides support for application components that run on computers with differing endian architecture and code sets. It does not provide dedicated support for application components that run on computers with different locales or time zones.

The conventional method for solving locale and time zone mismatches across remote application components is to pass one or more extra parameters on all business methods needed to convey the client-side locale or time zone to the server. Although simple, this technique has the following limitations when used in Enterprise JavaBeans (EJB) applications:

- It is intrusive because it requires that one or more parameters be added to all bean methods in the call chain to locale-sensitive or time zone-sensitive methods.
- It is inherently error-prone.
- It is impracticable within applications that do not support modification, such as legacy applications.

The internationalization service addresses the challenges posed by locale and time zone mismatch without incurring the limitations of conventional techniques. The service systematically manages the distribution of internationalization contexts across the various components of EJB applications, including client applications, enterprise beans, and servlets. For more information, see the WebSphere Application Server Network Deployment information center.

Bidirectional language support

WebSphere Process Server uses the bidirectional language format of ILYNN (implicit, left-to-right, on, off, nominal), which is also the Windows bidirectional language format. All other bidirectional language formats must be converted prior to being introduced to WebSphere Process Server.

Bidirectional attributes

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

Letter Position	Purpose	Values	Description	Default Setting
1	Order Schema	I or V	Implicit (Logical) or Visual	I
2	Orientation	L or R	Left-to-Right or Right-to-Left	L
3	Symmetric Swapping	Y or N	Symmetric Swapping is on or off	Y
4	Shaping	Y or N	Text is shaped or not shaped	N
5	Numeric Shaping	H, C, N	Hindi, Contextual, Nominal	N

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

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

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

Note: The locale setting of the user interface (browser) defines the bidirectional language display and edit format. WebSphere Process Server user interfaces must convert locale specific formats to the WebSphere Process Server default format.

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

Chapter 2. Example: Using the bidirectional (bi-di) transformation on DataObject-type data

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

Note: If you are using JDK 1.4.1 for Windows , AIX or Linux operating systems, you will not have to install these classes.

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

Note: If you are unfamiliar with the formats, see Chapter 4, "Values for the bidirectional (bi-di) format string," on page 25.

1. Include all bi-di classes that contain the bi-di engine implementation.

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

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

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

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

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

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

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

5. Define the recursive function that applies the bi-di transformation on the entire DataObject.

   Note: The following explain basic assumptions that code logic follows.
   - bi-di transformation is applied on properties of string type only
   - the properties of type string in the DataObject are stored in one bi-di format

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

   Type type;
   Property property;

   if (boIn == null) return null;

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

   a. Skip all non-string properties.

      Note: The code below follows the indentation format from step above.

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

          if (property.isContainment()) {
               if (property.isMany()) {
                     List childsList = boIn.getList(property);
                     for (int childNumber = 0; childNumber < childsList.size();
        childNumber++){
```

   b. Recursively call the (the following function?) transformation to handle children objects.

      Note: The code below follows the indentation format from substep above.

```
                      BiDiDataObjTransformationBO(connectionContext,
      ((DataObject)childsList.get(childNumber)),formatIn, formatOut);
                    }
               } else {
```

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

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

   d. Transform the simple string attributes.

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

Chapter 3. Example: Using bidirectional (bi-di) transformation on string-type data

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

Make sure that all the classes that contain the bi-di engine implementation are installed on the server on which you are developing your modules.

Note: If you are using JDK 1.4.1 for Windows , AIX or Linux operating systems, you will not have to install these classes.

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

Note: If you are unfamiliar with the formats, see Chapter 4, "Values for the bidirectional (bi-di) format string," on page 25.

1. Include all bi-di classes that contain the bi-di engine implementation.

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

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

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

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

3. Call the BidiStringTransformation function.

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

   a. Test if input string is null.

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

   b. Perform transformation

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

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

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

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

Chapter 4. Values for the bidirectional (bi-di) format string

The values for the bidirectional (bi-di) format string control the transformation of bidirectional scripts from one format to another.

Purpose

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

Table 1. Bi-Di format string values

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

Notices

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

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

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

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

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

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

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, a nd 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.

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

i5/OS
IBM
the IBM logo
AIX
AIX 5L
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
HelpNow
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
Notes
OS/390
OS/400
Passport Advantage
pSeries
Redbooks
SupportPac
WebSphere
z/OS

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

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

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

WebSphere Process Server, Version 6.0

Printed in USA