WebSphere® Business Integration Server Foundation, Version 5.1

Process Choreographer

Note
Before using this information, be sure to read the general information under "Notices" on page 239.

Compilation date: April 16, 2004

Contents

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-0206.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

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.

Chapter 1. Using process choreographer

With process choreographer, IBM WebSphere Application Server can choreograph intra-enterprise and inter-enterprise services using business processes. These processes can involve both human and IT resources. The types of processes can vary greatly, ranging from Web services or Web page navigation to business transaction support. Processes can be automatic processes or processes that require human interaction.

Using process choreographer, you can combine business-process technology with the other services that are available in WebSphere Application Server, that is, services offered by the Java 2 Platform Enterprise Edition (J2EE) architecture. You can script enterprise beans to manipulate processes. You can also create Web services from business processes that choreograph other Web services processes. For a detailed description of the process choreographer architecture, refer to the *Process Choreographer Concepts and Architecture* White paper in DeveloperWorks WebSphere

The process choreographer Web client provides a Web browser interface for work items involving people. For a high-level view of how you can use process choreographer to integrate your business processes, see the process choreographer overview.

If you want to use process choreographer in a distributed WebSphere environment, see process choreographer and network deployment and process choreographer scenarios for clustering.

To define your business processes, and create process modules that can be installed in a WebSphere business process container, use IBM WebSphere Studio Application Developer Integration Edition.

Using, managing, and developing business-process applications is described in the following topics.

You can find supplemental information about process choreographer listed in Process choreographer: Resources for learning.

- Planning to use process choreographer
- Configuring the business process container
- Uninstalling the business process container
- Configuring the staff service for process choreographer
- Administering process choreographer
- Managing business processes
- Using the process choreographer Web client
- 5.1 + Developing applications for BPEL-based processes
- Developing applications for V5.0-style processes
- Troubleshooting process choreographer

Process choreographer overview

Process choreographer is a powerful enterprise workflow tool that supports business processes in a Java 2 Platform Enterprise Edition (J2EE) environment. These processes can be used to integrate J2EE resources, Web services, and activities that require human interaction. Process choreographer manages the life cycle of business processes, navigates through the associated process model, and invokes the appropriate Web services.

Process choreographer supports:

- 5.1 + Processes based on Business Process Execution Language for Web Services (BPEL4WS, abbreviated to BPEL)

  BPEL is an XML-based workflow definition language with which businesses can describe sophisticated business processes that use and provide Web services.

  Process choreographer supports a subset of BPEL 1.1. It also supports IBM extensions to the BPEL language to provide, for example, human interaction capabilities, and activities for running inline Java code. Business processes that are specified in BPEL interoperate with the Web services provided by partners, regardless of whether these Web services are based on BPEL.

  For more information on BPEL, refer to the BPEL specification.
- Version 5.0-style processes based on the Flow Description Markup Language (FDML)

  In Version 5.1, process choreographer can run processes created with WebSphere Studio Version 5.0. Support for these processes is deprecated in Version 5.1. You can continue to run these processes or you can migrate them to BPEL processes. For information on how to migrate V5.0-style processes to BPEL-based processes, refer to the WebSphere Studio information center.

Process choreographer includes a Web client for process administrators and process participants to work with processes, activities, and their associated work items.

Process choreographer provides both generic programming interfaces and process-specific programming interfaces. When you model a process in WebSphere Studio, you can generate an associated Enterprise JavaBeans (EJB) module that provides strongly-typed interfaces, for example, for starting instances of a specific process. These interfaces simplify the coding of interactions with a specific process model.

About business processes

A process is a set of business-related activities that are invoked in a specific sequence to achieve a business goal. A process based on the Business Process Execution Language (BPEL) comprises:

- The activities that are the individual business tasks within the process. An activity can be one of several different types. Also, an activity can be categorized as either a basic activity or a structured activity.
  - Basic activities are activities that have no structure and do not contain other activities. Examples of basic activities are invoke, receive, assign, reply, and empty activities.
  - Structured activities are activities that contain other activities, such as sequence, pick, flow, switch, scope, and while activities.
- The partner links that are the logical names for external entities, partners, or services that interact with the process or vice versa.

- The variables that store the messages that are passed between activities. They represent the state of a business process instance.
- Correlation sets that are used to correlate multiple service requests or response messages with the same business process instance. Correlation sets are based on application data contained in Web Services Description Language (WSDL) message parts.
- Fault handlers that deal with exceptional situations that can occur when a business process runs.

For more information on these constructs, refer to the BPEL specification.

Process choreographer also supports the IBM extensions to the BPEL language, for example, staff activities for human interaction and script activities for running inline code. These extensions also include:

- Valid-from time stamps for process model versioning
- Business relevance flag for determining which events are recorded in the audit log
- Explicit checkpointing to support multiple activities in one transaction
- Timeouts for activities

Staff activities

Staff activities are assigned to various people in your organization through work items. Staff activities can be almost any business task: completing a form, approving a document or drawing, writing a letter, and so on.

The process is the design for how a series of tasks is done. When a process is started (by someone in your organization or even anonymously by someone completing a form at a Web site), work items are created for the potential owners, that is, all the people who can work on a particular activity.

When you claim an activity, you become the owner of that activity. Only you and the business process administrator can work on that activity in that particular instance of the process. If your work is complex or involved, you can save intermediate stages of it. When you finish the work, you complete the activity. The resulting information is saved and is then available to subsequent activities in the process. Navigation of the process continues until all of the activities are complete.

The following Business Process Execution Language (BPEL) code snippet shows how a staff activity might be implemented in a business process:

```
<bpel:invoke partnerLink="null"
   portType="QName" operation="ncName"
   inputVariable="ncname" outputVariable="ncname">

  <wpc:staff>
   <wpc:potentialOwner>staffQueryVerb</wpc:potentialOwner>
   <wpc:editor>staffQueryVerb</wpc:editor>
   <wpc:reader>staffQueryVerb</wpc:reader>

   <!-- Client-specific settings go here -->

  </wpc:staff>

 <!-- standard activity elements go here -->
</bpel:invoke>
```

Script activities

Script activities are used to run inline Java code as an activity of the process. The Java code can access all BPEL variables, correlation properties, and partner links, as well as process and activity contexts.

You can also use inline Java code for conditions. Each Java code snippet for activities or conditions is run in its own Java method. These methods belong to an enterprise bean that is created for the process. The run-time environment of the Java code is the Java 2 Platform Enterprise Edition (J2EE) environment of regular Enterprise JavaBeans (EJB).

The following BPEL code snippet shows a script activity with inline Java code for an activity:

```
<bpel:invoke  ....>
  <wpc:script>

   <wpc:javaCode>
     EObject myCustomer = getPurchaseOrder().eGet("customer") ;
     EObject account = getAccount() ;
     account.eSet("ID", myCustomer.eGet("accountID") ) ;
     setAccount(account) ;
   </wpc:javaCode>
  </wpc:script>

 <!-- standard activity elements go here -->
</bpel:invoke>
```

The following BPEL code snippet shows a script activity with inline Java code for a condition:

```
<bpel:while>
  <wpc:condition>
   <wpc:javaCode>
     return getCounter().getValue() > 0 ;
   </wpc:javaCode>
  </wpc:condition>

 <!-- loop activity goes here -->
</bpel:while>
```

Business process types

You can model both V5.0-style processes and processes that are based on the Business Process Execution Language (BPEL) to be either interruptible processes or non-interruptible processes. For interruptible processes, the intermediate process state is persisted into the process database. With this persistence, the process is forward-recoverable, which is a characteristic property of workflow-based applications. Alternatively, eligible processes can run as non-interruptible processes.

Interruptible processes

A business process is interruptible if each step of the process is run in its own physical transaction. Interruptible processes are generally long-running processes. Business processes need to be interruptible if they wait for external stimuli or if they involve human interaction. Examples of external stimuli are events sent by another business process in a business-to-business interaction, responses to asynchronous invocations, or the completion of a staff activity.

An interruptible process has the following characteristics:
- Runs as several transactions.
- Consists of synchronous and asynchronous services.
- Is started by the initiate method or the sendMessage method because the output message cannot be retrieved synchronously.
- Normally runs a long time.
- Stores run-time values persistently.

Non-interruptible processes

A non-interruptible business process runs in one physical thread from start to finish without interruption. Non-interruptible business processes are also known as *microflows*. Non-interruptible processes can have different transactional capabilities. A non-interruptible process can run within a distributed transaction, as part of an activity session, or with a local transaction.

A non-interruptible business process has the following characteristics:
- Runs as one transaction.
- Consists of only synchronous services and non-interruptible subprocesses. This means that a non-interruptible process cannot contain:
  - Staff or wait activities.
  - An asynchronous invoke.
  - Multiple receive activities.
  - Interruptible subprocesses.
- Is started using the call method so that an output message is returned when the process completes.
- Normally runs for a short time.
- Does not store run-time values in the database.
- Contains no interruptible processes.

Compensation in process choreographer

You can use WebSphere Studio to define compensation properties for business processes and activities. Compensation processing starts as a result of an error that occurred during the run of a process instance for which compensation is defined in the process model. The aim is to get back to a consistent state and to compensate for any operations that were committed up to when the error occurred.

You can specify compensation for interruptible processes and for non-interruptible processes.

Compensation for interruptible processes

Compensation is triggered if the overall process instance ends with a business error, or the process instance was explicitly terminated with a request for compensation. If compensable activities (activities with associated undo actions) have successfully completed, they are undone by calling the undo actions in the reverse order of the forward navigation. The sphere of compensation spans all subprocesses of the overall process.

Compensation for non-interruptible processes

Compensation is triggered on rollback of the work unit (the transaction or the activity session) that contains the process. Therefore, undo actions are typically specified for activities that cannot be reversed by rolling back the unit of work. When a process instance runs, undo actions for compensable activities are registered with the enclosing unit of work. Depending on the outcome of this unit of work (rollback or commit), compensation starts.

If the non-interruptible process is a child of a compensable interruptible process, the undo actions of the non-interruptible process are promoted to the parent process when the non-interruptible process completes. It can, therefore, potentially participate in the compensation of the parent process. It is recommended that you also specify undo actions for the activities of the non-interruptible process.

If an error occurs during compensation processing, the compensation action requires manual resolution to overcome the error. You can use the process choreographer Web client to repair these compensation actions in error.

Process choreographer scenarios for clustering

The main advantages of using WebSphere clusters and Network Deployment (ND) to create and administer process choreographer instances are:
- Increased workload capacity.
- Improved resource utilization.
- Workload sharing.
- Easier administration.

Configuration options

You can configure process choreographer in many different ways, so cluster configurations are usually very complex. Some of the main options to consider before you start creating application servers are outlined in the following descriptions:

Number of nodes in the WebSphere cell
One or more. All nodes are administered from a single deployment manager.

Number of nodes in each WebSphere cluster
One or more. Horizontal WebSphere clustering can increase service availability and increase the total workload capacity.

Number of application servers in each node
One or more. Vertical WebSphere clustering can increase resource utilization.

Database host
- Remote, on a dedicated machine
- Local to one of the application servers in the cluster

To share workloads, all business process containers in the same WebSphere cluster must use the same database. Hosting the database on a dedicated machine, preferably one with a hot standby, is recommended.

Application messaging queues
- Local queues
- Remote queues

Connection (WebSphere MQ queue managers)
- One central (remote) queue manager hosting the queues for the application servers within one cluster.
- One local queue manager per application server.
- Two local queue managers per node, and WebSphere MQ clustering used to balance workload across several application servers.

  Workload balancing between different process choreographer instances requires that the queue managers used by the business process container of each application server are members of the same WebSphere MQ cluster.

Database system
You can use any of the supported databases except Cloudscape.

Hot standby machines
You have the following options for hot standby machines:
- None
- For the database
- For a central queue manager

Cluster types: This topic refers to two different types of *cluster*. A *WebSphere cluster* groups application servers together to share workload and increase service availability. A *WebSphere MQ cluster*, previously known as an MQSeries cluster, groups together WebSphere MQ queue managers and can be used to achieve intraprocess workload balancing.

High availability

To achieve high availability of process choreographer services, consider the following:
- By creating cloned application servers in a WebSphere cluster, the services provided by the application servers become highly available.
- The process choreographer database is a single point of failure that can be protected using a hot standby system.
- A central queue manager can be protected by hot standby hardware.

Vertical clustering to maximize resource utilization

To improve performance, you might have to create multiple application server instances on the same node so that process choreographer can use the available system resources.

Workload balancing

If you want different instances of process choreographer to share the same workloads, they must:
- Use the same process choreographer database.
- Use one of the following queue manager configurations:

  Central queue manager
  A central queue manager hosts the four queues that are needed by process choreographer. All process choreographer instances in the WebSphere cluster read from the same queues.

  WebSphere MQ cluster
  Each application server has two queue managers. One queue manager hosts four local queues and is used for getting messages, the other queue manager hosts no queues and is only used for putting messages. All the queue managers of all the process choreographer instances in the WebSphere cluster are made members of a WebSphere MQ cluster.

  The result of only putting to queue managers that host no queues is that the messages are distributed evenly across all the ″get″ queue managers in the cluster. After using the installation wizard to install and configure the business process container on the cluster, you must manually change the two connection factories per application server to point to the local ″get″ and ″put″ queue managers.

For more details about WebSphere clustering, see the information on balancing workloads with clusters in *WebSphere Business Integration Server Foundation Applications* PDF.

Process choreographer database

Hosting the database on a dedicated machine, preferably one with a hot standby is recommended. The database can be on a machine that is outside the WebSphere cell, however the deployment manager must have access to it.

When planning the database, consider the following points:

- All business process containers in the same WebSphere cluster access the same database. By contrast, any business process container that is not in a WebSphere cluster must have its own database.
- To enable access to a remote process choreographer database, you must install the appropriate database client, or a type-4 Java database connectivity (JDBC) driver, on all application servers that do not have a local database.
- The deployment manager requires access to all databases for process choreographer instances in the WebSphere cell, regardless of whether they are in a cluster, or not. You must enable this access before you can use the deployment manager to install a business process.
- Your database can be any of the supported databases except Cloudscape, because Cloudscape Network Server has no XA support and embedded Cloudscape cannot be accessed remotely.
- Each database, used by process choreographer instances in the same WebSphere cell, must be accessible using a unique name. The same database name must be used on the deployment manager and on the application server.
- The database is a single point of failure. This problem can be solved only by using a high-availability hot standby solution such as High Availability Cluster Multiprocessing (HACMP) on AIX.

WebSphere MQ

Process choreographer uses MQ queues for receiving requests and sending replies. Therefore each application server that hosts process choreographer requires one of the following options:

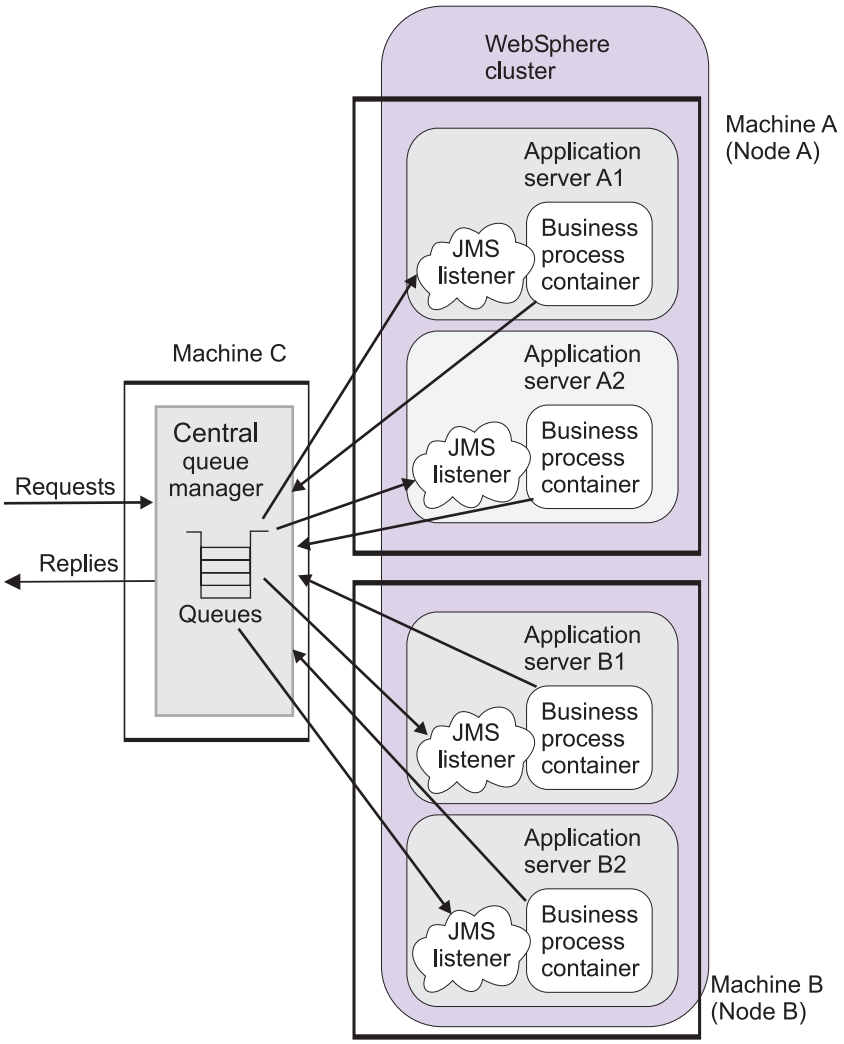
- Access to a central queue manager that hosts all queues
- A local queue manager that is not a member of a WebSphere MQ cluster
- Two local queue managers that are members of a WebSphere MQ cluster

Process choreographer also supports embedded messaging. However, embedded messaging is not recommended for complex configurations because it does not support WebSphere MQ clustering.

Central queue manager

By using a central queue manager for all queues, administration becomes easier. One queue manager is used by all cloned business process containers. However, using a central queue manager creates a single point of failure that needs to be hosted on a high availability system.

The following figure shows all the application servers in a WebSphere cluster using a single central queue manager on another machine.



Local queue manager without WebSphere MQ clustering

This is the standard, stand-alone process choreographer configuration. Each business process container uses its own (local or remote) database, and has one local queue manager. This approach does not offer intraprocess workload sharing.

WebSphere MQ clustering

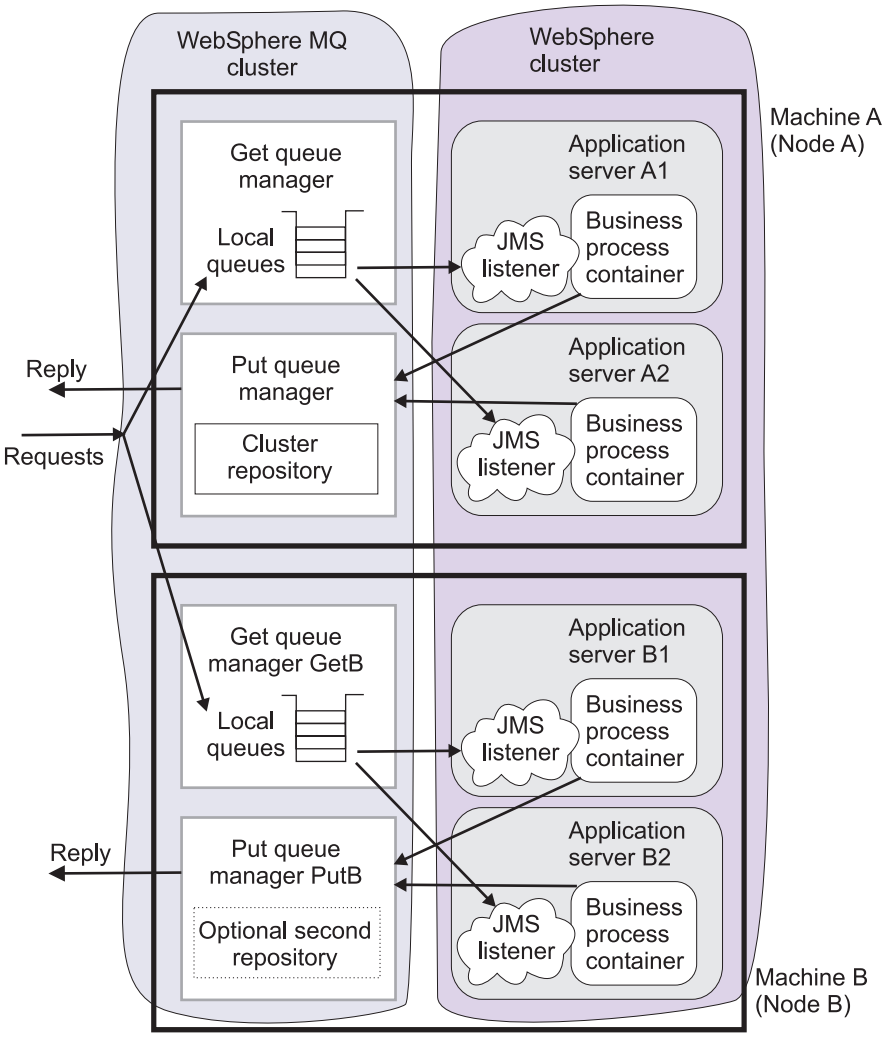
This complex technique supports intraprocess workload sharing for process choreographer services in a WebSphere cluster. The business processes in the cluster must all run on UNIX only, or Windows machines only; a combination of UNIX and Windows machines does not work.

Each application server requires two local queue managers, one for putting and one for getting messages. All the queue managers become members of the same WebSphere MQ cluster. On Windows systems, all the queue managers must use the same binding protocol. On UNIX systems, the put and get queue managers must use different protocols. For example, you can modify the queue connection factories so that all the put queue managers use the binding protocol (interprocess communications) and all the get queue managers use the default, client (TCP/IP) protocol.

Each business process container in the WebSphere cluster must be customized to reflect its own queue managers.

It is recommended that more than one queue manager in the WebSphere MQ cluster is made a cluster repository.

The following figure shows how the queue managers used by the application servers are grouped together in a WebSphere MQ cluster. The WebSphere MQ cluster (of queue managers) is parallel to the WebSphere cluster of application servers. Requests are shared across the get queues in the cluster.



How the WebSphere cluster is created

Several different sequences are available for you to follow when creating a cluster for process choreographer. The following sequence is recommended:

1. Install the prerequisites on each machine:
   - WebSphere MQ.
   - Database server, database client, or a type-4 Java database connectivity (JDBC) driver.
2. On the first node in the cell, define the application servers.

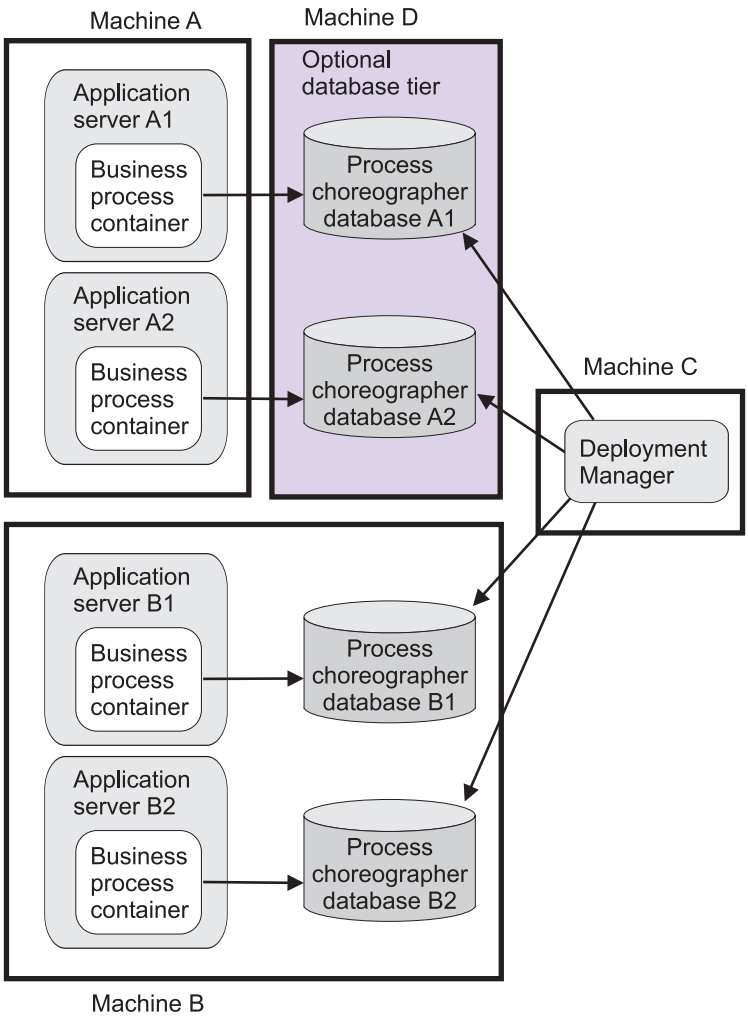
3. Create the required clones of the application server.
4. Use the Install wizard on the deployment manager machine to install the business process container on any one of the application servers in the cluster. The business process container is simultaneously installed on all the other application servers in the cluster.
5. If your WebSphere MQ configuration is a WebSphere MQ cluster of local queue managers you must modify the connection factories. Because each queue manager has a different name, you must modify the connection factories in each of the cloned application servers to reflect its unique differences from the cluster-wide, standard process choreographer Install wizard configuration.

Process choreographer and Network Deployment

The main advantage of using Network Deployment (ND) to create and administer process choreographer instances is that it makes clustering scenarios possible, where all the business process containers in a cluster are administered centrally.

Deployment manager must have access to the process choreographer database

The deployment manager must have access to all the process choreographer databases that are used by business process containers in the cell. You must install an appropriate database client on the deployment manager machine, and add the database driver to the deployment manager class path. The following figure shows this configuration.



Customization required after installing and configuring process choreographer on a cluster

If you are creating a clustered setup that uses WebSphere MQ clusters of queue managers, you must perform some manual customization to make each process choreographer instance use its own queue managers. The necessary actions are described in Configuring the business process container.

For more information about using clustering with process choreographer, see process choreographer scenarios for clustering

Installing a business process application 5.1 +

When you install a business process application, you add configuration data to the WebSphere configuration repository and process metadata to the process choreographer databases.

For Version 5.0-style business processes, the process modules (FAR files) are not distributed to the deployment targets where you install any of the Enterprise JavaBeans (EJB) modules or Web modules belonging to your application.

Processes based on the Business Process Execution Language (BPEL) are modeled as enterprise beans and are therefore part of the EJB modules. When a business process application is installed, the processes are deployed to the targets where the containing EJB module is installed. If the process contains interruptible processes, you can install the process application on only one deployment target.

If only some of the required servers are available at installation, install the application on the servers and the WebSphere clusters that are running. You can map the application to additional servers and WebSphere clusters later. See Editing a business process application for details.

Restrictions when editing a business process application

When you edit a business process application, you change the mapping of the application modules to the application servers and the WebSphere clusters in the cell. You can use the administrative console or administration scripts to change this mapping.

Only the EJB modules and Web modules appear in the administrative console. If you change the mapping of an EJB module to a deployment target, then the processes associated with the module are written to the databases that belong to the new deployment target. The processes are not automatically deleted from the database that belongs to the previous deployment target. These processes are only deleted if you remove all the modules belonging to the process application from the deployment target.

When you edit a business process:

- All application servers and at least one member of each WebSphere cluster that you want to change (remove or add) must be running. The associated database servers must be running also.
- If you remove the mapping of a module to an application server or a WebSphere cluster, you must stop the process templates of that application. All instances of the templates must be removed from the database belonging to the corresponding application server or WebSphere cluster.

Chapter 2. Planning to use process choreographer

For each application server where you want to use process choreographer, you must configure the business process container before installing any enterprise applications that contain processes. Before configuring the business process container, consider the following items:

1. To gain more understanding about the process choreographer architecture, refer to the architecture white paper in the DeveloperWorks WebSphere.
2. Decide which database system to use:
   - Cloudscape
   - 5.1+ DB2 UDB for iSeries
   - DB2 UDB for Linux, UNIX, and Windows
   - DB2 for z/OS
   - 5.1+ Informix Dynamic Server
   - Microsoft SQL Server
   - Oracle
   - Sybase Adaptive Server Enterprise; process choreographer uses distributed transactions, so you need to purchase and install the Distributed Transaction Management (DTM) feature for Sybase Adaptive Server Enterprise.
3. Check the required software levels for the supported database version and JDBC driver.
4. Decide which machine will host the database. If the database machine is remote, you need a suitable database client or a type-4 JDBC driver that has XA-support.
5. If you are not are not going to create a WebSphere cluster setup, decide whether you want to use the Java Message Service (JMS) API provided by:
   - The messaging service that is embedded in WebSphere Application Server.
   - A separate WebSphere MQ installation.

   Embedded messaging: If you intend to use embedded messaging, you must have selected this option when you installed WebSphere Application Server. If you want to use WebSphere MQ messaging, it must be installed before you start configuring the business process container. Embedded messaging is not supported in a cluster setup.
6. If you intend to use process choreographer in a WebSphere cluster environment, plan your process choreographer cluster.
7. Decide if you will use the Install Wizard (recommended) rather than configuring the business process container manually:
   - If you are going to use the Install Wizard, plan the Install Wizard settings.
   - If you are going to configure the business process container manually, plan the business process container settings.

You are ready to configure the business process container.

Chapter 3. Configuring the business process container

Use this task to configure the business process container.

Complete planning to use process choreographer.

You must configure the necessary resources and install the business process container before you can use process choreographer.

1. If you are preparing a clustered process choreographer setup, create the cluster. See *WebSphere Business Integration Server Foundation Applications* PDF for more information on creating clusters.
2. 5.1 + If you are using Java 2 security in a Network Deployment environment, perform granting permission to the JDBC driver on the deployment manager.
3. If you are using an external Java Message Service (JMS) provider, WebSphere MQ, create the queue manager and queues.
4. Create the database.
5. If you are preparing a clustered process choreographer setup perform Configuring the business process container on a cluster.
6. If you are not preparing a clustered process choreographer setup, decide whether you want to configure the business process container using the **Install Wizard** (recommended) or **manually**, and perform one of the following actions:
   - Using the Install Wizard to configure the business process container.
   - Configuring the business process container manually.
7. 5.1 + Setting the JNDI name of the calendar EJB for process choreographer.
8. Activate the business process container.
9. Verify that the business process container works. In case of problems, see troubleshooting the business process container.

The business process container is configured and working.

Now you can install and run enterprise applications that contain processes, as described in managing business processes.

Creating the database for the business process container

The business process container requires a database. This topic describes how to create the database for process choreographer.

Your database system must already be installed and your application server must be configured to use the database. In a clustered process choreographer setup, one database serves all the business process containers in the WebSphere cluster. In a non-clustered setup, the database is dedicated to the business process container on one application server.

1. If your database server is not on the same machine as your Enterprise Application Server, copy the DDL scripts for your database system to your database server machine. On Windows systems, copy the DDL files from *install_root*\ProcessChoreographer. On UNIX systems, copy these files from *install_root*/ProcessChoreographer.

2. If you will use a UNIX machine to host the database server, create a dedicated file system for the databases.
3. On the machine hosting the database server, create the database according to the description for your database system.
   - Creating a Cloudscape database.
   - 5.1+ Creating a DB2 Universal Database for iSeries database.
   - Creating a DB2 Universal Database for Linux, UNIX, and Windows database.
   - Creating a DB2 Universal Database for z/OS database.
   - 5.1+ Creating an Informix Dynamic Server database.
   - Creating a Mircrosoft SQL Server database..
   - Creating an Oracle database.
   - Creating a Sybase Adaptive Server Enterprise database.
4. On each machine that will run process choreographer without a local database, you must make the remote database accessible:
   a. Install a suitable database client or JDBC driver on the application server machine.
   b. If you are not using a type-4 JDBC driver, make the new database know to the database client as follows:
      Cloudscape
      Process choreographer only supports embedded Cloudscape, which does not allow remote access. Cloudscape Network Server is not supported because it has no XA support.
      For DB2 Universal Database
      The database must be cataloged and accessible through an alias name.
      For Informix Dynamic Server
      This step does not apply because only the type-4 JDBC provider is supported.
      For Microsoft SQL Server
      The node and internet protocol port number that hosts the database is used as the key to access the database.
      For Oracle
      The TCP net service name (TNS) is used to access the database.
      For Sybase Adaptive Server Enterprise
      The node and internet protocol port number that hosts the database is used as the key to access the database.
   c. Verify that you can access the remote database.
5. If you are preparing a clustered process choreographer setup, or if you intend to use Network Deployment, you must also make the database accessible to the deployment manager. On the deployment manager machine, perform the following actions:
   a. Install a suitable database client or JDBC driver on the Enterprise Application Server machine.
   b. If you are not using a type-4 JDBC driver, make the new database know to the database client as follows:
      Cloudscape
      Process choreographer only supports embedded Cloudscape, which does not allow remote access. Cloudscape Network Server is not supported because it has no XA support.

For DB2 Universal Database
The database must be cataloged and accessible through an alias name.

For Informix Dynamic Server
This step does not apply because only the type-4 JDBC provider is supported.

For Microsoft SQL Server
The node and internet protocol port number that hosts the database is used as the key to access the database.

For Oracle
The TCP net service name (TNS) is used to access the database.

For Sybase Adaptive Server Enterprise
The node and internet protocol port number that hosts the database is used as the key to access the database.

c. Locate and install the appropriate JDBC driver for your database system. For the driver name, refer to the default values for Classpath (data source) for your database.

d. Click on **System Administration> Deployment Manager> Process Definition> Java Virtual Machine> Configuration Page** and add the database driver to the **Classpath** of the deployment manager.

e. Restart the deployment manager.

The process choreographer database exists and is accessible from the application server machine.

Creating a Cloudscape database for process choreographer

The Cloudscape database system is implemented in the language Java. It comes with the WebSphere Application Server package as several Java Archive (JAR) files (`db2j.jar`, `db2jcc.jar`, `db2jcc_license_c.jar`, `db2jtools.jar`, `db2jcview.jar` and `db2jnet.jar`).

The Cloudscape license that comes with WebSphere Application Server Enterprise Edition is only for development and test, not for production purposes. The Cloudscape version that comes with this product includes the Cloudscape Network Server that allows client/server JDBC access over Distributed Relational Database Architecture (DRDA) protocol. Because Cloudscape Network Server has no XA support, process choreographer can only use the Embedded Cloudscape version that can not be accessed remotely. That is why Cloudscape cannot be used as database system for process choreographer in a clustered environment.

To create a Cloudscape database named `BPEDB`, perform the following:

1. Make sure that you have administrator rights on the machine where your application server is installed.
2. To use a database location other than the default location, either change to the directory where you want the new database created, or edit the `createDatabaseCloudscape.ddl` script located in the `ProcessChoreographer` subdirectory of the application server installation directory, and add the fully qualified path to the database name. For example, on Windows systems change the line:

   ```
   connect 'BPEDB;create=true' as BPEDB;
   ```

   to:

   ```
   connect 'install_root\ProcessChoreographer\BPEDB;create=true' as BPEDB
   ```

Note: If you do not use the default location, *install_root*\ProcessChoreographer\BPEDB, and you will use the business process container install wizard, you must remember to also change the custom properties field when selecting the database. You will have to change the value of the property databaseName to your fully qualified database location.

3. At the command prompt, enter the command to run the database creation script using the Cloudscape command line processor: On Windows systems, enter:

```
java -Djava.ext.dirs=install_root\cloudscape\lib
     -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
     install_root\ProcessChoreographer\createDatabaseCloudscape.ddl
```

On UNIX systems, enter:

```
java -Djava.ext.dirs=install_root/cloudscape/lib
     -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
     install_root/ProcessChoreographer/createDatabaseCloudscape.ddl
```

The Cloudscape database for process choreographer exists.

Creating a DB2 UDB for iSeries database for process choreographer

This topic describes how to create the database schema on DB2 for iSeries for process choreographer.

Perform this task on the application server machine.

1. Make sure you have DB2 Connect Gateway (Version 8.1, fix pack 3 or higher) installed. This component is part of the DB2 UDB ESE package, but you can also install it separately.
2. Catalog the remote database using the add database wizard in the configuration assistant.
3. Verify that you can establish a connection to the remote DB2 iSeries system by entering the following command:

```
db2 connect to iSeries-database user userid using password
```

where *userid* and *password* are the user ID and password that you will use to connect to the database using process choreographer.

4. If no collection has been created for the user ID you use, enter:

```
db2 create collection userid
```

5. Change to the ProcessChoreographer subdirectory in the application server installation root directory.
6. Run the createSchemaDb2iSeries.ddl script file as described in the header of the script file. If this script does not work, use the dropSchemaDb2iSeries.ddl script to drop the schema.
7. To use the Toolbox driver, download it from your iSeries system. The necessary jar file for this driver is: /QIBM/ProdData/Http/Public/jt400/lib/jt400.jar.

The DB2 for iSeries schema for process choreographer exists.

Creating a DB2 UDB for Linux, UNIX, and Windows database for process choreographer

This topic describes how to create a DB2 UDB database for process choreographer.

1. If your database server is not on the same machine as your application server, copy the following DDL scripts from the ProcessChroeographer subdirectory of your application server *install_root* directory to your database server machine.

   ```
   clearSchemaDb2.ddl
   createDatabaseDb2.ddl
   createTablespaceDb2.ddl
   createSchemaDb2.ddl
   dropSchemaDb2.ddl
   dropTablespaceDb2.ddl
   ```

2. Change to the directory where the configuration scripts for process choreographer are located:
   - If your database server is on the same machine as your Enterprise Application Server: On Windows systems, enter: `cd install_root\ProcessChoreographer`. On UNIX systems, enter: `cd install_root/ProcessChoreographer`.
   - If your database server is on a different machine than your application server, change to the directory where you copied the DDL scripts.
3. Install DB2 UDB on the machine that will host the database.
4. Install the DB2 run-time client on:
   - All remote application servers that use a type-2 JDBC driver to access the database.
   - On the deployment manager machine, if you will use Network Deployment to administer process choreographer, for example if you are creating a clustered process choreographer setup.
5. If you want to use an existing database skip to step 11 to create the table space and schema. Unicode support: Make sure that the database supports Unicode (UTF-8). Without Unicode support, it cannot store all characters that can be handled in Java code, and you can run into code page conversion problems when a client uses an incompatible code page.
6. To avoid deadlocks, verify that the DB2 flag DB2_RR_TO_RS is set to YES. If necessary, enter the command:

   ```
   db2set DB2_RR_TO_RS=YES
   ```

   then restart the DB2 instance to activate the change.
7. If you want to administer the DB2 instance remotely, create an administrative DB2 instance, db2as.
8. Create a DB2 instance on the database machine. The default is db2inst1.
9. If you have a Symmetric Multi-Processor (SMP) machine, set the number of processors that can be used by DB2. On UNIX systems, use `/usr/opt/db2_08_01/adm/db2licm -l`.
10. Create a new database named BPEDB:
    a. Make sure that you are using a user ID that has administrator rights for the database system.
    b. In the DB2 command line processor, enter the command to run the quick database creation script:

       ```
       db2 -tf createDatabaseDb2.ddl
       ```

    c. Make sure that the script output contains no errors. In some cases, the CLI packages are not bound to the new database. To be sure that the CLI packages are bound to the new database, for a database named BPEDB:

       On Windows systems, enter:

```
db2 connect to BPEDB
db2 bind %DB2PATH%\bnd\@db2cli.lst blocking all grant public
```

On UNIX systems, enter:

```
db2 connect to BPEDB
db2 bind $DB2DIR/bnd/@db2cli.lst blocking all grant public
```

A DB2 database named BPEDB has been created. This database should only be used for test purposes. For a production environment it is recommended to consider using dedicated table space containers and adjusting the DB2 parameters.

11. To create the table space and schema:
    a. Analyze the results of your experiences during development and system testing. The size of your database depends on many factors. Non-interruptible processes require very little space. Each process template can require tens or hundreds of kilobytes. If possible, distribute table space containers across different logical disks, and implement an appropriate security policy. Consider the performance implications of your choices for buffer pools and log file settings.
    b. Edit the table space creation script `createTablespaceDb2.ddl` according to the instruction at the top of the file.
    c. Make sure that you have administrator rights for the database system. The user ID you use to create the schema must be the one that you specify for WebSphere Application Server to use to access the database.
    d. Make sure that you are attached to the correct instance. Check the `DB2INSTANCE` environment variable.
    e. To connect to a database named *databaseName*, in the DB2 command line processor, enter the command:

       ```
       db2 connect to databaseName
       ```

    f. To create the table space, enter the command:

       ```
       db2 -tf createTablespaceDb2.ddl
       ```

       Make sure that the script output contains no errors. If there were any errors, you can drop the table space using the script `dropTablespaceDb2.ddl`.
    g. To create the schema (tables and views), in the DB2 command line processor, enter the command:

       ```
       db2 -tf createSchemaDb2.ddl
       ```

       Make sure that the script output contains no errors. If there were any errors, you can use the `dropSchemaDb2.ddl` file to drop the schema.
12. On each application server that remotely accesses the database (and on the deployment manager machine if you are creating a clustered setup or if you want to use Network Deployment):
    a. Catalog the database by entering the command:

       ```
       db2 catalog database databaseName as databaseAlias at node nodeName
       ```

       For more information about cataloging a database refer to the DB2 documentation.

b. Verify that you can connect to the database by entering the commands:

```
db2 connect to databaseName user userID
db2 connect reset
```

The DB2 UDB database for process choreographer exists.

Creating a DB2 UDB for z/OS database for process choreographer

This topic describes how to create a DB2 UDB for z/OS database and how to verify that it is reachable from the application server machine.

1. You must have already installed WebSphere Business Integration Server Foundation on a UNIX or Windows machine.
2. On the z/OS machine that hosts the database:
   a. Log on the native z/OS environment.
   b. If multiple DB2 systems are installed, decide which subsystem you want to use.
   c. Make a note of the IP port to which the DB2 subsystem is listening.
   d. Using the DB2 administration menu, create a new database, for example, named `BPEDB`. Note the name of the database.
   e. Create a storage group and note the name.
   f. Decide which user ID is used to connect to the database from the remote machine running WebSphere Application Server. Normally, for security reasons, this user ID is not the one you used to create the database.
   g. Grant the user ID the rights to access the database and storage group. The user ID must also have permission to create new tables for the database.
3. On the Application Server machine:
   a. Make sure that you have DB2 Connect Gateway (Version 8.1 fix pack 3 or higher) installed. This component is part of the DB2 UDB ESE package, but you can also install it separately.
   b. Catalog the remote database using the following commands, either in a script or in a DB2 command line window:

      ```
      catalog tcpip node zosnode remote hostname server IP_port ostype mvs;
      catalog database subsystem as subsystem at node zosnode authentication dcs;
      catalog dcs database subsystem as subsystem parms ',,INTERRUPT_ENABLED'
      ```

      An important difference exists between DB2 UDB and DB2 UDB for z/OS. DB2 UDB does not have the concept of a subsystem, but DB2 UDB for z/OS does. To avoid confusion between Database name and Subsystem name, it is important to understand that because DB2 UDB for z/OS runs in a subsystem, the catalog node and catalog database commands must identify the appropriate subsystem. On DB2 UDB, the subsystem name is not a known concept, so the database name that it connects to is really the name of the DB2 UDB for z/OS subsystem.
   c. Verify that you can establish a connection to the remote subsystem by entering the following command: `db2 connect to subsystem user userid using password`
   d. Change to the *ProcessChoreographer* subdirectory in the application server installation root directory.

e. Edit the `createTablespaceDb2V7zOs.ddl` script. Replace **@STG@** with the storage group name and replace **@DBNAME@** with the database name (not the subsystem name).

f. Run your customized version of `createTablespaceDb2V7zOs.ddl`, as described in the header of the script. If this script does not work, or if you want to remove the table space, use `dropTablespaceDb2V7zOs.ddl` to drop the table space.

g. Edit the `createSchemaDb2V7zOs.ddl` script. Replace **@STG@** with the storage group name and replace **@DBNAME@** with the database name (not the subsystem name).

h. Run your customized version of the `createSchemaDb2V7zOs.ddl` script, as described in the header of the script. If this script does not work, or if you want to remove the tables and views, use `dropSchemaDb2V7zOs.ddl` to drop the schema.

The DB2 UDB for z/OS database for process choreographer exists.

Creating an Informix Dynamic Server database for process choreographer

This topic describes how to create an Informix Dynamic Server database for process choreographer.

1. Make sure that you have administrator rights on the machine where you want to install Informix.
2. Install the Informix server on the machine that will host the database.
3. Create an Informix server instance. Make sure that the Informix environment variables are set correctly. In particular, `INFORMIXSERVER` must point to the new instance and `ONCONFIG` must point to the configuration file for the instance. For more details about the different environment variables and how they need to be setup, refer to the Informix Dynamic Server documentation. If you are using Informix Dynamic Server 9.4 make sure that the Global Language Support (GLS) related environment variables are set to Unicode (UTF-8) support. Unicode support is required to store all characters that can be handled in Java code. With Informix Dynamic Server 9.3 or if Unicode support is not available, make sure that the client and database code pages are compatible so that you do not have problems with code page conversion.
4. Copy and configure the JDBC driver on:
   - All remote application servers that use the database server.
   - On the deployment manager machine if you are using Network Deployment to administer process choreographer, for example, to create a clustered process choreographer setup.
5. On your application server machine, change to the directory where the configuration scripts for process choreographer are located:

   On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX systems, enter: `cd install_root/ProcessChoreographer`.
6. If your database server is not on the same machine as your application server:
   a. Copy the scripts for your operating system from the `ProcessChroeographer` directory on your application server to a suitable directory on your database server machine:

For UNIX:	For Windows:

clearSchemaInformix9.sql createDatabaseInformix9.sql createSchemaInformix9.sql dropSchemaInformix9.sql createDbspaceInformix9.sh dropDbSpaceInformix9.sh	clearSchemaInformix9.sql createDatabaseInformix9.sql createSchemaInformix9.sql dropSchemaInformix9.sql createDbspaceInformix9.bat dropDbSpaceInformix9.bat

b. Change to the directory where you copied the DDL scripts.

7. If you want to create a non-production database, named BPEDB, for stand-alone development, evaluation, or demo purposes, you only need to enter the command:

   ```
   dbaccess - createDatabaseInformix9.sql
   ```

   This creates an Informix database named BPEDB for the user ID that you are using. Make sure that the script output contains no errors. If there are any errors, you can use the dropSchemaInformix9.sql file to only drop the schema or the SQL command DROP DATABASE to drop the whole database.

8. If you prefer to create your database manually:
   a. Create a database, for example named BPEDB.
   b. Create the Dbspaces for your database.

      On Windows systems, read the instructions in the createDbspaceInformix9.bat file.

      On UNIX systems, read the instructions in the createDbspaceInformix9.sh file.

   c. Run the script to create the schema, by entering the command:

      ```
      dbaccess databaseName createSchemaInformix9.sql
      ```

      where *databaseName* is the name of the database, for example BPEDB.

   d. Check the script output for any errors. If there were errors, you can use the dropSchemaInformix9.ddl file to drop the schema.

The Informix Dynamic Server database for process choreographer exists.

Creating a Microsoft SQL Server database for process choreographer

This topic describes how to create a Microsoft SQL Server database for process choreographer.

1. Make sure that you have administrator rights.
2. On your application server machine, change to the directory where the configuration scripts for process choreographer are located: On Windows systems, enter: cd *install_root*\ProcessChoreographer. On UNIX systems, enter: cd *install_root*/ProcessChoreographer.
3. If your database server is not on the same machine as your application server:
   a. Copy the DDL scripts for your database from the ProcessChroeographer directory on your application server to a suitable directory on your database server machine:

      ```
      clearSchemaMsSql2000.ddl
      createDatabaseMsSql2000.ddl
      createSchemaMsSql2000.ddl
      dropSchemaMsSql2000.ddl
      ```

   b. Change to the directory where you copied the DDL scripts.

4. Install an SQL Server, on the machine that will host the database. Make sure you select the option to create a case-sensitive instance. If you already have an SQL Server that was created with the case-insensitive option, run the `rebuild master` tool and change the collation settings to case-sensitive.
5. Make sure that the database server and the Distributed Transaction Coordinator (DTC) are running. To enable XA connections, refer to the *WebSphere Business Integration Server Foundation Applications* PDF for instructions on how to install "Stored Procedures for JTA" from the WebSphere CD.
6. If you want to create a non-production SQL Server database, named `BPEDB`, for stand-alone development, evaluation, or demonstration purposes, you only need to enter the following command:

   ```
   isql -U userID -P password -i createDatabaseMsSql2000.ddl
   ```
7. If you prefer to create your database manually:
   a. Create the database, for example, named `BPEDB`.
   b. Run the script to create the schema, by entering the following command:

      ```
      isql -S serverName -U userID -P password
           -d databaseName -i createSchemaMsSql2000.ddl
      ```

      where:
      userID is the user ID to use.
      password
      is the password for *userID*.
      databaseName
      is the name of the database you created, for example, `BPEDB`.

The SQL Server database for process choreographer exists.

Creating an Oracle database for process choreographer

There is no script to quickly create a default Oracle database for process choreographer.

1. Install the Oracle server on the machine that hosts the database. Be sure that you are using the 32-bit Oracle libraries located in the `lib32` subdirectory.
2. For the root user, set the environment variables `ORACLE_BASE` and `ORACLE_HOME`.
3. If you are not using the thin JDBC driver, install the database client on:
   - All remote application servers that use the database server.
   - On the deployment manager machine if you are using Network Deployment to administer process choreographer, for example if you are creating a clustered Process choreographer setup.
4. On UNIX systems, create soft links to the following Oracle libraries in the `/usr/lib` directory:
   - For Oracle 8i: Link to: `libwtc8.so`, `libclntsh.so.8.0`, and `libocijdbc8.so`.
   - For Oracle 9i: Link to: `libwtc9.so`, `libclntsh.so.9.0`, and `libocijdbc9.so`.
5. Create an Oracle database using the Database Configuration Assistant, for example with the name `BPEDB`. Make sure that you select the JServer option for the database. It is recommended that you use a Unicode code page when creating the database because the text data you pass to the APIs must be compatible with the selected code page.
6. Start the Oracle listener by entering the command:

   ```
   lsnrctl start
   ```

7. On your application server machine, change to the directory where the configuration scripts for process choreographer are located: On Windows systems, enter: `cd install_root\ProcessChoreographer`.

   On UNIX systems, enter: `cd install_root/ProcessChoreographer`.
8. If your database server is on a different machine to your application server:,
   a. Copy the following Oracle configuration DDL scripts from the process choreographer subdirectory on the application server machine to an appropriate directory on your database machine:
      - clearSchemaOracle*X*.ddl
      - createSchemaOracle*X*.ddl
      - createTablespaceOracle*X*.ddl
      - dropSchemaOracle*X*.ddl
      - dropTablespaceOracle*X*.ddl

      where *X* is your Oracle version digit ('8' or '9').
   b. On your database machine, change to the directory where you copied the DDL scripts.
9. Edit the table space creation script according to the instructions at the top of the file:
   - For Oracle 8i: Edit `createTablespaceOracle8.ddl`
   - For Oracle 9i: Edit `createTablespaceOracle9.ddl`
10. Make sure that you are using the user ID that has administrator rights for the database system.
11. If you do not want to create the schema in the default instance, set the `ORACLE_SID` environment variable.
12. To create the table space, run the `createTablespaceOracle*X*.ddl` script, where '*X*' is your Oracle version digit (8 or 9). For test purposes, you can use the same location for all table spaces and pass the path as command line argument to the script, for example, on Windows systems, using Oracle 8i, user ID `bpeuser`, password `bpepwd`, database name `BPEDB`, and table space path `d:\mydb\ts`, enter:

    ```
    sqlplus bpeuser/bpepwd@BPEDB @createTablespaceOracle8.ddl d:\mydb\ts
    ```

    If you get any errors creating the table space, you can use the `dropTablespaceOracle*X*.ddl` file to drop the table space, where '*X*' is your Oracle version digit (8 or 9).
13. To create the schema, run the `createSchemaOracle*X*.ddl` script, where '*X*' is your Oracle version digit (8 or 9). For example, on Windows systems using Oracle 9i, enter:

    ```
    sqlplus bpeuser/bpepwd@BPEDB @createSchemaOracle9.ddl
    ```

    If you get any errors creating the schema (tables and views), you can use the `clearSchemaOracleX.ddl` file to clear the schema, and the `dropSchemaOracleX.ddl` file to drop the schema.

The Oracle database for process choreographer exists.

Creating a Sybase Adaptive Server Enterprise database for the process choreographer

This topic describes how to create a Sybase Adaptive Server Enterprise database for process choreographer.

1. Make sure that you have administrator rights on the machine where you want to install Sybase.
2. Install the Sybase Adaptive Server Enterprise server, with the Distributed Transaction Management (DTM) option, on the machine that hosts the database.
3. Copy and configure the JDBC driver to the following machines:
   - All remote application servers that use the database server.
   - On the deployment manager machine if you are using Network Deployment to administer process choreographer, for example if you are creating a clustered process choreographer setup.
4. Make sure that you configured and enabled the DTM option for Sybase Adaptive Server Enterprise:
   a. Set `enable DTM` to `1` in the Sybase server configuration.
   b. Set `enable xact coordination` to `1` in the Sybase server configuration.
   c. Add the `dtm_tm_role` role to the Sybase administration user ID, for example, user ID `sa`.
   d. Restart the Sybase server.
5. On your application server machine, change to the directory where the configuration scripts for process choreographer are located: On Windows systems, enter: `cd` *`install_root`*`\ProcessChoreographer`. On UNIX systems, enter: `cd` *`install_root`*`/ProcessChoreographer`.
6. If your database server is not on the same machine as your application server:
   a. Copy the DDL scripts for your database version from the `ProcessChroeographer` directory on your application server to a suitable directory on your database server machine:

For Sybase 12.0:	For Sybase 12.5:

`clearSchemaSybase120.ddl`
`createDatabaseSybase120.ddl`
`createSchemaSybase120.ddl`
`dropSchemaSybase120.ddl`
	`clearSchemaSybase125.ddl`
	`createDatabaseSybase125.ddl`
	`createSchemaSybase125.ddl`
	`dropSchemaSybase125.ddl`

   b. Change to the directory where you copied the DDL scripts.
7. If you want to create a non-production database, named `BPEDB`, for stand-alone development, evaluation, or demo purposes, you only need to enter the command:

   ```
   isql -S serverName -U userID -P password
        -i createDatabaseSybaseXXX.ddl
   ```

   where
   serverName
   The name of the Sybase server, that you defined with the `dsedit` tool.
   userID The user ID to use.
   password
   The password for *userID*.
   XXX The three digit number based on your Sybase version number: Either`120` for version 12.0, or `125` for version 12.5.
8. If you prefer to create your database manually:

a. Inspect the options used in the createDatabaseSybase*XXX*.ddl quick database creation script, and make sure that you include these options in the database you create.

b. Create the database. For example, named BPEDB.

c. Run the script to create the schema, by entering the command:

```
isql -S serverName -U userID -P password
     -D databaseName -i createSchemaSybaseXXX.ddl
```

where

serverName
The name of the Sybase server, that you defined with the dsedit tool.

userID The user ID to use.

password
The password for *userID*.

databaseName
is the name of the database.

XXX The three digit number based on your Sybase version number: Either120 for version 12.0, or 125 for version 12.5.

d. Make sure that the script output contains no errors. If there were any errors, you can use the dropSchemaSybase*XXX*.ddl file to drop the schema.

The Sybase Adaptive Server Enterprise database for process choreographer exists.

Granting permission to the JDBC driver on the deployment manager

Use this task when using Java 2 security in an ND environment to grant the required permissions to the JDBC driver.

If you are using Java 2 security in an ND environment, depending on the JDBC provider you might need to manually update the properties\server.policy file on the deployment manager:

1. No update to the server.policy file is needed if you are using one of the following JDBC drivers:
   - DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA) - because the JAR files are in the WebSphere lib directory.
   - WebSphere embedded ConnectJDBC driver for MS SQL Server (XA) - because the JAR files are in the WebSphere lib directory.
   - Cloudscape JDBC Provider (XA) - is not supported on ND.
2. Otherwise, on the deployment manager, edit the server.policy file, and add the template text for your JDBC driver. Make sure you always use forward slashes (’/’), and resolve any WebSphere variables in the template.

JDBC driver	Template

DB2 universal provider	`// DB2 Universal JDBC Driver Provider (XA)`
	`grant codeBase "file:/${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar" {`
	`  permission java.security.AllPermission;`
	`};`
	`grant codeBase "file:/${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar" {`
	`  permission java.security.AllPermission;`
	`};`
	`grant codeBase "file:/${DB2UNIVERSAL_JDBC_DRIVER_PATH}/`
	`  db2jcc_license_cisuz.jar" {`
	`  permission java.security.AllPermission;`
	`};`

JDBC driver	Template

DB2 CLI provider	`// DB2 Legacy CLI-based Type 2 JDBC Driver (XA)`
	`grant codeBase "file:/${DB2_JDBC_DRIVER_PATH}/db2java.zip" {`
	`  permission java.security.AllPermission;`
	`};`

DB2 iSeries native driver	`// DB2 UDB for iSeries (Native XA - V5R2 and later)`
	`grant codeBase "file:/${OS400_NATIVE_JDBC_DRIVER_PATH}/db2_classes.jar" {`
	`  permission java.security.AllPermission;`
	`};`

DB2 iSeries toolbox driver	`// DB2 UDB for iSeries (Native XA - V5R2 and later)`
	`grant codeBase "file:/${OS400_TOOLBOX_JDBC_DRIVER_PATH}/jt400.jar" {`
	`  permission java.security.AllPermission;`
	`};`

Informix	`// Informix JDBC Driver (XA)`
	`grant codeBase "file:/${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbc.jar" {`
	`  permission java.security.AllPermission;`
	`};`
	`grant codeBase "file:/${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbcx.jar" {`
	`  permission java.security.AllPermission;`
	`};`

Oracle	`// Oracle JDBC Driver (XA)`
	`grant codeBase "file:/${ORACLE_JDBC_DRIVER_PATH}/ojdbc14.jar" {`
	`  permission java.security.AllPermission;`
	`};`

Sybase	`// Sybase JDBC Driver (XA)`
	`grant codeBase "file:/${SYBASE_JDBC_DRIVER_PATH}/jconn2.jar" {`
	`  permission java.security.AllPermission;`
	`};`

Note: Any syntax errors in the `server.policy` file can cause the deployment manager to fail to start.

3. Restart the deployment manager.

   For example, if DB2 is installed on Windows in `c:\Program Files\IBM\SQLLIB` and the template looks like this:

   ```
   // DB2 Legacy CLI-based Type 2 JDBC Driver (XA)
   grant codeBase "file:/${DB2_JDBC_DRIVER_PATH}/db2java.zip" {
     permission java.security.AllPermission;
   };
   ```

   You must add the following to the `server.policy` file:

   ```
   // DB2 Legacy CLI-based Type 2 JDBC Driver (XA)
   grant codeBase "file:/c:/Program Files/IBM/SQLLIB/java/db2java.zip" {
     permission java.security.AllPermission;
   };
   ```

   Note: Even on Windows platforms, you must only use forward slashes ('/') as a separator in the path.

The JDBC driver will work in an ND environment with Java security turned on.

Creating the queue manager and queues for the business process container

If you are using WebSphere MQ as an external Java Message Service (JMS) provider, you must create the queue manager and queues.

1. If you are not creating a WebSphere cluster setup, perform the following actions:
   a. Make sure that your user ID has the authority to create WebSphere MQ queues.
   b. Create the queue manager and queues: On Windows systems, enter:

      ```
      cd install_root\ProcessChoreographer
      createQueues.bat queueManager
      ```

      On UNIX systems, enter:

      ```
      cd install_root/ProcessChoreographer
      createQueues.sh queueManager
      ```

      where:
      queueManager
      is the name of an existing queue manager, or the name to give to a new queue manager. If the named queue manager already exists, it is used to create the queues. If the queue manager does not exist, it is created and started before the default queues are created.
2. If you are creating a WebSphere cluster setup that uses a WebSphere MQ cluster, only perform Creating clustered queue managers and queues.
3. If you are creating a WebSphere cluster setup that uses a central queue manager, perform the following actions:
   a. Copy the create queues script file from the WebSphere machine to the machine that hosts the central queue manager: On Windows systems, copy the file:

      ```
      install_root\ProcessChoreographer\createQueues.bat
      ```

      On UNIX, copy the file:

      ```
      install_root/ProcessChoreographer/createQueues.sh
      ```
   b. On the machine that hosts the queue manager, make sure that WebSphere MQ is installed, and that your user ID has the authority to create WebSphere MQ queues.
   c. Create the queue manager and queues: On Windows systems, enter:

      ```
      cd install_root\ProcessChoreographer
      createQueues.bat queueManager
      ```

      On UNIX systems, enter:

      ```
      cd install_root/ProcessChoreographer
      createQueues.sh queueManager
      ```

      where:
      queueManager
      is the name to give to the new queue manager.
   d. Add a listener for the new queue manager by entering the command:

      ```
      runmqlsr -t tcp -p port -m queueManager
      ```

      Where *port* is the port on which it will listen.

e. On UNIX systems, add definitions for the port and queue manager service:
   1) Add the port for the queue manager to the /etc/services file:

```
<Service:Name> <port>/tcp
  <Service:Name>    name of the queue manager service
  <port>            port for the queue manager
```

   2) Add the service specified in the /etc/services file to the /etc/inetd.conf file:

```
<Service:Name> stream tcp nowait mqm /usr/mqm/bin/amqcrsta amqrsta
               -m QueueManager
  <Service:Name>    name of the queue manager service
  <Service:Name>    name of the queue manager
```

The queue manager and queues exist.

Creating clustered queue managers and queues for the business process container

If you are creating a WebSphere cluster setup of process choreographer using a WebSphere MQ cluster, you must create the queue managers, queues, cluster, repositories, channels, and listeners.

1. If your WebSphere cluster consists of UNIX nodes, perform the following actions on each node:
   a. Make sure that your user ID has the authority to create WebSphere MQ queues.
   b. Create the "get" and "put" queue managers, make them members of the WebSphere MQ cluster, and create the queues by entering the commands:

      ```
      cd install_root/ProcessChoreographer
      createQueues.sh getQueueManager clusterName putQueueManagerName
      ```

      where:
      getQueueManager
      The unique name to give to the "get" queue manager. This queue manager will host all the local queues.
      clusterName
      The name of the WebSphere MQ cluster.
      putQueueManager
      The unique name to give to the "put" queue manager.
      If the queue managers already exist they are used. If the queue managers do not exist they are created and used.
   c. Start the WebSphere MQ command processor by entering the command:

      ```
      runmqsc getQueueManager
      ```

   d. For complex setups, it is recommended to enable remote administration of the queue manager by entering the following MQ command:

      ```
      DEFINE CHANNEL('SYSTEM.ADMIN.SVRCONN') TYPE(CHLTYPE)
      ```

   e. If this queue manager is to be a repository for the WebSphere MQ cluster enter the MQ command:

      ```
      ALTER QMGR REPOS('clusterName') REPOSNL(' ')
      ```

f. Define a sender and a receiver channel for the queue manager to each repository that is not hosted on this machine, by entering the following MQ commands. For each cluster receiver channel:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSRCVR) +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     CONNAME('repositoryIP-Address(port)') +
     DESCR('Cluster receiver channel at repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('principal') +
     REPLACE
```

For each cluster sender channel:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSSDR) +
     CONNAME('repositoryIP-Address(port)') +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     DESCR('Cluster sender channel to repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('targetPrincipal') +
     REPLACE +
     NPMSPEED (NORMAL)
```

where:

repositoryQueueManager
The name of the queue manager hosting a repository.

clusterName
The name of the WebSphere MQ cluster of which all the queue managers are a member.

repositoryIP-Address
The IP address of the node where the repository queue manager resides.

port The IP port that the repository queue manager is using.

principal, *targetPrincipal*
The MCAUSER to use for the receive and send channels. For more information about this value refer to the WebSphere MQ documentation.

g. For each queue manager, start a listener by entering the MQ command:

```
runmqlsr -t tcp -p port -m QueueManager
```

2. If your WebSphere cluster consists of Windows nodes, perform the following actions on each node:
   a. Make sure that your user ID has the authority to create WebSphere MQ queues.
   b. Create the "get" queue manager, make it a member of the WebSphere MQ cluster, and create the queues by entering the commands:

```
cd install_root\ProcessChoreographer
createQueues.bat queueManager clusterName putQueueManager
```

where:

getQueueManager
The unique name to give to the "get" queue manager.

clusterName
The name of the WebSphere MQ cluster that the WebSphere cluster nodes will use.

putQueueManager
The unique name to give to the "put" queue manager.

If the queues already exist they are used. If the queues do not exist, they are created and used.

c. Start the WebSphere MQ command processor by entering the command:

```
runmqsc queueManager
```

d. For complex setups, it is recommended that you enable remote administration of the queue manager by entering the following MQ command:

```
DEFINE CHANNEL('SYSTEM.ADMIN.SVRCONN') TYPE(CHLTYPE)
```

e. If this queue manager is to be a repository for the WebSphere MQ cluster enter the MQ command:

```
ALTER QMGR REPOS('clusterName') REPOSNL(' ')
```

f. Define a sender and a receiver channel for the queue manager to each repository that is not hosted on this machine, by entering the following MQ commands. For each cluster receiver channel:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSRCVR) +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     CONNAME('repositoryIP-Address(port)') +
     DESCR('Cluster receiver channel at repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('principal') +
     REPLACE
```

For each cluster sender channel:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSSDR) +
     CONNAME('repositoryIP-Address(port)') +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     DESCR('Cluster sender channel to repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('principal') +
     REPLACE +
     NPMSPEED (NORMAL)
```

where:

repositoryQueueManager
The name of the queue manager hosting a repository.

clusterName
The name of the WebSphere MQ cluster to which all the queue managers are a member.

repositoryIP-Address
The IP address of the node where the repository queue manager resides.

port The IP port that the repository queue manager is using.

principal
The MCAUSER to use. For more information about this value, refer to the WebSphere MQ documentation.

g. For each queue manager, start a listener by entering the MQ command:

```
runmqlsr -t tcp -p port -m QueueManager
```

3. To verify the status of the channels on a machine, enter the MQ command:

   ```
   display chstatus(*)
   ```

The queue managers, queues, cluster, repositories, channels, and listeners exist.

Configuring the business process container on a cluster

If you are preparing a clustered process choreographer setup, you can install and configure the business process container on a WebSphere cluster, but you must also customize the connection factories separately for each application server in the cluster.

1. Use the Process choreographer Install Wizard on the deployment manager machine, install and configure the business process container on any application server in the cluster. This action causes the business process container to be installed and configured on all the other application servers in the cluster.
2. You must modify the queue connection factory definitions one-by-one. For each application server in the WebSphere cluster:
   a. Click **Resources > WebSphere MQ JMS Providers**.
   b. Click **WebSphere MQ Queue Connection Factory**.
   c. Select the connection factory `BPECF` and set the property values for the type of queue manager configuration you are using:
      - For a central queue manager:

Host	The host name of the machine that is hosting the central queue manager.
Port	The port number that the central queue manager is using.
Transport Type	`Client`
Client ID	The MCA user ID to use.
CCSID	On UNIX systems: 819. On Windows systems: 437

      - For a cluster of queue managers:

Host	The host name of the application server's node.
Transport Type	`Binding`

   d. Select the connection factory `BPECFC` and set the property values for the type of queue manager configuration you are using:
      - For a central queue manager:

Host	The host name of the machine that is hosting the central queue manager.
Port	The port number that the central queue manager is using.
Transport Type	`Client`
Client ID	The MCA user ID to use.
CCSID	On UNIX: 819. On Windows: 437

      - For a cluster of queue managers on UNIX:

Host	The host name of the application server node.

Port	The port number used by the ″put″ queue manager of this application server's .
Transport Type	`Client`
Client ID	The MCA user ID to use.
CCSID	On UNIX systems: 819.

- For a cluster of queue managers on Windows systems:

Host	The host name of the application server node.
Transport Type	`Binding`

The business process containers have been installed in the cluster and are configured.

Now you can activate the business process containers.

Using the Install Wizard to configure the business process container

You must configure the necessary resources and install the business process container before you can use it. This topic describes how to do so using the Install Wizard.

1. Make sure that you are logged on to the administrative console with a user ID with sufficient administration rights. On Windows, use the user ID that will be used to start WebSphere.
2. Select **Servers > Application Servers > *serverName*** Where *serverName* is the name of the application server where you want to install the business process container. In a cluster, you can select any application server, and the business process container will be installed simultaneously on all application servers in the cluster.

   Note: When installed on a non-clustered application server, the name of the business process container will be `BPEContainer_nodeName_serverName`, on a cluster it will be named `BPEContainer_clusterName`.
3. In the **Additional Properties** section, click **Business Process Container**.
4. Scroll down, past the Business Process Container settings. Near the bottom of the page click on the link for the **Business Process Container Install Wizard**.

   Note: Where possible, the Install Wizard offers appropriate default values in the parameter fields. However, with some combinations of browser and platform no defaults are provided. In this case, you can view the recommended values on the Install Wizard settings page.
5. Select the database configuration:
   a. Select **JDBC Providers**.
   b. In the drop-down list, select the entry with the database system, system version and JDBC driver that you are using.
   c. For the **Implementation class name** use the default class name provided for the JDBC driver implementation.
   d. For **Classpath** enter the location of the Java archive (JAR) or zip file of the JDBC driver. To use the path variable that is displayed in the text field, you must set it explicitly in **Environment> Manage WebSphere Variables**.

e. The **Data source user name** must be a user ID that has the authority to connect to the database and to manipulate the data. If you want to have the database schema updated automatically, the user ID must also have the authority to create tables and indexes in the database.

f. Enter the **Data source password** for the Data source user name.

g. Make sure that the options in the **Custom properties** field match your database requirements. For more information, see the Install Wizard settings page and the product documentation for your database system.

h. Click **Next** to go to the next step in the Install Wizard.

6. Select the JMS provider and security configuration:

a. In the drop-down list for **JMS provider**, select the messaging service that the business process container will use.

b. For the **Queue manager**, use the default provided (`WAS_nodeName_serverName`).

c. If you are using external messaging (WebSphere MQ) and you have not defined the WebSphere environment variable `${MQ_INSTALL_ROOT}`, make sure that the **Classpath** points to the MQ Java lib directory.

d. For the **JMS user ID**, enter a user ID that has administration rights for the messaging service. On UNIX, use `root`. On Windows, use the default.

e. For the **JMS password**, enter the password for the JMS user ID.

f. For the **Scheduler calendar** field, if you have your own scheduler calendar, enter its JNDI name. Otherwise, if you leave it blank, the default value, `com/ibm/Websphere/scheduler/calendar/DefaultUserCalendar`, will be used.

g. For the **Security role mapping**, enter the user or group from your user registry that is to be mapped onto the role of Business Process Administrator.

h. For the **JMS API user ID**, enter the user ID that is to be used when processing asynchronous API calls.

i. For the **JMS API password**, enter the password for the JMS API User ID.

j. Click **Next** to go to the next step in the Install Wizard.

7. To select the JMS resources, either select **Create new JMS resources using default values** and click **Next**, or perform the following:

a. Select **Select existing JMS resources**.

b. Use the **Connection Factory** drop-down list to select `BPECF`.

c. Use the **Internal queue** drop-down list to select `BPEIntQueue`.

d. Use the **External request processing queue** drop-down list to select `BPEApiQueue`.

e. Use the **Hold queue** drop-down list to select `BPEHldQueue`.

f. Use the **Retention Queue** drop-down list to select `BPERetQueue`.

8. 5.1 + Select **Web client** if you want to install the Web client (recommended), otherwise clear the check box.

9. Click **Next** to view the summary page.

10. Check that the information on the summary page is correct.

Note: The summary includes reminders of which external resources are necessary. If you have not already created them, you can continue configuring the business process container, but you must create the resources before you activate the business process container. Printing the summary page will help you to create the correct resources.

a. To make corrections, click **Previous**.
b. To install the business process container and define its resources click **Finish**.

11. The progress is shown on the **Installing** page:
    a. If the container did not install successfully, check for any error messages that can help you correct the problem, then repeat this task from step 1.
    b. If the container was installed successfully, click **Save Master Configuration**, then click **Save**.

Business process container Install Wizard settings

Use the Install Wizard to install and configure the business process container.

This page describes the Install Wizard fields, in the order that they appear in the Install Wizard.

Step 1 database configuration:

- XA data source: JDBC providers
- Implementation class name
- Classpath (for data source)
- Data source user name
- Data source password
- Custom properties

Step 2 JMS provider and security:

- JMS provider
- Queue manager
- Classpath (for JMS provider)
- JMS user ID
- JMS password
- Scheduler calendar
- Security role mapping
- JMS API user ID
- JMS API password

Step 3 JMS resources and Web client:

- JMS resources (new or existing)
- Connection factory
- Internal queue
- External request processing queue
- Hold queue
- Retention queue
- Web client

Attention: You cannot change these fields after they have been applied. If you want to change any of these values after the container is configured, you must uninstall and reinstall the `BPEContainer.ear` enterprise application file. This action can result in the loss of data such as pending messages in queues, process templates, and process instances in the process choreographer database.

XA data source

You must create a new data source that will only be used by process choreographer.

Mandatory	Yes
Data type	Drop-down list
Choices	Create a new XA data source: • Cloudscape 5.1 (Cloudscape JDBC Provider (XA)) • DB2 UDB 8.1 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA)) • DB2 UDB 8.1 (DB2 Universal JDBC Driver Provider (XA)) • DB2 z/OS 7 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA)) • DB2 zOS 7 (DB2 Universal JDBC Driver Provider (XA)) • DB2 UDB for iSeries (Native XA V5R2 and later) • DB2 UDB for iSeries (Toolbox XA) • Informix 9.3 & 9.4 (Informix JDBC Driver (XA)) • Oracle 8i OCI (Oracle JDBC Driver (XA)) • Oracle 8i thin (Oracle JDBC Driver (XA)) • Oracle 9i OCI (Oracle JDBC Driver (XA)) • Oracle 9i thin (Oracle JDBC Driver (XA)) • SQL Server 2000 (DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA)) • SQL Server 2000 (WebSphere embedded ConnectJDBC driver for MS SQL Server (XA)) • Sybase 12.0 (Sybase JDBC Driver (XA)) • Sybase 12.5 (Sybase JDBC Driver (XA))

Implementation class name

The Java class name of the Java Database Connectivity (JDBC) driver implementation.

Mandatory	Yes
Data type	String
Default for Cloudscape 5.1 (Cloudscape JDBC Provider (XA))	`com.ibm.db2j.jdbc.DB2jXADataSource`
Default for DB2 UDB 8.1 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA)) and for DB2 z/OS 7 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA))	`COM.ibm.db2.jdbc.DB2XADataSource`
Default for DB2 UDB 8.1 (DB2 Universal JDBC Driver Provider (XA)) and for DB2 z/OS 7 (DB2 Universal JDBC Driver Provider (XA))	`com.ibm.db2.jcc.DB2XADataSource`
Default for DB2 UDB for iSeries (Native XA V5R2 and later)	`com.ibm.db2.jdbc.app.UDBXADataSource`
Default for DB2 UDB for iSeries (Toolbox XA)	`com.ibm.as400.access.AS400JDBCXADataSource`
Default for Informix 9.3 & 9.4 (Informix JDBC Driver (XA))	`com.informix.jdbcx.IfxXADataSource`
Default for Oracle 8i OCI, 8i thin, 9i OCI, and 9i thin (Oracle JDBC Drivers (XA))	`oracle.jdbc.xa.client.OracleXADataSource`
Default for SQL Server 2000 (DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA))	`com.ddtek.jdbcx.sequelink.SequeLinkDataSource`

Default for SQL Server 2000 (WebSphere embedded ConnectJDBC driver for MS SQL Server (XA))
`com.ibm.websphere.jdbcx.sqlserver.SQLServerDataSource`

Default for Sybase 12.0 (Sybase JDBC Driver (XA)) and for Sybase 12.5 (Sybase JDBC Driver (XA))
`com.sybase.jdbc2.jdbc.SybXADataSource`

Classpath (data source)

The path to the Java archive (jar) or zip file that contains the JDBC driver. The JDBC driver provides the data source implementation class. If the database is remote, this is the path where the JDBC driver is installed on the client machine.

Mandatory

For Cloudscape
No, the JDBC driver is already on the WebSphere classpath.

For DB2 UDB, DB2 z/OS, Informix, Oracle, SQL Server, and Sybase
Yes

Data type
String

Default for Cloudscape 5.1
`${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar`

The value for `${CLOUDSCAPE_JDBC_DRIVER_PATH}` is predefined and does not need to be set.

Default for DB2 UDB 8.1 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA)) and for DB2 z/OS 7 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA))
`${DB2_JDBC_DRIVER_PATH}/db2java.zip`

The value for `${DB2_JDBC_DRIVER_PATH}` depends on the DB2 Client installation directory and must be set explicitly in **Environment > Manage WebSphere Variables**. Typical values are:

On Windows:
`c:\Program Files\SQLLIB\java`

On AIX and HP-UX:
`/home/db2inst1/sqllib/java`

On Solaris:
`/export/home/db2inst1/sqllib/java`

Default for DB2 UDB 8.1 (DB2 Universal JDBC Driver Provider (XA)) and for DB2 z/OS 7 (DB2 Universal JDBC Driver Provider (XA))

```
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/
              db2jcc.jar
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/
              db2jcc_license_cu.jar
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/
              db2jcc_license_cisuz.jar
```

The value for `${DB2UNIVERSAL_JDBC_DRIVER_PATH}` depends on the installation root directory of the corresponding DB2 Client or DB2 Connect, and must be set in **Environment> Manage WebSphere Variables**. Typical values for `${DB2UNIVERSAL_JDBC_DRIVER_PATH}` are:

On Windows:
`c:\Program Files\SQLLIB\java`

On AIX and HP-UX:
`/home/db2inst1/sqllib/java`

On Solaris:
`/export/home/db2inst1/sqllib/java`

Default for Informix 9.3 and 9.4

```
${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbc.jar
${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbcx.jar
```

The value for `${INFORMIX_JDBC_DRIVER_PATH}` depends on the JDBC driver installation directory and must be set in **Environment> Manage WebSphere Variables**.

Default for Oracle 8i and Oracle 9i

```
${ORACLE_JDBC_DRIVER_PATH}/ojdbc14.jar
```

The value for `${ORACLE_JDBC_DRIVER_PATH}` depends on the Oracle client installation or JDBC driver installation directory and must be set in **Environment> Manage WebSphere Variables**.

Default for SQL Server 2000 (DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA))

```
${WAS_LIBS_DIR}/sljc.jar
${WAS_LIBS_DIR}/spy-sl53.jar
```

The value for `${WAS_LIBS_DIR}` depends on the WebSphere installation directory and must be set in **Environment> Manage WebSphere Variables**. Typical values are:

On Windows:

```
C:\Program Files\WebSphere\AppServer\lib
```

On UNIX:

```
/opt/WebSphere/AppServer/lib
```

Default for SQL Server 2000 (WebSphere embedded ConnectJDBC driver for MS SQL Server (XA))

```
${WAS_LIBS_DIR}/sqlserver.jar
${WAS_LIBS_DIR}/base.jar
${WAS_LIBS_DIR}/util.jar
${WAS_LIBS_DIR}/spy.jar
```

The value for `${WAS_LIBS_DIR}` depends on the WebSphere installation directory and must be set in **Environment> Manage WebSphere Variables**. Typical values are:

On Windows:

```
C:\Program Files\WebSphere\AppServer\lib
```

On UNIX:

```
/opt/WebSphere/AppServer/lib
```

Default for Sybase 12.0 and Sybase 12.5

```
${SYBASE_JDBC_DRIVER_PATH}/jconn2.jar
```

The value for `${SYBASE_JDBC_DRIVER_PATH}` depends on the Sybase JDBC driver installation directory and must be set in **Environment> Manage WebSphere Variables**. Typical values are:

On Windows:

```
C:\Program Files\Sybase\jConnect-5_5\classes
```

On UNIX:

```
/opt/sybase/jConnect-5_5/classes
```

Data source user name

A user ID that has the authority to connect to the database and to manipulate the data. If you want to have an automatic update of the database schema the user ID must have the authority to create tables and indexes in the database.

Mandatory
For Cloudscape
No
For DB2 UDB, DB2 iSeries, DB2 z/OS, Informix, Oracle, SQL Server, and Sybase
Yes

Data type
String

Default
The user ID that is currently logged on to the administrative console.

Data source password

The password for the data source user ID.

Mandatory
For Cloudscape
No
For DB2 UDB, DB2 z/OS, Informix, Oracle, SQL Server, and Sybase
Yes

Data type
String

Default
None

Custom properties

Extra parameters that are required by the database system.

Mandatory
Yes

Data type
String

Data format
Multiple lines of *Property=Value*

Properties for Cloudscape

databaseName=
install_root/ProcessChoreographer/BPEDB. Required string. Defines which database to access. The value must be a fully qualified path.

shutdownDatabase=
Optional string. If set to `shutdown`, the database shuts down when a java.sql.Connection object is obtained from the data source. For example, if the data source is an XADataSource, a `getXAConnection().getConnection()` is necessary to cause the database to shut down.

dataSourceName=
Optional string. Name for the ConnectionPooledDataSource or XADataSource. Not used by the data source object. Used for information purposes only.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Not used by the data source object. Used for information purposes only.

connectionAttribute=
Optional string. Connection attributes specific to Cloudscape. Refer to the Cloudscape documentation for a complete list of attributes.

createDatabase=
Optional string. If set to `create`, and the database specified in the databaseName parameter does not already exist, the database will be created. The database is created when a connection object is obtained from the data source.

enableMultithreadedAccessDetection=false
Optional boolean. If set to `true`, it automatically detects multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. This SQL statement is used for pre-test connection function. If pre-test connection is enabled in j2c.properties, this SQL statement will be executed to the connection to make sure the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

remoteDataSourceProtocol=
Optional string. If the database is remote and the data source accesses the database using a client, set this property to specify which client and server protocol to use. Currently, the only protocol value supported is `rmi`.

Properties for DB2 UDB 8.1 & DB2 z/OS 7 (DB2 Legacy CLI-based Type 2 JDBC Driver (XA))

databaseName=BPEDB
Required string. For DB2 UDB it defines which database to access. For DB2 z/OS it defines which subsystem contains the DB2 z/OS database.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Not used by the data source object. Used for information purposes only.

portNumber=
Optional integer. Specifies the TCP/IP port number where the JDBC provider resides.

connectionAttribute=cursorhold=0
Optional string. Connection attributes specific to DB2. Refer to the DB2 documentation for a complete list of connection attributes.

loginTimeout=0
Optional integer. If set to zero, no timeout takes place. Non-zero values specify the maximum number of seconds allowed to establish a connection to the database.

enableMultithreadedAccessDetection=false
Optional boolean. If set to `true`, it automatically detects multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. If you enabled pre-test connection in the `j2c.properties` file, this pre-test string is used to make sure that the connection is good. If you leave this field blank, the default SQL statement, `SELECT 1 FROM TABLE1`, will be used at runtime. This will reduce performance because of the exception handling if table TABLE1 is not defined in the database. It is recommended to provide your own SQL statement to improve the performance.

Properties for DB2 UDB 8.1 & DB2 z/OS 7 (DB2 Universal JDBC Driver Provider (XA))

databaseName=BPEDB
Required string. For DB2 UDB it defindes which database to access. For DB2 z/OS it defines which subsystem contains the DB2 z/OS database.

driverType=2
Required integer. The JDBC connectivity-type of a data source. The only permitted value is 2.

serverName=""
Optional string. The TCP/IP address or host name for the DRDA server.

portNumber=50000
Optional integer. The TCP/IP port number where the DRDA server resides.

enableSQLJ=false
Optional boolean. This value is used to indicate whether SQLJ operations may be performed with this data source. If enabled, this data source can be used for both JDBC and SQLJ calls. Otherwise, only JDBC usage is permitted.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Not used by the data source object. Used for information purposes only.

traceLevel=""
Optional integer. The DB2 trace level for logging to the logWriter or trace file. Possible trace levels are: TRACE_NONE = 0,TRACE_CONNECTION_CALLS = 1,TRACE_STATEMENT_CALLS = 2,TRACE_RESULT_SET_CALLS = 4,TRACE_DRIVER_CONFIGURATION = 16,TRACE_CONNECTS = 32,TRACE_DRDA_FLOWS = 64,TRACE_RESULT_SET_META_DATA = 128,TRACE_PARAMETER_META_DATA = 256,TRACE_DIAGNOSTICS = 512,TRACE_SQLJ = 1024,TRACE_ALL = -1, .

traceFile=""
Optional string. The trace file to store the trace output.

fullyMaterializeLobData=true
Optional boolean. This setting controls whether or not LOB locators are used to fetch LOB data. If enabled, LOB data is not streamed, but is fully materialized with locators when the user requests a stream on the LOB column. The default value is true.

resultSetHoldability=2
Optional integer. Determine whether ResultSets are closed or kept open when committing a transaction. The possible values are: 1 (HOLD_CURSORS_OVER_COMMIT), 2 (CLOSE_CURSORS_AT_COMMIT).

Properties for DB2 UDB 8.1 & DB2 z/OS 7 (DB2 Universal JDBC Driver Provider (XA)) (continued)

currentPackageSet=""
Optional string. This property is used in conjunction with the DB2Binder - collection option which is given when the JDBC/CLI packageset is bound during installation by the DBA.

readOnly=false
Optional boolean. This property creates a read only connection.

deferPrepares=false
Optional boolean. This property provides a performance directive that affects the internal semantics of the input data type conversion capability of the driver. If it is set to "true" the Universal driver defers 'internal prepare requests'. In this case, the driver works without the benefit of described parameter or result set metadata until execute time. So undescsribed input data is sent 'as is' to the server without any data type cross-conversation of the inputs.

currentSchema=""
Optional string. Identifies the default schema name used to qualify unqualified database object references where applicable in dynamically prepared SQL statements. Unless currentSchema is used, the default schema name is the authorization id of the current session user.

cliSchema=""
Optional string. Indicates the schema of the DB2 shadow catalog tables or views to search when you issue a database metadata catalog query.

enableMultithreadedAccessDetection=false
Optional boolean. If set to `true`, it automatically detects multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

retrieveMessagesFromServerOnGetMessage=true
If set to true it directs all calls to the standard JDBC SQLException.getMessage() to invoke a server-side stored procedure which retrieves the readable message text for the error.

Properties for DB2 UDB 8.1 & DB2 z/OS 7 (DB2 Universal JDBC Driver Provider (XA)) (continued)

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. This SQL statement is used for pre-test connection function. If pre-test connection is enabled in j2c.properties, this SQL statement will be executed to the connection to make sure the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

enableSQLJ=false
Optional boolean. This SQL statement is used for pre-test connection function. If pre-test connection is enabled in j2c.properties, this SQL statement will be executed to the connection to make sure the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

Properties for Properties for Informix 9.3 & 9.4

databaseName=BPEDB
Required string. The name of the database.

serverName=""
Required string. The name of the Informix instance on the physical machine. There is no default for the string and therefore you need to set it explicitly.

portNumber=1526
Required integer. The TCP/IP port number of the Informix instance. The value "1526" is the default Informix port on Windows. On UNIX it is "61000".

ifxIFXHOST=*hostname*
Required string. The physical machine name of the server that hosts the informix database. "Localhost" does not work.

informixLockModeWait=2
Required integer. By default, Informix throws an exception when it cannot acquire a lock, rather than waiting for the current owner of the lock to release it. To modify this behavior, set this property to the number of seconds to wait for a lock. The default is 2 seconds. Any negative value means to wait forever.

informixAllowNewLine=false
Optional boolean. his property allows newLines to be added on a query String. Its disabled by default.

roleName=""
Optional string. The role name.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Not used by the data source object. Only used for information purposes.

loginTimeout=""
Optional integer. The maximum time to attempt to connect a database. If this value is non-zero, attempt to connect to the database will timeout when this specified value is reached.

dataSourceName=""
Optional string. The name of the data source. Only used for informational purposes.

ifxUSE_DTENV to ifxPSORT_DBTEMP
Optional string. The value for Informix specific variables. Please refer to the Informix documentation for possible values.

Properties for Properties for Informix 9.3 & 9.4 (continued)

enableMultithreadedAccessDetection=false
Optional boolean. Indicates whether or not to detect multithreaded access to a Connection and its corresponding Statements, ResultSets, and MetaDatas.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. This SQL statement is used for pre-test connection function. If pre-test connection is enabled in j2c.properties, this SQL statement will be executed to the connection to make sure the connection is good. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

Properties for Oracle 8i OCI & Oracle 9i OCI

driverType=oci8
Required string. Defines the type of the JDBC driver.

oraclelogPrintMask=62
Optional integer. The oraclelogPrintMask controls which information is printed with each trace message. Oracle9i requires the `ojdbc14_g.jar` file. The default is 62, which is ([OracleLog.FIELD_OBJECT for 9i / OracleLog.FIELD_CONN for 8i] 32 | OracleLog.FIELD_CATEGORY 16 | OracleLog.FIELD_SUBMOD 8 | OracleLog.FIELD_MODULE 4 | OracleLog.FIELD_TIME 2).

Possible values: OracleLog.FIELD_TIME 2, OracleLog.FIELD_MODULE 4, OracleLog.FIELD_SUBMOD 8, OracleLog.FIELD_CATEGORY 16, OracleLog.FIELD_OBJECT 32, OracleLog.FIELD_THREAD 64.

oraclelogModuleMask=1
Optional integer. The oraclelogModuleMask controls which modules write debug output. Oracle9i requires the `ojdbc14_g.jar` file. The default is 1, which is (OracleLog.MODULE_DRIVER 1).

Possible values: OracleLog.MODULE_DRIVER 1, OracleLog.MODULE_DBACCESS 2.

oraclelogCategoryMask=47
Optional integer. The oraclelogCategoryMask controls which category to be output. Oracle9i requires the `ojdbc14_g.jar` file. The default is 47, which is (OracleLog.USER_OPER 1 | OracleLog.PROG_ERROR 2 | OracleLog.ERROR 4 | OracleLog.WARNING 8 | OracleLog.DEBUG1 32).

Possible values: OracleLog.USER_OPER 1, OracleLog.PROG_ERROR 2, OracleLog.ERROR 4, OracleLog.WARNING 8, OracleLog.FUNCTION 16, OracleLog.DEBUG1 32, OracleLog.SQL_STR 128.

TNSEntryName=BPEDB
Required string. The entry name that is used in tnsnames.ora to identify the database.

networkProtocol=
Optional string. Specifies which protocol is used, for example, `TCP/IP` or `IPC`.

databaseName=BPEDB
Optional string. Defines which database to access.

serverName=
Optional string. The name of the server where the Oracle database resides.

portNumber=1521
Optional integer. The TCP/IP port number where the JDBC driver resides.

Properties for Oracle 8i OCI & Oracle 9i OCI (continued)

dataSourceName=
Optional string. The name of the datasource. Only used for informational purposes.

URL=jdbc:oracle:oci8:@BPEDB
Optional string. The URL specifies the database from which the data source obtains connections.

loginTimeout=""
Optional integer. The maximum time to attempt to connect the database. If set to zero, there will be no timeout. If this value is non-zero, attempt to connect to the database will timeout when this specified value is reached.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Used for information purposes only.

enableMultithreadedAccessDetection=false
Optional boolean. If set to `true`, it automatically detects multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

transactionBranchesLooselyCoupled=false
Optional boolean. This property is introduced as a result of Oraclebug 2511780. Oracle Patch for 2511780 must be installed before setting this property to `true`. Failure to install this patch causes a program error. Check the WebSphere readme file for more information.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. If you have enabled a pre-test connection in the `j2c.properties` file, this pre-test string is used to make sure that the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

Properties for Oracle 8i thin & Oracle 9i thin

driverType=thin
Required string. Defines the type of the JDBC driver.

oraclelogPrintMask=62
Optional integer. The oraclelogPrintMask controls which information is printed with each trace message. Oracle9i requires the use of ojdbc14_g.jar. Default is 62 which is ([OracleLog.FIELD_OBJECT for 9i / OracleLog.FIELD_CONN for 8i] 32 | OracleLog.FIELD_CATEGORY 16 | OracleLog.FIELD_SUBMOD 8 | OracleLog.FIELD_MODULE 4 | OracleLog.FIELD_TIME 2). Possible values: OracleLog.FIELD_TIME 2, OracleLog.FIELD_MODULE 4, OracleLog.FIELD_SUBMOD 8, OracleLog.FIELD_CATEGORY 16, OracleLog.FIELD_OBJECT 32, OracleLog.FIELD_THREAD 64.

oraclelogModuleMask=1
Optional integer. The oraclelogModuleMask controls which modules write debug output. Oracle9i requires the use of ojdbc14_g.jar. Default is 1 which is (OracleLog.MODULE_DRIVER 1). Possible values (OracleLog.MODULE_DRIVER 1, OracleLog.MODULE_DBACCESS 2)″

oraclelogCategoryMask=47
Optional integer. The oraclelogCategoryMask controls which category to be output. Oracle9i requires the use of ojdbc14_g.jar. Default is 47 which is (OracleLog.USER_OPER 1 | OracleLog.PROG_ERROR 2 | OracleLog.ERROR 4 | OracleLog.WARNING 8 | OracleLog.DEBUG1 32). Possible values (OracleLog.USER_OPER 1, OracleLog.PROG_ERROR 2, OracleLog.ERROR 4, OracleLog.WARNING 8, OracleLog.FUNCTION 16, OracleLog.DEBUG1 32, OracleLog.SQL_STR 128)″

TNSEntryName=″″
Optional string. Not used for the thin JDBC driver.

networkProtocol=″″
Optional string. Specifies which protocol is used, for example, TCP/IP or IPC.

databaseName=BPEDB
Required string. Defines which database to access.

serverName=*hostname*
Required string. The name of the server where the Oracle database resides

portNumber=1521
Required integer. The TCP/IP port number where the JDBC driver resides.

Properties for Oracle 8i thin & Oracle 9i thin (continued)

dataSourceName=""
Optional string. The name of the data source. Only used for informational purposes.

URL=jdbc:oracle:thin:@*hostname*:1521:BPEDB
Required string. The URL specifies the database and the database host from which the data source will obtain connections.

loginTimeout=""
Optional integer. The maximum time to attempt to connect the database. If set to zero, there will be no timeout. If this value is non-zero, attempt to connect to the database will timeout when this specified value is reached.

description=DataSource for Process Choreographer
Optional string. Description of the datasource. Only used for information purposes.

enableMultithreadedAccessDetection=false
Optional boolean. If set to true, it will automatically detect multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

transactionBranchesLooselyCoupled=false
Optional boolean. This property is introduced as a result of Oraclebug 2511780, Oracle Patch for 2511780 must be installed before setting this property to true, failure to do that would cause a program error. Please check the WebSphere readme file for more info.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. If you have enabled pre-test connection in j2c.properties, this pre-test string is used to make sure that the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

Properties for SQL Server 2000 (DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA))

databaseName=BPEDB
Required string. The database name.

serverName=*hostname*
Required string. The TCP/IP address of the SequeLink server in dotted format or host name format.

portNumber=19996
Required integer. The TCP/IP port number where the jdbc driver resides.

spyAttributes=""
Optional string. The SPY attributes. See the ConnectJDBC documentation for a list of attributes.

loginTimeout=""
Optional integer. The maximum time to attempt to connect a database. If this value is non-zero, attempt to connect to the database will timeout when this specified value is reached.

description=DataSource for Process Choreographer
Optional string. The description of this datasource.

cipherSuites=""
Optional string. The Secure Socket Layer(SSL) cipher suites with which the SequeLink Java Client can use to connect. This property is required when networkProtocol=ssl.

blockFetchForUpdatee=1
Optional integer. When the isolation level is ReadCommitted and a SELECT FOR UPDATE statement is issued against some data stores, the SequeLink Java Client does not lock the expected row. Default is 1 which does not lock the expected row.

SLKStaticCursorLongColBuffLen=4
Optional integer. The amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns which an insensitive result set.

certificateChecker=com.merant.sequelink.cert.CertificateCheckerInterface
Optional string. The fully qualified class name of a user-defined server certificate checker class.

serverDataSource=""
Optional string. A property that specifies a string to identify the server data source to be used for the connection. If unspecified, the configuration of the default server data source will be used for the connection.

enable2Phase=true
Optional boolean. When true, two phase connections are used. Do not change this value because this value is set for with this provider and is required by Process Choreographer.

applicationName=""
Optional string. Identifies the application that is establishing the Connections. When the application does not provide a value, the default value is SequeLink JDBC Application.

Properties for SQL Server 2000 (DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA)) (continued)

MSSMapLongToDecimal=""
Optional boolean. Enable client-side workarounds. Refer to the SequeLink Administrator's Guide.

ORANumber0IsNumeric=""
Optional boolean. Enable client-side work-arounds. Refer to the SequeLink Administrator's Guide.

enableMultithreadedAccessDetection=false
Optional boolean. Indicates whether or not to detect multithreaded access to a Connection and its corresponding Statements, ResultSets, and MetaDatas.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. This SQL statement is used for pre-test connection function. If pre-test connection is enabled in j2c.properties, this SQL statement will be executed to the connection to make sure the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

allowPrefetch=0
Optional integer. Enables the prefetch feature. When enabled, the JDBC driver requests a next set of rows from the server while the client is processing the previous set of rows. 0 = disable; 1 = enable.

insensitiveResultSetBufferSize=2058
Optional integer. The memory caching scheme for scroll-insensitive cursors. When set to 0, the driver uses a memory caching mechanism that does not use disk overflow. When set to a value greater than 0, data overflows to disk when the size of cached data exceeds the specified amount, specified in kilobytes. When set to a value less than 0, the data provider providers better performance. However, memory use may be affected.

Properties for SQL Server 2000 (WebSphere embedded ConnectJDBC driver for MS SQL Server (XA))

databaseName=BPEDB
Required string. Defines which database to access.

serverName
Required string. The TCP/IP address of the SequeLink server in dotted format or host name format.

portNumber=1433
Required integer. The TCP/IP port number where the Microsoft SQL Server resides.

selectMethod=
Optional string. Determine whether or not Microsoft SQL Server 'server cursors' are used for SQL queries. Values are 'cursor' or 'direct'. See the ConnectJDBC documentation for more information.

dataSourceName=
Optional string. Name for the data source. Only used for information purposes.

spyAttributes=
Optional string. The SPY attributes. See the ConnectJDBC documentation for a list of attributes.

loginTimeout=
Optional integer. The maximum time to attempt to connect a database. If this value is non-zero, attempt to connect to the database will timeout when this specified value is reached. Zero implies no time limit.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Not used by the data source object. Used for information purposes only.

enable2Phase=true
Required boolean. When true, two phase connections are used. Do not change this value because this value is set for with this provider and process choreographer requires two-phase connections.

maxPooledStatements=
Optional integer. The maximum number of pooled PreparedStatements for this connection.

sendStringParametersAsUnicode=""
Optional boolean. Determines whether string parameters are sent to the SQL Server database as Unicode or in the default character encoding of the database. For more information, refer to the ConnectJDBC documentation.

enableMultithreadedAccessDetection=false
Optional boolean. If set to `true`, it automatically detects multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

Properties for SQL Server 2000 (WebSphere embedded ConnectJDBC driver for MS SQL Server (XA)) (continued)

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. If you enabled pre-test connection in the `j2c.properties` file, this pre-test string is used to make sure that the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

codePageOverride=""
Optional string. Specifies the code page the driver uses when converting character data. See the ConnectJDBC documentation for more information.

insensitiveResultSetBufferSize=2048
Optional integer. Determines the amount of memory used by the driver to cache insensitive result set data. See the ConnectJDBC documentation for more information.

Properties for Sybase 12.0 & Sybase 12.5

databaseName=BPEDB
Required string. Defines which database to access.

serverName=*hostname*
Required string. The name of the server where the Sybase database resides.

portNumber=4100
Required integer. The TCP/IP port number where the JDBC driver resides.

networkProtocol=
Optional string. Specifies which protocol is used, for example, `socket` or `SSL`. When set to `socket`, Secure Sockets Layer (SSL) encryption is not used.

dataSourceName=
Optional string. Name for the data source. Used for information purposes only.

version=
Optional integer. The version number of the driver.

resourceManagerName=
Optional string. The name of the resource manager.

loginTimeout=
Optional integer. The maximum time to attempt to connect a database. If this value is non-zero, attempt to connect to the database will timeout when this specified value is reached. If set to zero, there will be no timeout.

description=DataSource for Process Choreographer
Optional string. Description of the data source. Used for information purposes only.

connectionProperties=SELECT_OPEN_CURSORS=true
Required string. Refer to the Sybase documentation for more information about SELECT_OPEN_CURSORS and other properties.

enableMultithreadedAccessDetection=false
Optional boolean. If set to `true`, it automatically detects multithreaded access to a connection and its corresponding Statements, ResultSets, and MetaDatas.

preTestSQLString=SELECT 1 FROM TABLE1
Optional string. If you enabled pre-test connection in the `j2c.properties` file, this pre-test string is used to make sure that the connection is good. If you leave this field blank, the default SQL statement, SELECT 1 FROM TABLE1, will be used at runtime. This will slow down the execution because of the exception handling if table TABLE1 is not defined in the database. Users are recommended to provide their own SQL statement to improve the performance.

JMS provider

Specifies which messaging service the business process container uses.

Mandatory	Yes
Data type	Drop-down list

Choices

For external WebSphere MQ
WebSphere MQ JMS Provider
For the messaging embedded in WebSphere
WebSphere JMS Provider

Queue manager

The name of the queue manager that is used by the business process container.

Mandatory
If you selected WebSphere MQ JMS Provider; otherwise, this field is disabled.

Data type
String

Value
Your queue manager name, for example, WAS_*nodeName_serverName*.

Classpath (JMS provider)

The path to the MQ Java lib directory.

Mandatory
If you have not defined the WebSphere environment variable ${MQ_INSTALL_ROOT} to point to the WebSphere MQ installation root directory.

Enabled
If you selected WebSphere MQ JMS Provider; otherwise, this field is disabled.

Data type
String

Default
The default value for the class path depends on the local WebSphere MQ installation:
For AIX
/usr/mqm/java/lib
For Solaris and HP-UX
/opt/mqm/java/lib
For Windows
c:\Program Files\ibm\WebSphere MQ\Java\lib

JMS user ID

Used to authenticate the connection to the JMS provider. This user ID must have administration rights for the messaging service.

Mandatory
If you selected WebSphere JMS Provider; otherwise, this field is disabled.

Data type
String

Restrictions
If you are using embedded messaging, the JMS user ID must be less than or equal to 12 characters. For example, the default Windows NT user ID, Administrator, is not valid for use with WebSphere embedded messaging because the ID contains 13 characters.

Default
The user ID you used to log into the administrative console.

For UNIX
Use root. The user ID must be a member of the group mqm.

For Windows
Use the default user ID. This user ID must be the same one used to start WebSphere Application Server.

JMS password

The password for the JMS user ID.

Mandatory	If you selected `WebSphere JMS Provider`; otherwise, this field is disabled.
Data type	String
Default	None

Scheduler calendar

JNDI name of the UserCalendar scheduler for the business process container to use.

Mandatory	No
Data type	String
Default	`com/ibm/Websphere/scheduler/calendar/DefaultUserCalendar`

Security role mapping

The user or group from the domain user registry that is mapped onto the role of Business Process Administrator.

Mandatory	Yes
Data type	String
Default	**For UNIX and Linux** Depends on the local settings. AIX example: `Administrator` **For Windows** `Administrators`
Restrictions	The user registry can be the local operating system, Lightweight Directory Access Protocol (LDAP), or custom registry. The user or group specified must already exist in the user registry being used.

JMS API user ID

The user ID that the process choreographer message-driven bean uses when processing asynchronous API calls.

Mandatory	If WebSphere security is enabled (even if you do not use the Java Message Service API).
Data type	String

Description

If WebSphere security is enabled and you do not use the JMS API, you must specify a valid user ID. This ID does not need any special authorizations.

If WebSphere security is enabled and you plan to use the JMS API, this user ID must either be one that is given the appropriate authorities when the process is modeled, or more commonly, it must be a member of a group that was granted the necessary authorities during modeling. The possible staff authorities associated with processes are: Administrator, Reader, and Starter. For activities, a user ID can only perform the sendEvent action if it is a potential owner of the associated receiveEvent.

If you want to support all the actions on processes through the JMS API, you can specify a user ID that is a member of the J2EE BPESystemAdministrator role. However, in a production system, the more fine-grained security approach is recommended.

JMS API password

The password for the JMS API User ID.

Mandatory
If WebSphere security is enabled (even if you do not use the JMS API)

Data type
String

JMS resources (new or existing)

You must either create new JMS resources or select existing JMS resources.

Mandatory
Yes

Data type
Radio buttons

Choices
- Create new JMS resources using default values.
- Use existing JMS resources.

Default values for JMS resources:

Connection factory:
BPECF

Internal queue
BPEIntQueue

External request processing queue
BPEApiQueue

Hold queue
BPEHldQueue

Retention queue
BPERetQueue

Connection factory
The queue connection factory for the business process container to use.

Mandatory	Only if you chose Select existing JMS resources
Data type	Drop-down list
Default	`BPECF`

Internal queue
The JNDI name of the queue for internal business process container messages.

Mandatory	Only if you chose Select existing JMS resources
Data type	Drop-down list
Default	`BPEIntQueue`

External request processing queue
The JNDI name of the queue for external (JMS API) requests to the business process container.

Mandatory	Only if you chose Select existing JMS resources
Data type	Drop-down list
Default	`BPEApiQueue`

Hold queue
The JNDI name of the queue that holds any messages that have failed processing more times than the retry limit.

Mandatory	Only if you chose Select existing JMS resources
Data type	Drop-down list
Default	`BPEHldQueue`

Retention queue
The JNDI name of the queue that contains messages that temporarily cannot be processed, and that will be retried.

Mandatory	Only if you chose Select existing JMS resources
Data type	Drop-down list
Default	`BPERetQueue`

Web client
If this check-box is selected, the process choreographer Web client is also installed.

Data type	check-box
Default	`selected`
URL	`http://`*`hostname`*`/bpe/webclient`

Configuring the business process container manually

Before you can use the business process container, you must install it and configure the necessary resources. This topic describes how to do this manually.

1. Configure the JDBC provider and data source according to one of the following:
   - Using a script (recommended).
   - Using the administrative console.
2. Configuring the queue resources.
3. Creating and configuring the scheduler.
4. Installing the business process container.

The business process container has been installed and configured.

You must activate the business process container.

Using a script to configuring the JDBC provider and data source for the business process container

The business process container requires a JDBC provider and data source.
To configure your JDBC provider and data source using one of the JACL scripts provided, perform one of the following to actions:

- Configuring a Cloudscape JDBC provider and data source.
- 5.1+ Configuring a DB2 UDB for iSeries JDBC provider and data source.
- Configuring a DB2 UDB for Linux, UNIX, and Windows JDBC provider and data source.
- 5.1+ Configuring a DB2 UDB for z/OS JDBC provider and data source.
- 5.1+ Configuring an Informix Dynamic Server JDBC provider and data source.
- Configuring an Oracle JDBC provider and data source.
- Configuring an SQL Server JDBC provider and data source.
- Configuring a Sybase Adaptive Server JDBC provider and data source.

The process choreographer data source has been defined.

Configure the queue resources.

Configuring a Cloudscape JDBC provider and data source for the business process container

This topic describes how to configure a Cloudscape JDBC provider and data source for the business process container.

1. Change to the process choreographer directory: On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```

2. Run the configuration script. On Windows, enter:

   ```
   install_root\bin\wsadmin -f create_ds_Cloudscape.jacl
       [-node <nodeName>] -server <serverName> -dbPath <databasePath>
   ```

On UNIX, enter:

```
install_root/bin/wsadmin.sh -f create_ds_Cloudscape.jacl
    [-node <nodeName>] -server <serverName> -dbPath <databasePath>
```

Where:

wsadmin
is located in the bin subdirectory of the IBM WebSphere Application Server installation directory.

create_ds_Cloudscape.jacl
is the script file that performs the configuration within WebSphere. It is located in the ProcessChoreographer subdirectory of the IBM WebSphere Application Server installation directory.

nodeName
is the optional node name.

serverName
is the application server name.

databasePath
is the path to the process choreographer database directory. Note, on Windows use forward slashes (/) instead of back slashes (\). For example, "c:/mydbs/BPEDB".

For example, on Windows:

```
install_root\bin\wsadmin -f create_ds_Cloudscape.jacl
    -server server1
    -dbPath "c:/Program Files/WebSphere/AppServer/ProcessChoreographer/BPEDB"
```

3. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The Cloudscape JDBC provider and data source have been configured for the business process container.

Configure the queue resources.

Configuring a DB2 UDB for iSeries JDBC provider and data source for the business process container

This topic describes how to configure a DB2 UDB for iSeries JDBC provider and data source for the business process container.

1. On UNIX, configure WebSphere Application Server for DB2 access. Note: On Windows the environment is set up automatically.
2. Change to the process choreographer directory. On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```

3. Run the configuration script to create a data source that uses the DB2 UDB for iSeries Toolbox XA driver. On Windows, enter:

   ```
   install_root\bin\wsadmin -f create_ds_Db2iSeries.jacl
       [-node <nodeName>] -server <serverName>
       -jdbcDriverPath <driver path to toolbox driver>
       -dbServer <db server name>] -userid <userID> -password <password>
   ```

   On UNIX, enter:

   ```
   install_root/bin/wsadmin.sh -f create_ds_Db2iSeries.jacl
       [-node <nodeName>] -server <serverName>
       -jdbcDriverPath <driver path to toolbox driver>
       -dbServer <db server name> -userid <userID> -password <password>
   ```

Where:

wsadmin
is located in the `bin` subdirectory of the IBM WebSphere Application Server installation directory.

create_ds_Db2iSeries.jacl
is the script file that performs the configuration within WebSphere. It is located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory. For example, on a default Windows installation, it is in the directory `c:\Program Files\WebSphere\AppServer\ProcessChoreographer`.

nodeName
is an optional node name.

serverName
is the application server name.

driver path to toolbox driver
is the location where the Toolbox driver is installed on your Application Server machine. If necessary, you can download the JAR file for the driver from `/QIBM/ProdData/Http/Public/jt400/lib/jt400.jar`.

db server name
is the name of your iSeries system.

userID
is the user ID for accessing the database. This must be the same user ID that was used to create the database and the schema.

password
is the password for *userID*.

4. If you had problems using the script, you can perform Using the administrative console to configure the JDBC provider and data source.

The data source `BPEDataSourceDb2iSeries` has been created and is now listed in the data source panel.

Configuring a DB2 UDB for Linux, UNIX, and Windows JDBC provider and data source for the business process container

This topic describes how to configure a DB2 UDB JDBC provider and data source for the business process container.

1. On UNIX, configure WebSphere Application Server for DB2 access. Note: On Windows the environment is set up automatically.
2. Change to the process choreographer directory: On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```

3. Run the configuration script. It will create a data source that uses the DB2 Legacy CLI-based type-2 JDBC driver (XA). On Windows, enter:

   ```
   install_root\bin\wsadmin -f create_ds_Db2.jacl
       [-node <nodeName>] -server <serverName> -dbPath <databasePath>
       [-dbName <databaseName>] -userid <userID> -passwd <password>
   ```

   On UNIX, enter:

```
install_root/bin/wsadmin.sh -f create_ds_Db2.jacl
    [-node <nodeName>] -server <serverName> -dbPath <databasePath>
    [-dbName <databaseName>] -userid <userID> -passwd <password>
```

Where:

wsadmin
: is located in the `bin` subdirectory of the IBM WebSphere Application Server installation directory.

create_ds_Db2.jacl
: is the script file that performs the configuration within WebSphere. It is located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory. For example, on a default Windows installation, it is in the directory `c:/Program Files\WebSphere\AppServer\ProcessChoreographer`.

nodeName
: is the optional node name.

serverName
: is the application server name.

dbPath
: is the location where your DB2 system is installed. Note: On Windows you must use forward slashes (/) instead of back slashes (\) to specify the path, for example, `"c:/sqllib"`.

databaseName
: is the optional database name. If you do not provide a name the default `BPEDB` will be used.

userID
: is the user ID for accessing the database. This must be the same user ID that was used to create the database and the schema.

password
: is the password for *userid*.

4. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The DB2 JDBC provider and data source have been configured for the business process container.

Configure the queue resources.

Configuring a DB2 UDB for z/OS JDBC provider and data source for the business process container

This topic describes how to configure a DB2 UDB for z/OS JDBC provider and data source for the business process container.

1. Change to the process choreographer directory. On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```

2. Run the configuration script. It will create a data source for DB2 for z/OS that uses the DB2 Legacy CLI-based type-2 JDBC driver (XA). On Windows, enter:

   ```
   install_root\bin\wsadmin -f create_ds_Db2_zOs.jacl
       [-node <nodeName>] -server <serverName> -dbPath <databasePath>
       [-subsystem <subsystem>] -userid <userid> -password <password>
   ```

   On UNIX, enter:

```
install_root/bin/wsadmin.sh -f create_ds_Db2_zOs.jacl
    [-node <nodeName>] -server <serverName> -dbPath <databasePath>
    [-subsystem <subsystem>] -userid <userid> -password <password>
```

Where:

wsadmin
The script located in the bin subdirectory of the application server installation directory.

create_ds_Db2_zOs.jacl
The script file that performs the configuration within the application server. It is located in the `ProcessChoreographer` subdirectory of the application server installation directory. For example, on a default Windows installation, it is in the directory `c:\Program Files\WebSphere\AppServer\ProcessChoreographer`.

nodeName
The optional node name.

serverName
The application server name.

dbPath
The location where your DB2 Connect is installed. Note: On Windows you must use forward slashes ('/') instead of back slashes ('\') to specify the path, for example, `c:/sqllib`.

subsystem
The cataloged subsystem name. If you do not provide a name the default, `BPEDB`, will be used.

userid The user ID for accessing the database. This must be the same user ID that was used to create the database and the schema.

password
The password for userid.

3. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The DB2 z/OS CLI-based type 2 JDBC provider and data source have been configured for the business process container.

Configure the queue resources.

Configuring an Informix Dynamic Server JDBC provider and data source for the business process container

This topic describes how to configure an Informix Dynamic Server JDBC provider and data source for the business process container.

1. Change to the process choreographer directory: On Windows, enter:
   ```
   cd install_root\ProcessChoreographer
   ```
   On UNIX, enter:
   ```
   cd install_root/ProcessChoreographer
   ```
2. Run the configuration script. It will create an XA data source for Informix. On Windows, enter:

```
install_root\bin\wsadmin -f create_ds_Informix.jacl
  [-node <nodeName>] -server <serverName>
  -dbServer <dbServerMachineName> -dbPath <path>
  -dbInstance <dbServerInstanceName> [-dbPort <port>] [-dbName <database>]
  -userid <userID> -passwd <password>
```

On UNIX, enter:

```
install_root/bin/wsadmin.sh -f create_ds_Informix.jacl
  [-node <nodeName>] -server <serverName>
  -dbServer <dbServerMachineName> -dbPath <path>
  -dbInstance <dbServerInstanceName> [-dbPort <port>] [-dbName <database>]
  -userid <userID> -passwd <password>
```

Where:

wsadmin
: is located in the `bin` subdirectory of the IBM WebSphere Application Server installation directory.

create_ds_Informix.jacl
: is the script file that performs the configuration within WebSphere. It is located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory. For example, on a default Windows installation, it is in the directory `c:\Program Files\WebSphere\AppServer\ProcessChoreographer`.

nodeName
: is the node name. This is optional and defaults to the current node.

serverName
: is the application server name.

path
: is the directory where the Informix JDBC driver is installed in the subdirectory `jdbc/lib`. Note: On Windows you must use forward slashes (/) instead of back slashes (\) to specify the path, for example, if your driver is installed in `c:/Program Files/Informix/jdbc/lib` specify the path `c:/Program Files/Informix`.

dbServerMachineName
: is the name of the machine where the database server is installed.

dbServerInstanceName
: is the Informix instance name that contains the process choreographer database.

port
: is the port where the database server runs. If none is specified, the default is `1526`.

database
: is the database name. If you do not provide a name the default `BPEDB` will be used.

userID
: is the user ID for accessing the database. This must be the same user ID that was used to create the database and the schema.

password
: is the password for *userid*.

3. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The Informix JDBC provider and XA data source have been configured for the business process container.

Configure the queue resources.

Configuring a Microsoft SQL Server JDBC provider and data source for the business process container

This topic describes how to configure a Microsoft SQL Server JDBC provider and data source for the business process container.

1. Change to the process choreographer directory by entering the command:
   ```
   cd install_root\ProcessChoreographer
   ```
2. Run the configuration script by entering the command.

   ```
   install_root\bin\wsadmin -f create_ds_MsSql.jacl
       [-node <nodeName>] -server <serverName>
       [-driverType ( Embedded | SequeLink )]
       [-dbServer <dbServerName>] [-dbPort <port>]
       [-dbName <databaseName>]
       -userid <userid> -password <password>
   ```

   Where:

   wsadmin
   : is located in the `bin` subdirectory of the IBM WebSphere Application Server installation directory.

   create_ds_MsSql.jacl
   : is the script file that performs the configuration within WebSphere. It is located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory. For example, on a default Windows installation, it is in the directory `c:\Program Files\WebSphere\AppServer\ProcessChoreographer`.

   nodeName
   : is the optional node name.

   serverName
   : is the application server name. The default value is `localhost`.

   Embedded or SequeLink
   : indicates which type of JDBC driver to configure for the database access. If nothing is specified `Embedded` is the default. `Embedded` specifies the ConnectJDBC driver that is embedded in WebSphere. `SequeLink` specifies the DataDirect SequeLink driver that is embedded in WebSphere.

   dbServerName
   : is the name of the database server machine.

   dbPort
   : is the port used on the database server. The default value is `1433` for `Embedded` and `19996` for `SequeLink`.

   databaseName
   : is the optional name of the process choreographer database. The default value is `BPEDB`.

   userid is the user ID for accessing the database. This must be the same user ID that was used to create the database.

   password
   : is the password for *userid*.

   For example:

   ```
   install_root\bin\wsadmin -f create_ds_MsSql.jacl
       -server server1 -dbServer mymachine
       -dbPort 1433 -dbName BPEDB -userid sa -password Password
   ```
3. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The Microsoft SQL Server JDBC provider and XA data source have been configured for the business process container.

Configure the queue resources.

Configuring an Oracle JDBC provider and data source for the business process container

This topic describes how to configure an Oracle JDBC provider and data source for the business process container.

1. Change to the process choreographer directory: On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```

2. Run the configuration script. On Windows, enter:

   ```
   install_root\bin\wsadmin -f create_ds_Oracle.jacl
       [-node <nodeName>] -server <serverName>
       [-driverType ( oci8 | thin ) ]
       [-dbServer <dbServerName>] [-dbPort <port>]
       -dbPath <databasePath> [-tnsName <databaseName>]
       -userid <userid> -password <password>
   ```

   On UNIX, enter:

   ```
   install_root/bin/wsadmin.sh -f create_ds_Oracle.jacl
       [-node <nodeName>] -server <serverName>
       [-driverType ( oci8 | thin ) ]
       [-dbServer <dbServerName>] [-dbPort <port>]
      -dbPath <databasePath> [-tnsName <databaseName>]
       -userid <userid> -password <password>
   ```

   Where:

   wsadmin
   : is located in the `bin` subdirectory of the IBM WebSphere Application Server installation directory.

   create_ds_Oracle.jacl
   : is the script file that performs the configuration within WebSphere. It is located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory. For example, on a default Windows installation, it is in the directory `c:\Program Files\WebSphere\AppServer\ProcessChoreographer`.

   nodeName
   : is the optional node name.

   serverName
   : is the application server name.

   oci8 or thin
   : indicates which JDBC driver to configure for the database access. If nothing is specified `oci8` is the default, which selects the Oracle OCI JDBC driver. `thin` selects the Oracle thin JDBC driver. For performance reasons, `oci8` is recommended.

   dbServerName
   : is the machine name of the server that hosts the database. This is only required for the `thin` driver.

port is the port number where the database server listens to JDBC requests. This is only required for the `thin` driver. If no value is specified, the default value `1521` is used.

databasePath
is the location where your Oracle system is installed. Note: On Windows you must use forward slashes (/) instead of back slashes (\) to specify the path, for example, `"c:/oracle/ora90"`.

databaseName
is the TNS name. This is only required for the `oci8` driver and must match the name used for the process choreographer database in the file `tnsnames.ora`. If no value is specified the default used is `BPEDB`.

userid is the user ID for accessing the database. This must be the same user ID that was used to create the database.

password
is the password for *userid*.

For example, on Windows:

```
install_root\bin\wsadmin -f create_ds_Oracle.jacl
    -server server1 -dbPath "c:/oracle/ora90" -tnsName BPEDB
    -userid system -password OraclePassword
```

3. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The Oracle JDBC provider, an XA and a non-XA data source have been configured for the business process container.

Configure the queue resources.

Configuring a Sybase JDBC provider and data source for the business process container

This topic describes how to configure a Sybase JDBC provider and data source for the business process container.

1. Change to the process choreographer directory: On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```

2. Run the configuration script. On Windows, enter:

   ```
   install_root\bin\wsadmin -f create_ds_Sybase.jacl
       [-node <nodeName>] -server <serverName> -dbPath <databasePath>
       [-dbServer <dbServerName>] [-dbPort <port>] [-dbName <databaseName>]
       -userid <userID> -password <password>
   ```

   On UNIX, enter:

   ```
   install_root/bin/wsadmin.sh -f create_ds_Sybase.jacl
       [-node <nodeName>] -server <serverName> -dbPath <databasePath>
       [-dbServer <dbServerName>] [-dbPort <port>] [-dbName <databaseName>]
       -userid <userID> -password <password>
   ```

   Where:

   wsadmin
   is located in the `bin` subdirectory of the IBM WebSphere Application Server installation directory.

create_ds_Sybase.jacl
is the script file that performs the configuration within WebSphere. It is located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory. For example, on a default Windows installation, it is in the directory `c:\Program Files\WebSphere\AppServer\ProcessChoreographer`, and on UNIX, the default directory is `/opt/WebSphere/AppServer/ProcessChoreographer`.

nodeName
is the optional node name. The default is the local node.

serverName
is the application server name.

databasePath
is the location where your Sybase system is installed. Note: On Windows you must use forward slashes (`/`) instead of back slashes (`\`) to specify the path. For example, if your JDBC driver is located in the path `c:/Sybase/jConnect-5_2/classes/jconn2.jar` your value for the `dbPath` must be `c:/Sybase`.

dbServerName
is the name of the database server machine. The default is the local host.

dbPort
is the port used on the database server. The default is 4100.

databaseName
is the name of the process choreographer database. If you do not specify a name, the default `BPEDB` is used.

userID
is the user ID for accessing the database. This must be the same user ID that was used to create the database.

password
is the password for *userID*.

For example, on Windows:

```
install_root\bin\wsadmin -f create_ds_Sybase.jacl
    -server server1 -dbPath "c:/sybase" -dbServer mymachine
    -dbPort 4100 -dbName BPEDB -userid sa -password SybaseAdminPassword
```

3. If you had problems using the script, you can use the administrative console to configure the JDBC provider and data source.

The Sybase JDBC provider and data source have been configured for the business process container.

Configure the queue resources.

Using the administrative console to configure the JDBC provider and data source

This topic describes how to configure a JDBC provider and data source for the business process container using the administrative console.

1. In the administrative console, click **Resources> JDBC Providers**.
2. Click **New**.
3. In the drop-down list for **JDBC Providers**, select the template for your database:
   - For Cloudscape, select **Cloudscape JDBC Driver (XA)**.

- For DB2 UDB for Linux, UNIX, and Windows, select **DB2 Legacy CLI-based Type 2 JDBC Driver (XA)**.
- For DB2 UDB for iSeries, select **template DB2 UDB for iSeries (Toolbox XA)**.
- For a remote DB2 UDB for z/OS, select **DB2 Legacy CLI-based Type 2 JDBC Driver (XA)**.
- For Informix Dynamic Server, select **Informix JDBC Driver (XA)**.
- For Oracle, select **Oracle JDBC Driver (XA)**.
- For Microsoft SQL Server, using the WebSphere embedded ConnectJDBC driver, select **WebSphere embedded ConnectJDBC driver for MS SQL Server (XA)**.
- For Microsoft SQL Server, using the DataDirect SequeLink JDBC driver, select **DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA)**.
- For Sybase Adaptive Server, select **Sybase JDBC Driver (XA)**.

4. Click **Apply**.
5. Enter the JDBC driver name:
   - For Cloudscape, enter `BPEJdbcDriverCloudscape`.
   - For DB2 UDB for Linux, UNIX, and Windows, enter `BPEJdbcDriverDB2`.
   - For DB2 UDB for iSeries, enter `BPEJdbcDriverDB2iSeries`.
   - For DB2 UDB for z/OS, enter `BPEJdbcDriverDB2zOS`.
   - For Informix Dynamic Server, enter `BPEJdbcDriverInformix`.
   - For Oracle, enter `BPEJdbcDriverOracle`.
   - For Microsoft SQL Server, enter `BPEJdbcDriverMsSql`.
   - For Sybase Adaptive Server, enter `BPEJdbcDriverSybase`.
6. Enter a suitable description for the JDBC provider, for example, `JDBC Provider for Process Choreographer`
7. Enter the class path for the JDBC driver:
   - For Cloudscape, enter `${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar`.
   - For DB2 UDB for Linux, UNIX, and Windows, enter `${DB2_JDBC_DRIVER_PATH}/db2java.zip`.
   - For DB2 UDB for iSeries, enter `${OS400_TOOLBOX_JDBC_DRIVER_PATH}/jt400.jar`.
   - For DB2 UDB for z/OS, enter `${DB2_JDBC_DRIVER_PATH}/db2java.zip`.
   - For Informix Dynamic Server, enter:

     ```
     ${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbc.jar
     ${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbcx.jar
     ```

   - For Oracle, enter `${ORACLE_JDBC_DRIVER_PATH}/ojdbc14.jar`.
   - For Microsoft SQL Server, using the WebSphere embedded ConnectJDBC driver, enter:

     ```
     ${WAS_LIBS_DIR}/sqlserver.jar
     ${WAS_LIBS_DIR}/base.jar
     ${WAS_LIBS_DIR}/util.jar
     ${WAS_LIBS_DIR}/spy.jar
     ```

   - For Microsoft SQL Server, using the DataDirect SequeLink JDBC driver, enter:

     ```
     ${WAS_LIBS_DIR}/sljc.jar
     ${WAS_LIBS_DIR}/spy-sl53.jar
     ```

   - For Sybase Adaptive Server, enter `${SYBASE_JDBC_DRIVER_PATH}/jconn2.jar`.

Note: Click **Environment> Manage WebSphere Variables**, and check that the variable for the JDBC driver path is set to an appropriate value. For example, on Windows systems to `c:/sqllib/java`. For Microsoft SQL Server, the variable `WAS_LIBS_DIR` must point to the WebSphere `lib` directory

8. Enter the implementation class name:
   - For Cloudscape, enter `com.ibm.db2j.jdbc.DB2jXADataSource`.
   - For DB2 UDB for Linux, UNIX, and Windows, enter `COM.ibm.db2.jdbc.DB2XADataSource`.
   - For DB2 UDB for iSeries, enter `com.ibm.as400.access.AS400JDBCXADataSource`.
   - For DB2 UDB for z/OS, enter `COM.ibm.db2.jdbc.DB2XADataSource`.
   - For Informix Dynamic Server, enter `com.informix.jdbcx.IfxXADataSource`.
   - For Oracle, enter `oracle.jdbc.xa.client.OracleXADataSource`.
   - For Microsoft SQL Server, using the WebSphere embedded ConnectJDBC driver, enter `com.ibm.websphere.jdbcx.sqlserver.SQLServerDataSource`.
   - For Microsoft SQL Server, using the DataDirect SequeLink JDBC driver, enter `com.ddtek.jdbcx.sequelink.SequeLinkDataSource`.
   - For Sybase Adaptive Server, enter `com.sybase.jdbc2.jdbc.SybXADataSource`.
9. Click **Apply** and **Save**.
10. Click **Resources> JDBC Provider**
11. Select the name of the new provider that you just created.
12. In **Additional Properties**, select **Data Sources**.
13. Click **New**.
14. On the data source panel perform the following actions:
    a. Enter the data source name:
       - For Cloudscape, enter `BPEDataSourceCloudscape`.
       - For DB2 UDB for Linux, UNIX, and Windows, enter `BPEDataSourceDb2`.
       - For DB2 UDB for iSeries, enter `BPEDataSourceDb2iSeries`.
       - For DB2 UDB for z/OS, enter `BPEDataSourceDb2zOS`.
       - For Informix Dynamic Server, enter `BPEDataSourceInformix`.
       - For Oracle, enter `BPEDataSourceOracle`.
       - For Microsoft SQL Server, enter `BPEDataSourceMsSql`.
       - For Sybase Adaptive Server, enter `BPEDataSourceSybase`.
    b. Enter the JNDI name, for example, `jdbc/BPEDB`.
    c. Select the check box for container managed persistence.
    d. Enter a suitable description, for example, `DataSource for Process Choreographer`.
    e. Enter the category `Process Choreographer`.
    f. Enter the statement cache size, for example, `30` .
    g. Enter the data source helper class name:
       - For Cloudscape, enter `com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper`.
       - For DB2 UDB for Linux, UNIX, and Windows, enter `com.ibm.websphere.rsadapter.DB2DataStoreHelper`.
       - For DB2 UDB for iSeries, enter `com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper`.

- For DB2 UDB for z/OS, do not use the default value for this field, enter `com.ibm.websphere.rsadapter.DB2390DataStoreHelper`.
- For Informix Dynamic Server, enter `com.ibm.websphere.rsadapter.InformixDataStoreHelper`.
- For Oracle, enter `com.ibm.websphere.rsadapter.OracleDataStoreHelper`.
- For Microsoft SQL Server, using the WebSphere embedded ConnectJDBC driver, enter `com.ibm.websphere.rsadapter.WSConnectJDBCDataStoreHelper`.
- For Microsoft SQL Server, using the DataDirect SequeLink JDBC driver, enter `com.ibm.websphere.rsadapter.SequeLinkDataStoreHelper`.
- For Sybase Adaptive Server, enter `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`.

h. If you are not using a Cloudscape database, select the authentication alias that you want to use to access the database. If necessary, create a new component-managed authentication alias in **Security> JAAS Configuration> J2C Authentication Data**, by selecting **New**, entering the user ID and password to access the database, then clicking **Apply** and **Save**.

i. Click **Apply**.

15. Scroll down to **Additional Properties** and select **Custom Properties**.
16. Verify that the settings for your data source match the values described in Install Wizard settings. If necessary click on a property and change the value.
17. Save your changes.
18. If you are using Oracle, you must also create a non-XA data source, by repeating all the steps in this task, but using the following values:

Field	Value for non-XA Oracle data source
JDBC provider name	`Oracle JDBC Driver`
JDBC driver name	`BPEJdbcDriverOracleNonXA`
Classpath	`${ORACLE_JDBC_DRIVER_PATH}/ojdbc14.jar`
Implementation class name	`oracle.jdbc.pool.OracleConnectionPoolDataSource`
Data source name	`BPEDataSourceOracleNonXA`
Data source helper class name	`OracleDataStoreHelper`

The JDBC provider and data source for your database have been configured for the business process container. The new data source is now listed in the data source panel.

Configure the queue resources.

Configuring the queue resources for the business process container

The business process container uses a JMS provider for sending and receiving messages. If you have installed the IBM WebSphere Application Server ″embedded messaging″ option, it is recommended to use it. Otherwise, you must have already installed your external messaging system and created the queue manager and queues.

Configure the queues, JMS provider, and message listener service, by performing one of the following:

- **Embedded messaging:** Configuring queue resources using the JMS provider embedded in WebSphere.
- **External messaging:** Configuring queue resources using WebSphere MQ.

The queue resources needed by the business process container have been created.

Create and configure the scheduler

Configuring queue resources for the business process container using the JMS provider embedded in WebSphere

Each business process container needs message queues, a JMS provider, and a message listener service. This topic explains how to create these resources using WebSphere's embedded messaging.

1. Make sure that you have the embedded messaging option installed on your WebSphere Application Server.
2. Change to the process choreographer directory: On Windows, enter:

   ```
   cd install_root\ProcessChoreographer
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer
   ```
3. Configure the queue resources, by running `config_queue_emb.jacl` with the appropriate parameters. On Windows, enter:

   ```
   install_root\bin\wsadmin -f config_queue_emb.jacl
       -userid <userid> -password <password>
       [-node <nodeName>] -server <serverName>
       [-internalQueue <internalQueueName>]
       [-apiQueue <apiQueueName>]
       [-holdQueue <holdQueueName>]
       [-retentionQueue <retentionQueueName>]
   ```

   On UNIX, enter:

   ```
   install_root/bin/wsadmin.sh -f config_queue_emb.jacl
       -userid <userid> -password <password>
       [-node <nodeName>] -server <serverName>
       [-internalQueue <internalQueueName>]
       [-apiQueue <apiQueueName>]
       [-holdQueue <holdQueueName>]
       [-retentionQueue <retentionQueueName>]
   ```

   The default values for the optional parameters are:

node	<local node>
internalQueueName	BPEIntQueue
apiQueueName	BPEApiQueue
holdQueueName	BPEHldQueue
retentionQueueName	BPERetQueue

4. You can check or change the configuration settings using the Administrative Console.

The queue resources needed by the business process container have been created.

Create and configure the scheduler.

Configuring the queue resources for the business process container using WebSphere MQ

Each business process container needs a JMS provider, and message listener ports defined for it. This topic explains how to create these resources.

1. Make sure that you know the password for a user ID that has the authority to create WebSphere MQ queues.
2. Change to the process choreographer directory: On Windows, enter:
   ```
   cd install_root\ProcessChoreographer
   ```
   On UNIX systems, enter:
   ```
   cd install_root/ProcessChoreographer
   ```
3. Configure the queue resources by running script `config_queue_mq.jacl` with the appropriate parameters: On Windows, enter:

   ```
   install_root\bin\wsadmin -f  config_queue_mq.jacl
       [-node <nodeName>] -server <serverName>
       -mqPath <MQPath> [-qmName <queue manager name>]
       [-qmNamePut <cluster queue manager name>
       [-internalQueue <internalQueueName>] [-apiQueue <apiQueueName>]
       [-holdQueue <holdQueueName>] [-retentionQueue <retentionQueueName>]
   ```

   For example, for a Windows configuration based on the defaults:

   ```
   install_root\bin\wsadmin -f config_queue_mq.jacl
       -server server1 -mqPath C:\Progra~1\MQSeries
   ```

   On UNIX systems, enter:

   ```
   install_root/bin/wsadmin.sh -f config_queue_mq.jacl
       [-node <nodeName>] -server <serverName>
       -mqPath <MQPath> [-qmName <queue manager name>]
       [-qmNamePut <cluster queue manager name>]
       [-internalQueue <internalQueueName>] [-apiQueue <apiQueueName>]
       [-holdQueue <holdQueueName>] [-retentionQueue <retentionQueueName>]
   ```

   For example, for an AIX configuration based on the defaults:

   ```
   install_root/bin/wsadmin.sh -f  config_queue_mq.jacl
     -server server1 -mqPath /usr/mqm
   ```

   The default values for the optional parameters are:
   node <local node>
   qmName
   WAS_<nodeName>_<serverName>
   qmNamePut
   WAS_<nodeName>_<serverName>. This parameter is only required if

you are configuring a cluster setup that uses a WebSphere MQ cluster with separate ″put″ and ″get″ queue managers.

internalQueueName
BPEIntQueue

apiQueueName
BPEApiQueue

holdQueueName
BPEHldQueue

retentionQueueName
BPERetQueue

4. If you have problems using the script `config_queue_mq.jacl`, you can also configure the resources manually using the administrative console.
5. You can check or change the configuration settings using the administrative console.

The queue resources needed by the business process container have been created.

Create and configure the scheduler.

Configuring MQ resources for the business process container using the administrative console:

You must have already created the queues for the business process container.

It is recommended that you configure the resources using the script provided. If you must create the resources manually, this topic describes how to do it using the administrative console:

1. Select **Resources > WebSphere MQ JMS Providers**.
2. Click **WebSphere MQ Queue Connection Factory**.
3. Click **New**.
4. Enter the following values:

Field	Example value
Name	`BPECF`
JNDI Name	`jms/BPECF`
Queue Manager	`WAS_nodename_servername`

Leave the other fields blank or accept the default values.

5. Click **Apply**, and click **Save**. The MQ Queue Connection factory `BPECF` has been created and is listed on the MQ queue connection factory panel.
6. Define the MQ Queue Destination for the external message queue:
   a. Select **Resources > WebSphere MQ JMS Providers**.
   b. Click **WebSphere MQ Queue Destination**.
   c. Click **New**.
   d. Enter the following values:

Field	Example value
Name	`BPEApiQueue`
JNDI Name	`jms/BPEApiQueue`
Specified Priority	`3`
Specified Expiry	`3`

Base Queue Name	BPEApiQueue

Leave the other fields blank or accept the default values.

e. Click **Apply** then **Save**.

7. Define the MQ Queue Destination for the internal messages queue:
   a. Click **WebSphere MQ Queue Destination**.
   b. Click **New**.
   c. Enter the following values:

Field	Example value
Name	BPEIntQueue
JNDI Name	jms/BPEIntQueue
Specified Priority	3
Specified Expiry	3
Base Queue Name	BPEIntQueue

Leave the other fields blank or accept the default values.

   d. Click **Apply**, then **Save**.
8. Define the MQ Queue Destination for the hold queue:
   a. Click **WebSphere MQ Queue Destination**.
   b. Click **New**.
   c. Enter the following values:

Field	Example value
Name	BPEHldQueue
JNDI Name	jms/BPEHldQueue
Specified Priority	3
Specified Expiry	3
Base Queue Name	BPEHldQueue

Leave the other fields blank or accept the default values.

   d. Click **Apply**, then **Save**.
9. Define the MQ Queue Destination for the retention queue:
   a. Click **WebSphere MQ Queue Destination**.
   b. Click **New**.
   c. Enter the following values:

Field	Example value
Name	BPERetQueue
JNDI Name	jms/BPERetQueue
Specified Priority	3
Specified Expiry	3
Base Queue Name	BPERetQueue

Leave the other fields blank or accept the default values.

   d. Click **Apply**, then **Save**.

Now the four queue destinations have been defined and are listed on the queue panel.

10. Create the listener port for external messages:
    a. Click **Servers > Applications >** *YourServer* **> Message Listener Service > Listener Ports**.
    b. Click **New**.
    c. Enter the following values:

Field	Example value
Name	`BPEApiListenerPort`
Description	`Process choreographer API`
Connection Factory JNDI Name	`jms/BPECF`
Destination JNDI Name	`jms/BPEApiQueue`
Max Sessions	`5`
Max Retries	`10`
Max Messages	`1`

    d. Click **Apply**, then **Save**.
11. Create the listener port for the hold queue:
    a. In **Servers > Applications >** *YourServer* **> Message Listener Service > Listener Ports**, click **New**.
    b. Enter the following values:

Field	Example value
Name	`BPEHoldListenerPort`
Description	`Process choreographer hold`
Connection Factory JNDI Name	`jms/BPECF`
Destination JNDI Name	`jms/BPEHldQueue`
Max Sessions	`5`
Max Retries	`10`
Max Messages	`1`

    c. Click **Apply**, then **Save**.
12. Create the listener port for internal messages:
    a. In **Servers > Applications >** *YourServer* **> Message Listener Service > Listener Ports**, click **New**.
    b. Enter the following values:

Field	Example value
Name	`BPEInternalListenerPort`
Description	`Process choreographer internal`
Connection Factory JNDI Name	`jms/BPECF`
Destination JNDI Name	`jms/BPEIntQueue`
Max Sessions	`5`
Max Retries	`10`
Max Messages	`1`

    c. Click **Apply**, then **Save**.

The three listener ports have been created and are listed on the listener port panel.

The queue resources needed by the business process container have been created.

Create and configure the scheduler.

Creating and configuring the scheduler service for process choreographer

The business process container uses the scheduler service to provide time-dependent services. To create and configure the scheduler, perform the following:

1. Change to the process choreographer directory: On Windows, enter:
   ```
   cd install_root\ProcessChoreographer
   ```
   On UNIX, enter:
   ```
   cd install_root/ProcessChoreographer
   ```
2. Run the createScheduler.jacl script. On Windows, enter:
   ```
   install_root\bin\wsadmin -f createScheduler.jacl
       -server <serverName> [-node <nodeName>]
   ```
   On UNIX, enter:
   ```
   install_root/bin/wsadmin.sh -f createScheduler.jacl
       -server <serverName> [-node <nodeName>]
   ```
   Where:
   serverName
   is the name of the application server.
   nodeName
   is the name of the node. This is optional. If the node is omitted, the local node is used.
3. You can check or change the configuration settings using the administrative console.

The scheduler has been created and configured for process choreographer.

Install the business process container.

Installing the business process container

Create a new business process container and enter its settings using the administrative console:

1. Click **Servers > Application Servers >** *server_name*.
2. In the **Additional Properties** section, click **Business Process Container**.
3. On the configuration page, if necessary, replace the default values with your values.
4. For **JMS API User ID** enter the user ID to be used by the business process container when processing asynchronous API calls.
5. For **JMS API password** enter the password for the JMS API user ID.
6. Click **Install**. This causes the business process container's EAR file to be installed and configured. It is OK if you get an error message telling you that

the container is already configured. However, if you want to change any of the container's configuration settings, you must uninstall the appropriate `BPEContainer` application and then repeat this task from step 1.

7. Click **Save**.

The business process container has been installed and configured.

You must activate the business process container.

Business process container settings

Use this page to manage business process containers.

A business process container provides services to run business processes within an application server. To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Business Process Container**.

Attention: You cannot change these fields after they have been applied. If you want to change any of these values after the container is configured, you must uninstall the appropriate `BPEContainer` application and reinstall it. This action can result in the loss of data such as pending messages in queues, process templates, and process instances in the process choreographer database.

Datasource:

The Java Naming and Directory Interface (JNDI) name of the data source, which stores the process data.

Data type	String
Default	jdbc/BPEDB

Security Role For Process Administrator:

The user or group from the domain user registry that is to map onto the role of Business Process Administrator.

Data type	String
Default	ProcessAdministrator
Restrictions	The user registry can be the local operating system, Lightweight Directory Access Protocol (LDAP), or custom registry. The user or group specified must already exist in the user registry that is used.

Listener Port For Internal Messages:

This listener port provides the settings for the message-driven bean that handles messages exchanged within process choreographer processes.

Data type	String
Default	BPEInternalListenerPort

Listener Port For External Requests:

This listener port provides the settings for the message-driven bean that handles requests from process choreographer JMS API clients.

Data type	String
Default	BPEApiListenerPort

Listener Port For Unprocessed Messages:

This listener port provides the settings for the queue that contains the messages that cannot be processed.

Data type	String
Default	BPEHoldListenerPort

Retry Limit:

Specifies the maximum number of retries for processing a message. When the limit is reached, the message is sent to the Listener Port for Unprocessed Messages.

Data type	Integer
Default	5
Range	2 to 10 (recommended)

Retention Queue:

The queue that contains messages that temporarily cannot be processed.

Data type	String
Default	jms/BPERetQueue

Retention Queue Factory:

The factory for the retention queue.

Data type	String
Default	jms/BPECF

Retention Queue Message Limit:

The maximum number of messages that can be stored in the retention queue. When the limit is reached, the messages are sent to the Queue for internal messages again and the process container switches into the quiesce mode.

Data type	Integer
Default	20

JMS API User ID:

The user ID that the process choreographer message-driven bean uses when processing asynchronous API calls. This ID is only required when WebSphere security is enabled (even if you do not use the Java Message Service API).

Mandatory	If WebSphere security is enabled (even if you do not use the JMS API).
Data type	String

Description	If you do not use the JMS API, you must specify a valid user ID. This ID does not need any special authorizations. If you will use the JMS API, this user ID must either be given the appropriate authorities when the process is modeled, or more commonly, it must be a member of a group that has the necessary authorizations. The possible staff authorities associated with processes are: Administrator, Reader, and Starter. A user ID can only perform a sendEvent action if it is a potential owner of the associated receiveEvent activity. If you want to support all actions on processes through the JMS API, you can specify a user ID that is a member of the J2EE BPESystemAdministrator role. However, in a production system, the more fine-grained security approach is recommended.

JMS API Password:

The password for the JMS API user ID.

Data type	String

Scheduler Calendar:

The JNDI name of the user calendar scheduler for the process container to use.

Data type	String
Default	`com/ibm/Websphere/scheduler/calendar/DefaultUserCalendar`

Setting the JNDI name of the calendar EJB for process choreographer

This topic describes how to set the scheduler calendar property. It will be used when a duration is specified on the staff activity server property page for a V5.0-style process.

Process choreographer allows you to specify the maximum time that a staff activity can take to be completed after it has been reached by the process navigation. To specify a limit, you must set the **Calendar** and **Duration** fields on the server property page of the staff activity. The duration must be specified in a format that is suitable for the user calendar specified. If a calendar is specified, it is interpreted as a method of a user calendar session bean whose JNDI name is specified in the **Scheduler Calendar** property of the business process container. If no JNDI name is specified, the user calendar that is supplied with the WebSphere scheduler is used. To modify the **Scheduler Calendar** property, perform the following:

1. Change to the process choreographer directory: On Windows, enter:

   `cd` *`install_root`*`\ProcessChoreographer`

   On UNIX, enter:

```
cd install_root/ProcessChoreographer
```

2. Run the `setCalendar.jacl` script: If you are configuring for a single server on Windows, enter:

```
install_root\bin\wsadmin -f setCalendar.jacl
    -jndiName <jndiName> -server <Server> -node <Node>
```

When configuring for a single server on UNIX, enter:

```
install_root/bin/wsadmin.sh -f setCalendar.jacl
    -jndiName <jndiName> -server <Server> -node <Node>
```

When configuring for a cluster on Windows, enter:

```
install_root\bin\wsadmin -f setCalendar.jacl
    -jndiName <jndiName> -cluster <Cluster>
```

When configuring for a cluster on UNIX, enter:

```
install_root/bin/wsadmin.sh -f setCalendar.jacl
    -jndiName <jndiName> -cluster <Cluster>
```

Where:

Cluster
is the name of the cluster.

jndiName
is the JNDI name of the calendar EJB to use. The default value is `com/ibm/websphere/scheduler/calendar/DefaultUserCalendarHome`.

Node is the name of the node. This is optional. If the node is omitted, the local node is used.

Server is the name of the application server. If only one server exists, this parameter is optional.

If you omit a parameter, you will be prompted for it.

3. Optional: You can check or change the configuration settings using the administrative console to access the server property page of the staff activity.

The JNDI name of the calendar EJB has been configured for process choreographer.

Activating the business process container

To activate the business process container, you must restart your application servers.

1. If you installed the business process container on a cluster of application servers, restart the cluster using RippleStart.
2. If you installed the business process container on one application server, restart the application server:

The business process container is ready to run business processes.

Verify that the business process container works.

Verifying that the business process container works

The business process container must be configured and the database system and messaging service must have been started.

To verify that the business process container is working, you can either use the sample provided, or run your own application.

1. If you want to run the **process choreographer** sample, open the Samples Gallery.
2. If you want to use your own enterprise application that contains processes: Install your application using the administrative console. If it contains processes, the process templates will be written into the process choreographer database. The process templates are automatically enabled, and process instances will be created as soon as requests arrive from a client.
3. In case of problems, see troubleshooting the business process container.

The business process container is working.

Troubleshooting the business process container

For information about process-specific messages, tracing, and audit trails, see Troubleshooting process choreographer. Here are some things to check if you have problems getting the business process container to work:

1. If tables or views cannot be found in the database. When configuring the authentication alias for the data source, you must specify the same user ID that was used to create the database (or to run the scripts to create it).
2. If you get a database error when installing an enterprise application that contains a process. When an enterprise application is installed, any process templates are written into the process choreographer database. Make sure that the database system used by the business process container is running and accessible.
3. If you cannot invoke Cloudscape tools. Make sure that you have set up the Java environment, and have included the necessary JAR files in the `classpath` environment variable.
4. If you have problems using national characters. Make sure that your database was created with support for Unicode character sets.
5. If you have problems creating the queues using WebSphere MQ: WebSphere MQ dll not found. Add the WebSphere MQ Java lib directory to your `path` environment variable. For example, on Windows, enter the command:

   ```
   set path=MQInstallationDirectory\java\lib;%path%
   ```

   where *MQInstallationDirectory* is the installation directory for WebSphere MQ.
6. Problem on AIX connecting to the queue manager. Edit the file `/var/mqm/mqs.ini`, and add one of the following properties to the definition for your queue manager:

   ```
   IPCCBaseAddress=12
   IPCCBaseAddress=4
   ```
7. To avoid deadlocks, make sure your DB2 system is configured to use sufficient memory, especially for the bufferpool. Use the DB2 Configuration Advisor to determine reasonable values for your configuration.
8. If Sybase reports the exception: `The 'ALTER TABLE' command is not allowed within a multi-statement transaction in the 'databaseName' database.` This can happen after you have installed a new PTF level that requires the database to be migrated. Start the Sybase `isql` command program, enter the following commands:

```
use master
go
sp_dboption databaseName, "ddl in tran", true
go
```

where *databaseName* is the name of your database.

Chapter 4. Uninstalling the business process container

Use this task to remove the business process container configuration.

This task uses a script to remove the whole business process container configuration except for the non-WebSphere resources: the database and queue manager. If you want to reuse parts of the existing configuration, use the administrative console to remove part or all of the business process container configuration.

Before you can uninstall the business process container, you must stop all process templates, delete all process instances, then stop and uninstall all enterprise applications that contain business processes.

1. Ensure that all the stand-alone servers, the database, and the application server (or at least one application server per cluster) are running.
2. For each enterprise application that contains business processes, stop all process templates:
   a. For V5.0-style processes , select **Enterprise Applications> Application Name> Business Process Modules**. Then for each process archive (.far file), click **Templates> Select All> Stop> Save> Save**
   b. 5.1+ For BPEL-based processes, select **Enterprise Applications> Application Name> EJB Modules**. Then for each EJB archive (.jar file), click **Business Processes> Select All> Stop> Save> Save**.
   c. Wait for any running instances to finish.

      Note: To see if any instances are running, you can log on to the Web client as administrator, and view the processes that are "Administered By Me". If the list is empty, no instances exist.
   d. If there are any instances that have the `autoDelete` flag set to false, or that cannot be completed for some other reason, you must delete them manually.
   e. Stop and uninstall the application.
   f. If there are any more applications that contain business processes, repeat step 2.
3. Change to the process choreographer `sample` directory: On Windows, enter the command:

   ```
   cd install_root\ProcessChoreographer\sample
   ```

   On UNIX, enter:

   ```
   cd install_root/ProcessChoreographer/sample
   ```
4. Run the `bpeunconfig.jacl` script. When removing the configuration for a single server on Windows with WebSphere security enabled, enter the command:

   ```
   install_root\bin\wsadmin -f bpeunconfig.jacl -server Server -node Node
                -userid userID  -password password
   ```

   When removing the configuration for a single server on Windows with WebSphere security disabled, enter the command:

   ```
   install_root\bin\wsadmin -f bpeunconfig.jacl -server Server -node Node
   ```

   When removing the configuration for a single server on UNIX with WebSphere security enabled, enter the command:

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl  -server Server -node Node
            -userid userID  -password password
```

When removing the configuration for a single server on UNIX with WebSphere security disabled, enter the command:

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl  -server Server -node Node
```

When removing the configuration for a cluster on Windows with WebSphere security enabled, enter the command:

```
install_root\bin\wsadmin -f bpeunconfig.jacl -cluster Cluster
            -userid userID  -password password
```

When removing the configuration for a cluster on Windows with WebSphere security disabled, enter the command:

```
install_root\bin\wsadmin -f bpeunconfig.jacl -cluster Cluster
```

When removing the configuration for a cluster on UNIX with WebSphere security enabled, enter the command:

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl -cluster Cluster
            -userid userID  -password password
```

When removing the configuration for a cluster on UNIX with WebSphere security disabled, enter the command:

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl -cluster Cluster
```

Where:

Server The name of the application server. If only one server exists, this parameter is optional.

Node The name of the node. This is optional. If the node is omitted, the local node is used.

Cluster
The name of the cluster.

If you omit a parameter, you will be prompted for it.

The business process container has been uninstalled, but the process choreographer database and queue manager have not been removed. If you used the script, all resources related to the business process container (such as scheduler, data sources, listener ports, connection factories, and queue destinations) have been removed. If you uninstalled manually, the resources remain, and can either be uninstalled separately, or reused.

Using the administrative console to remove part or all of the business process container configuration

Use this task to remove part of the business process container configuration. The database and queue manager will not be deleted.

Before you can uninstall the business process container, you must stop all process templates, delete all process instances, then stop and uninstall all enterprise applications that contain business processes.

1. Ensure that all the stand-alone servers, the database, and the application server (or at least one application server per cluster) are running.
2. For each enterprise application that contains business processes, stop all process templates:
   a. For V5.0-style processes , select **Enterprise Applications> Application Name> Business Process Modules**. Then for each process archive (.far file), click **Templates> Select All> Stop> Save> Save**

b. 5.1 + For BPEL-based processes, select **Enterprise Applications> Application Name> EJB Modules**. Then for each EJB archive (.jar file), click **Business Processes> Select All> Stop> Save> Save**.

c. Wait for any running instances to finish.

Note: To see if any instances are running, you can log on to the Web client as administrator, and view the processes that are "Administered By Me". If the list is empty, no instances exist.

d. If there are any instances that have the `autoDelete` flag set to false, or that cannot be completed for some other reason, you must delete them manually.

e. Stop and uninstall the application.

f. If there are any more applications that contain business processes, repeat step 2.

3. In the administrative console, select **Applications > Enterprise Applications > BPEContainer_*identifier***. Where *identifier* depends on where you installed the business process container.

Installed the business process container on	**BPEContainer_*identifier***
Application server	`BPEContainer_nodeName_serverName`
Cluster	`BPEContainer_clusterName`

4. Click **Stop**.
5. Click **Uninstall > Save > Save**.
6. If the uninstall fails, make sure that all business process applications have been stopped and uninstalled.
7. Using the administrative console, find and remove all or any of the following resources that you do not want to reuse:
   a. Find the process choreographer data source (usually named `BPEDataSource`*dbType*) and note its associated authentication data alias (if any) and JNDI name (usually, `jdbc/BPEDB`) before removing it. If using Oracle there is a second data source named `BPEDataSourceOracleNonXA` which you must remove as well.
   b. If the above data source had an authentication data alias, remove it. Usually, it is named *cellName*`/BPEAuthDataAlias`*dbType_identifier*. Where *cellName* is the name of the cell, *dbType* is the database type, and *identifier* is one of the values given in the previous table.
   c. Remove the JDBC provider of the above data source unless it contains further data sources that you still need.
   d. Remove the process choreographer CMP connector factory, usually named `BPEDataSource`*dbType*`_CF`.
   e. Find the scheduler configuration for the data source JNDI name that you noted in step 7a and remove it as well as the associated work manager.
   f. On the process container configuration page, note the listener port names of the following:
      - `inputListenerPort` - normally has the value `BPEInternalListenerPort`
      - `externalRequestListenerPort` - normally has the value `BPEApiListenerPort`
      - `holdListenerPort` - normally has the value `BPEHoldListenerPort`
   g. In the application server configuration, locate and note their queue and queue connection factory JNDI names, these are normally:

- jms/BPEIntQueue
- jms/BPEApiQueue
- jms/BPEHldQueue
- jms/BPECF
- jms/BPECFC

h. Remove the three listener ports.

i. On the process container configuration page, note the retention queue factory name (normally jms/BPECF).

j. In the following, note the associated authentication data aliases (if any) before removing the queue connection factories:

   1) When using WebSphere MQ, remove all queues and queue connection factories in the WebSphere MQ JMS provider whose JNDI names match the ones noted in the previous steps.

   2) When using embedded messaging, note the associated queue names of all queues before removing them, they are normally:

      - BPEIntQueue_*serverName*
      - BPEApiQueue_*serverName*
      - BPEHldQueue_*serverName*
      - BPERetQueue_*serverName*

k. Remove any authentication data aliases noted in the previous step.

l. In a cluster, repeat the removal of any other server level resources.

m. Save your configuration changes.

n. Restart the application server.

The business process container has been uninstalled, but the process choreographer database and queue manager have not been removed. Resources related to the business process container (such as scheduler, data sources, listener ports, connection factories, and queue destinations) that you have not removed can either be uninstalled separately, or reused.

Chapter 5. Configuring the staff service for process choreographer

Process choreographer uses staff plug-ins to determine who can start a process or claim an activity. Your business processes can also use the staff plug-in services to resolve staff queries. Each type of directory service requires a different staff plug-in. You can register multiple staff plug-ins. The user registry and system plug-ins are already installed and can be used without any configuration. To configure a staff plug-in provider:

1. In the administrative console, click **Resources > Staff Plugin Provider**. The system plug-in and the user registry plug-in require no customization and are ready to use. The preconfigured Lightweight Directory Access Protocol (LDAP) staff plug-in configuration assumes that the LDAP server is on the same host as the application server.
2. To create a new LDAP configuration:
   a. Click the name of the LDAP staff plug-in provider.
   b. Select **Staff Plugin Configuration**.
   c. Click **New**.
   d. Click **Browse**, and select the sample Extensible Style Language (XSL) transformation file to use. The standard XSL transformation for LDAP is located:
      - on Windows systems, in *install_root*\ProcessChoreographer\Staff\LDAPTransformation.xsl
      - On UNIX systems in *install_root*/ProcessChoreographer/Staff/LDAPTransformation.xsl

      Do not modify this transformation file. If you need to customize the transformations to match the LDAP schema of your organization, modify a copy that has a different file name.
   e. Click **Next**.
   f. Enter an administrative name for the staff plug-in provider.
   g. Enter a description.
   h. Enter the Java Naming and Directory Interface (JNDI) name for business processes to use in referencing this plug-in, for example, bpe/staff/ldapserver1
   i. Click **Apply**.
   j. Click **Custom Properties**.
   k. For each of the required properties and for any optional properties that you want to set, click the name of the property, enter a value, and click **OK**.
   l. To apply the changes, click **Save**. This table describes each property for the LDAP plug-in.

LDAP plug-in property	Required or optional	Comments

AuthenticationAlias	Optional	The authentication alias used to connect to LDAP, for example, `mycomputer/My LDAP Alias`. You must define this alias in the administrative console by clicking **Security > JAAS > Configuration JAAS Configuration > J2C Authentication Data**. If this alias is not set, anonymous logon to the LDAP server is used.
AuthenticationType	Optional	If the AuthenticationType property is not set, the default logon is anonymous authentication. In all other cases, the default is simple authentication.
BaseDN	Required	The base distinguished name (DN) for all LDAP search operations, for example, `"o=mycompany, c=us"`
ContextFactory	Required	Sets the Java Naming and Directory Interface (JNDI) context factory, for example, `com.sun.jndi.ldap.LdapCtxFactory`
ProviderURL	Required	This Web address must point to the LDAP JNDI directory server and port. The format must be in normal JNDI syntax, for example, `ldap://localhost:389`
SearchScope	Required	The default search scope for all search operations. Determines how deep to search beneath the baseDN property. Specify one of the following values: `objectScope`, `onelevelScope`, or `subtreeScope`
additionalParameterName1-5 and additionalParameterValue1-5	Optional	Use these name-value pairs to set up to five arbitrary JNDI properties for the connection to the LDAP server.

3. To activate the plug-in, stop and start the server.
4. In case of problems, refer to troubleshooting the staff service and staff plug-ins.

Processes can now use the staff support services to resolve staff queries, and to determine which activities can be performed by certain people.

Depending on the queries that you want to create and your directory structure, you might need to create your own transformations. For more information about this topic, see about the staff service in process choreographer.

About the staff service in process choreographer

With process choreographer you can separate your business process logic from the staff resolution. Staff queries are resolved using a plug-in that is specific to the directory service. Some examples of using the staff service are described here:

- Staff queries using the staff support service
- Staff query verb set
- Explicit staff assignments using the system staff plug-in
- Staff queries transformed for the user registry staff plug-in
- Staff queries transformed for the Lightweight Directory Access Protocol (LDAP) staff plug-in

For detailed information on the staff resolution plug-ins, refer to the *Process Choreographer: Staff Resolution Architecture* and the *Process Choreographer: Programming Model for Staff Resolution* White papers in DeveloperWorks WebSphere

Staff queries using the staff support service

You can use WebSphere Studio Application Developer Integration Edition to define staff queries for the staff support service. Staff queries are templates that define how to retrieve the list of users authorized for a certain work item. You can use the abstract query templates (staff verbs) in the WebSphere Studio process editor when you model a business process. These staff verbs are transformed during deployment into a set of queries that can be run on a staff repository.

Before the initial staff query verbs in the process editor and the parameterized verbs in the process models can be run as queries on a specific staff repository, they must be translated into executable queries using an XSL transformation. The result of a transformation (mapping) can be run by staff resolution plug-ins that provide access to specific directories, such as LDAP or the user registry. At run time, the plug-in runs the query by invoking the staff repository APIs and by creating the list of authorized user IDs for the corresponding work item.

During process deployment, the staff support service is invoked to deploy the staff query. The staff service retrieves the staff plug-in provider configuration with the corresponding Java Naming and Directory Interface (JNDI) name. The JNDI name is set in WebSphere Studio, the default is the user registry. The staff support service then converts the parameterized staff query verb into a query run by a specific staff resolution plug-in defined in the configuration for the Extensible Stylesheet Language transformation (XSLT) verb mapping file and the staff resolution plug-in. All staff query verbs belonging to a process template must use the same provider configuration.

The following example illustrates a snippet for a V5.0-style process as generated by the process editor to retrieve the manager of the employee that started the process:

```
<wf:staff type="staffSupportService">
  <staff:verb>
    <staff:id>Manager of Employee by user ID</staff:id>
    <staff:name>Manager of Employee by user ID</staff:name>
    <staff:parameter id="EmployeeUserID">
      %wf:process.starter%
    </staff:parameter>
  </staff:verb>
</wf:staff>
```

Staff query verb set

The staff support service accepts queries in an abstract form that is independent of the directory infrastructure used. The process editor has a set of predefined staff query verbs. The individual staff resolution plug-ins and the XSLT mapping files do not support all the verbs. The Manager of Employee verb, for example, is not available if you use the user registry or the system plug-in.

The staff query verbs are contained in the `VerbSet.xml` file. This file is located in:

- On UNIX systems and z/OS in *install_root*`/ProcessChoreographer/Staff/`
- On Windows systems in *install_root*`\ProcessChoreographer\Staff\`

You can modify this set of staff query verbs. Make your changes to a copy of the file. Ensure that the copied file has a different file name.

The following predefined set of verbs is available. For information on the parameters that you can use with each of the verbs, see Predefined staff verbs and their parameters.

Department Members
Use this verb to define a query to retrieve the members of a department. The retrieved users belong to any of the specified departments (DepartmentName, AlternativeDepartmentName1, or AlternativeDepartmentName2). This verb is supported by the LDAP plug-in.

Everybody
Use this verb to assign a work item to every user authenticated by the WebSphere Application Server. This verb is supported by the system, user registry, and LDAP plug-ins.

Group Members
Use this verb to define a query to retrieve the members of a group. The retrieved users belong to any of the specified groups (GroupName, AlternativeGroupName1, or AlternativeGroupName2). This verb is supported by the user registry and LDAP plug-ins.

Group Search
Use this verb to search for a group based on an attribute match and to retrieve the members of the group. This verb is supported by the user registry and LDAP plug-ins.

Manager of Employee
Use this verb to retrieve the manager of a person using the person's name. This verb is supported by the LDAP plug-in. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Manager of Employee by user ID
Use this verb to retrieve the manager of a person using the person's user ID. This verb is useful in combination with context queries. This verb is supported by the LDAP plug-in. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Native Query
Use this verb to define a native query based on directory-specific parameters. This verb is supported by the user registry and LDAP plug-ins. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Nobody
Use this verb to deny normal users access to the work item; only the process administrator and the process choreographer system administrator have access. This verb is supported by the system, user registry, and LDAP plug-ins.

Person Search
Use this verb to search for a person based on an attribute match. This verb is supported by the user registry and LDAP plug-ins. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Role Members
Use this verb to retrieve the users associated with a business process role. The retrieved users belong to any of the specified roles (RoleName, AlternativeRoleName1, or AlternativeRoleName2). This verb is supported by the LDAP plug-in. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Users Use this verb to define a staff query for a user who is known by name. It is not recommended that you hard code user names in process templates. This verb is useful for testing processes. This verb is supported by the

system, user registry, and LDAP plug-ins. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Users by user ID

Use this verb to define a staff query for a user whose user ID is known. Even though it is not recommended that you hard code user IDs in process templates, this verb is useful in combination with context queries, for example:

```
User [username='%wf:process.starter%']
```

This verb is useful for testing processes. This verb is supported by the system, user registry, and LDAP plug-ins. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Explicit staff assignments using the system staff plug-in

The system staff plug-in does not require any configuration parameters; it is preinstalled and ready for use. Using this plug-in, you can hard code user names in your business process. This practice is not normally recommended, but it can be useful for testing if you are using context variables or special queries, as illustrated by the following examples:

```
<sur:staffQueries>
    <staff:userID name="%wf:process.starter%">
    <staff:userID name="%wf:process.administrators%">
</sur:staffQueries>
```

```
<sur:staffQueries>
    <staff:everybody>
</sur:staffQueries>
```

```
<sur:staffQueries>
    <staff:nobody>
</sur:staffQueries>
```

The following XML snippet explicitly retrieves the WebSphere user ID `smith`:

```
<staff:staffQueries>
    <staff:userID name="smith">
</staff:staffQueries>
```

Staff queries transformed for the user registry staff plug-in

Using this plug-in, staff queries can refer to users and groups that are known to the WebSphere Application Server user registry. This plug-in does not require any configuration parameters, comes preinstalled and is ready for use.

The following XML snippet example retrieves the user IDs of all the members of the Administrators group, and all users whose names begin with `Mi`:

```
<sur:staffQueries>
    <sur:usersOfGroup groupName="Administrators"/>
    <sur:search type="user" name="Mi*"/>
</sur:staffQueries>
```

This XML snippet is an example of the transformation output.

Staff queries transformed for the LDAP staff plug-in

If you want to use the LDAP plug-in, you probably need to create a customized version of the LDAP XSL transformation to match the LDAP schema of your organization. The standard XSLT provided for LDAP is located:

- On UNIX systems and z/OS in the *install_root*/ProcessChoreographer/Staff/LDAPTransformation.xsl file
- On Windows systems in the *install_root*\ProcessChoreographer\Staff\LDAPTransformation.xsl file

Make your changes to a copy that has a different file name. When you have made your changes:

1. Configure a staff plug-in provider that points to the new transformation file.
2. Ensure that when you deploy the EAR file, you use the JNDI name of the new staff plug-in provider.

The following XML snippet illustrates the results of the LDAP transformation for a search:

```
<sldap:staffQueries>
  <sldap:search baseDN="ou=mydivision, o=mycompany, c=us" filter="cn=*"
              searchScope="onelevelScope" recursive="no">
     <sldap:attribute name="uid" objectclass="person" usage="simple"/>
  </sldap:search>
</sldap:staffQueries>
```

Predefined staff verbs and their parameters

You can use staff verbs in the WebSphere Studio Application Developer Integration Edition process editor to model staff assignments in a business process. These staff verbs are transformed during modeling and deployment into a set of queries that can be run on a staff repository. The parameters for the following predefined staff verbs are listed here:

- Department Members
- Everybody
- Group Members
- Group Search
- Manager of Employee
- Manager of Employee by user ID
- Native Query
- Nobody
- Person Search
- Role Members
- Users
- Users by user ID

Department Members

Use this verb to define a query to retrieve the members of a department.

Parameter	Use	Type	Supported by	Description
DepartmentName	Mandatory	string	LDAP	Department name of the users to retrieve.
IncludeNestedDepartments	Mandatory	boolean	LDAP	Specifies whether nested departments are considered in the query.

Parameter	Use	Type	Supported by	Description
Domain	Optional	string	LDAP	The domain to which the department belongs. Use this parameter to limit the query to a subset of the directory.
AlternativeDepartmentName1	Optional	string	LDAP	An alternative department to which the users can belong.
AlternativeDepartmentName2	Optional	string	LDAP	An alternative department to which the users can belong.

Everybody

Use this verb to assign a work item to every user authenticated by the WebSphere Application Server. This verb has no parameters; it is supported by the system, user registry, and LDAP plug-ins.

Group Members

Use this verb to define a query to retrieve the members of a group.

Parameter	Use	Type	Supported by	Description
GroupName	Mandatory	string	User registry, LDAP	Group name of the users to retrieve.
IncludeSubgroups	Mandatory	boolean	LDAP	Specifies whether nested subgroups are considered in the query.
Domain	Optional	string		The domain to which the group belongs. Use this parameter to limit the query to a subset of the directory.
AlternativeGroupName1	Optional	string	User registry, LDAP	An alternative group to which the users can belong.
AlternativeGroupName2	Optional	string	User registry, LDAP	An alternative group to which the users can belong.

Group Search

Use this verb to search for a group based on an attribute match and to retrieve the members of the group.

Parameter	Use	Type	Supported by	Description
GroupID	Optional	string	User registry, LDAP	The group ID of the users to retrieve.

Parameter	Use	Type	Supported by	Description
Type	Optional	string	LDAP	The group type of the users to retrieve.
IndustryType	Optional	string	LDAP	The industry type of the group to which the users belong.
BusinessType	Optional	string	LDAP	The business type of the group to which the users belong.
GeographicLocation	Optional	string	LDAP	An indication of where the users are located.
Affiliates	Optional	string	LDAP	The affiliates of the users.
DisplayName	Optional	string	LDAP	The display name of the group.
Secretary	Optional	string	LDAP	The secretary of the users.
Assistant	Optional	string	LDAP	The assistant of the users.
Manager	Optional	string	LDAP	The manager of the users.
BusinessCategory	Optional	string	LDAP	The business category of the group to which the users belong.
ParentCompany	Optional	string	LDAP	The parent company of the users.

Manager of Employee

Use this verb to retrieve the manager of a person using the person's name.

Parameter	Use	Type	Supported by	Description
EmployeeName	Mandatory	string	LDAP	The name of the employee whose manager is retrieved.
Domain	Optional	string		The domain to which the employee belongs. Use this parameter to limit the query to a subset of the directory.

Manager of Employee by user ID

Use this verb to retrieve the manager of a person using the person's user ID.

Parameter	Use	Type	Supported by	Description
EmployeeUserID	Mandatory	string	LDAP	The user ID of the employee whose manager is retrieved. Supports context variables, such as `%wf:process.starter%`

Parameter	Use	Type	Supported by	Description
Domain	Optional	string		The domain to which the employee belongs. Use this parameter to limit the query to a subset of the directory.

Native Query

Use this verb to define a native query based on directory-specific parameters.

Parameter	Use	Type	Supported by	Description
QueryTemplate	Mandatory	string	User registry, LDAP	The query template to use for the query. The default mapping files for the user registry and LDAP plug-ins support the templates search, user, and usersOfGroup.
Query	Mandatory	string	User registry, LDAP	Specifies the query. You can use context variables, such as `%wf:process.starter%`. The type of query depends on the plug-in and the query template. **User registry** • search template: search pattern • user template: user name • usersOfGroup: group name **LDAP** • search template: search filter • user template: user dn • usersOfGroup: group dn

Parameter	Use	Type	Supported by	Description
AdditionalParameter1	Optional	string	User registry, LDAP	Specifies the query. You can use context variables, such as `%wf:process.starter%`. The type of parameter depends on the plug-in and the query template. **User registry** • search template. Used for the search type. Supported values: `group` and `user`. • user template. Not supported • usersOfGroup. Not supported **LDAP** • search template. Used to specify whether recursive search is done. Supported values: `yes` and `no` • user template. Not supported • usersOfGroup. Used to specify whether recursive search is done. Supported values: `yes` and `no`
AdditionalParameter2	Optional	string	User registry, LDAP	Use this verb to specify an additional parameter.
AdditionalParameter3	Optional	string	User registry, LDAP	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.
AdditionalParameter4	Optional	string	User registry, LDAP	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.
AdditionalParameter5	Optional	string	User registry, LDAP	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.

Nobody

Use this verb to deny normal users access to the work item; only the process administrator and the process choreographer system administrator have access. This verb has no parameters. and is supported by the system, user registry, and LDAP plug-ins.

Person Search

Use this verb to search for people based on an attribute match.

Parameter	Use	Type	Supported by	Description
UserID	Optional	string	User registry, LDAP	The user ID of the users to retrieve.
Profile	Optional	string	LDAP	The profile of the users to retrieve.
LastName	Optional	string	LDAP	The last name of the users to retrieve.
FirstName	Optional	string	LDAP	The first name of the users to retrieve.
MiddleName	Optional	string	LDAP	The middle name of the users to retrieve.
Email	Optional	string	LDAP	The e-mail address of the users.
Company	Optional	string	LDAP	The company to which the users belong.
DisplayName	Optional	string	LDAP	The display name of the users.
Secretary	Optional	string	LDAP	The secretary of the users.
Assistant	Optional	string	LDAP	The assistant of the users.
Manager	Optional	string	LDAP	The manager of the users.
Department	Optional	string	LDAP	The department to which the users belong.
Phone	Optional	string	LDAP	The telephone number of the users.
Fax	Optional	string	LDAP	The fax number of the users.
Gender	Optional	string	LDAP	Whether the user is male or female.
Timezone	Optional	string	LDAP	The time zone in which the users are located.
PreferredLanguage	Optional	string	LDAP	The preferred language of the user.

Role Members

Use this verb to retrieve the users associated with a business process role.

Parameter	Use	Type	Supported by	Description
RoleName	Mandatory	string	LDAP	Role name of the users to retrieve.
IncludeNestedRoles	Mandatory	boolean	LDAP	Specifies whether nested roles are considered in the query.

Parameter	Use	Type	Supported by	Description
Domain	Optional	string		The domain to which the role belongs. Use this parameter to limit the query to a subset of the directory.
AlternativeRoleName1	Optional	string	LDAP	An alternative role name for the user.
AlternativeRoleName2	Optional	string	LDAP	An alternative role name for the user.

Users

Use this verb to define a staff query for a user who is known by name.

Parameter	Use	Type	Supported by	Description
Name	Mandatory	string	System, user registry, LDAP	The name of the user to retrieve.
AlternativeName1	Optional	string	System, user registry, LDAP	An alternative user name. Use this parameter to retrieve more than one user.
AlternativeName2	Optional	string	System, user registry, LDAP	An alternative user name. Use this parameter to retrieve more than one user.

Users by user ID

Use this verb to define a staff query for a user whose user ID is known.

Parameter	Use	Type	Supported by	Description
UserID	Mandatory	string	System, user registry, LDAP	The user ID of the user to retrieve.
AlternativeID1	Optional	string	System, user registry, LDAP	An alternative user ID. Use this parameter to retrieve more than one user.
AlternativeID2	Optional	string	System, user registry, LDAP	An alternative user ID. Use this parameter to retrieve more than one user.

Troubleshooting the staff service and the staff plug-ins

One of the following situations might be caused by a problem with the staff service or a staff plug-in:

- Stopped staff activities
- Changes to the staff repository that are not immediately reflected in work-item assignments

Use this overview task to help resolve the problem. You can also go to the technical support search page, to look for additional information.

Stopped staff activities

You encountered one or more of the following problems:

- Work items resulting from staff activities cannot be claimed although the business process started navigating successfully.
- The SystemOut.log file contains the following message: `BPEE0057I: Activity 'MyStaffActivity' of processes 'MyProcess' has been stopped because of an unhandled failure...`

This message indicates that WebSphere Application Server security might not be enabled. Staff activities require that security is enabled and the user registry is configured. Take the following steps:

1. Check that WebSphere security is enabled. In the administrative console, go to **Security > Global Security**.
2. Check that the user registry is configured. In the administrative console, go to **Security > User Registries**.

Changes to the staff repository that are not immediately reflected in work-item assignments

For example, you added the user, Frank, to the staff repository, but Frank has not received any work items, although he is eligible for them.

This problem can occur when the cached staff query results for a process template expire. To optimize the staff query resolution performance, the retrieved query results are cached. These results are shared for all process instances of a process template if the content of the context variables is the same for all query instances. The cache content is checked for currency when a new process instance is created or the corresponding staff activity gets scheduled. By default, the time after which the shared staff query results expire is one hour.

To change the default value, modify the `StaffQueryResultValidTimeSeconds` variable in the `bpe.properties` file. This file is located:

- On UNIX system and z/OS, in the *install_root*`/properties` directory
- On Windows systems, in *install_root*`\properties` directory

You might need to create the `bpe.properties` file. For example, to set the expiry time to one minute, change the variable to:

```
StaffQueryResultValidTimeSeconds=60
```

Staff service settings

Use this page to enable or disable the staff service, which manages staff plug-in resources used by the server.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Staff Service**.

Startup

Specifies whether the server attempts to start the staff service.

Default Selected

Range

Selected
When the application server starts, it attempts to start the staff service automatically.

Cleared
The server does not try to start the staff service. If staff plug-in resources are used on this server, the system administrator must start the staff service manually or select this startup property and then start the server again.

Staff plugin provider collection

Use this page to manage staff plug-in providers.

A staff plug-in is responsible for the retrieval of user information.

To view this administrative console page, click **Resources > Staff Plugin Providers**.

Name

The name by which the staff plug-in provider is known for administrative purposes.

Data type String

Description

A description of the staff plug-in provider.

Data type String

Staff plugin provider settings

Use this page to modify staff plug-in provider settings.

Staff plug-ins are responsible for the retrieval of user information. Each staff plug-in provider is registered with the run-time environment by specifying a name and a Java archive (JAR) file containing the plug-in. A configuration file in the JAR file defines the class name, which represents the plug-in as well as the properties for the plug-in.

To view this administrative console page, click **Resources > Staff Plugin Providers > *staffpluginprovider_name***.

To inspect or change the staff plugin configuration and any custom properties, click on the link in the **Additional Properties** section.

Name

The name by which the staff plug-in provider is known for administrative purposes.

Data type String

Description
A description of the staff plug-in provider.

Data type String

Jar File
The file name, including the absolute path, of the JAR file containing the plug-in.

Data type String

Staff plugin configuration collection
Use this page to manage staff plug-in configurations.

A staff plug-in configuration is defined for a staff plug-in provider. The staff plug-in configuration can define any custom properties specified by the staff plug-in provider. Each staff plug-in provider can have multiple staff plug-in configurations.

To view this administrative console page, click **Resources > Staff Plugin Providers > *staffpluginprovider_name* > Staff Plugin Configuration**.

Name:

The name by which the staff plug-in configuration is known for administrative purposes.

Data type String

Description:

A description of the staff plug-in configuration.

Data type String

JNDI Name:

The Java Naming and Directory Interface (JNDI) name used to look up the staff plug-in configuration in the namespace.

Data type String

XSL Transform File:

The file name, including the absolute path, of the Extensible Style Language (XSL) transformation file.

Data type String

Staff plugin configuration settings:

Use this page to modify staff plug-in configuration settings.

To view this administrative console page, click **Resources > Staff Plugin Providers > *staffpluginprovider_name* > Staff Plugin Configuration > *staffpluginconfiguration_name***.

Name:

The name by which the staff plug-in configuration is known for administrative purposes.

Data type String

Description:

A description of the staff plug-in configuration.

Data type String

JNDI Name:

The Java Naming and Directory Interface (JNDI) name used to look up the staff plug-in configuration in the namespace.

Data type String

XSL Transform File:

The file name, including the absolute path, of the Extensible Style Language (XSL) transformation file.

Data type String

Chapter 6. Administering process choreographer

You can use the administrative console to administer some aspects of process choreographer, such as the compensation service or querying and replaying failed messages. Scripts are provided for other administrative tasks, such as deleting audit log entries.

You can use the administrative console to do the following administrative tasks:

- Administer the compensation service for a server.
- 5.1+ Query and replay failed messages.

You can use scripts to do the following administrative tasks:

- Query and replay failed messages.
- Delete audit log entries.
- Refresh staff entries that are cached in the database.
- Remove unused staff entries that are cached in the database.
- Add tables to the run-time database.

Administering the compensation service for a server

If the compensation service is installed and started on an application server, applications can use compensation in business processes to commit updates in several related transactions before the process completes.

You can use the administrative console to view and change properties of the compensation service for application servers.

1. Display the administrative console.
2. In the navigation pane, click **Servers > Application Servers >** *server_name*
3. In the content pane, under Additional Properties, click **Compensation Service** This action displays a panel with the compensation service properties.

   Startup
   Select this check box to enable the application server to automatically try to start the compensation service when the server starts up again.

   Recovery log directory
   Specifies the name of a directory for this server where the compensation service stores log files for recovery. When compensation is used, the WebSphere product stores information that is required for compensation.

   Recovery Log File Size
   Specifies the maximum size, in megabytes, for compensation log files on this application server.

   For more information about these properties, see "Compensation service settings" on page 110.
4. Optional: If needed, change the compensation service properties.
5. Click **OK**.
6. To save your configuration, click **Save** on the task bar of the administrative console window.

7. To have the changed configuration take effect, stop then restart the application server.

Compensation service settings

Use this page to modify compensation service settings.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Compensation Service**.

Startup

Specifies whether the server attempts to start the compensation service when the server next starts up.

Default Selected

Range

Selected
When the application server starts, it attempts to start the compensation service automatically.

Cleared
The server does not try to start the compensation service. If compensation is used in applications that run on this server, the system administrator must start the service manually or select this property and then restart the server.

Recovery log directory

Specifies the name of a directory for this server where the compensation service stores log files for recovery.

A blank value in the server configuration is expanded by the compensation service at startup as the directory *install_root*/recoveryLogs/*server_name*.

When compensation is used, the WebSphere product stores information that is needed to perform compensation after a system failure on a physical storage device. In a higher application load, this persistence slows down performance of the application server due to its dependency on the operating system and the underlying storage systems.

To achieve better performance, move the compensation log files to a storage device with more physical disk drives, or preferably RAID disk drives. When the log files are moved to the file systems on the raided disks, the task of writing data to the physical media is shared across the multiple disk drives. This sharing provides more concurrent access to persist compensation information and faster access to that data from the logs. Depending upon the design of the application and storage subsystem, performance gains can range from 10% to 100%, or even more in some cases.

This change is applicable only to the configuration where the application has compensation configured. Consider setting this property when the application server shows one or more of following signs:

- CPU utilization remains low despite an increase in compensatable requests
- Transactions fail with several timeouts
- The server stops and needs to be restarted
- The disk on which the server is running shows higher use

Data type	String
Default	*install_root*/recoveryLogs/*server_name*
Recommended	Create a file system with at least 3 to 4 disk drives raided together in a RAID-0 configuration. Create the compensation log on this file system with the default size. When the server is running under load, check the disk input and output. If the disk input and output time is more then 5%, consider adding more physical disks to lower the value. If the disk input and output is low, but the server load is still high, consider increasing the size of the log files.

Recovery Log File Size

Specifies the maximum size, in megabytes, for compensation log files on this application server.

The amount of data logged by the compensation service is influenced by the number of concurrently-active applications that use compensation, and the size of the application data that is provided as input to the compensation logic.

The compensation service reserves space on a physical storage device at server start for compensation recovery data. If this size is not sufficient for active compensation data, then the log files grow dynamically until they reach the maximum size specified by this value.

Data type	Integer
Units	Megabytes
Default	5
Range	1 through 2147483647 megabytes

Although the allowed range for Recovery Log File Size is 1 through 2147483647, the maximum size that is actually possible depends on the maximum size allowed by the operating system for a mapped file.

Querying and replaying failed messages

When a problem occurs processing an internal message, this message ends up in the retention queue or hold queue. This task describes how to determine if any failed messages exist, and to send them to the internal queue again.

1. To check how many messages are on the hold and retention queues:
   a. Click **Servers > Application Servers >***server_name* **> Business Process Container**
   b. On the **Configuration** tab, in the **Additional Properties** section, click **Replay Messages**. The number of messages on the hold queue and retention queue are displayed in the **General Properties** table.
2. If either the hold queue or the retention queue contains messages, you can move the messages to the internal work queue by clicking either **Replay Hold Queue**, or **Replay Retention Queue**.

The business process engine of the process choreographer tries to service all replayed messages again.

Business process container replay messages

Use the replay functions to move failed messages from the retention or hold queues back to the internal work queue.

To view this administrative console page, click **Servers > Application Servers >***server_name* **> Business Process Container > Replay Messages**. When messages cannot be processed, they are placed in the retention queue for retrying unless the message has reached the retry limit, in which case, the message is moved to the hold queue. Use the Replay messages panel to check how many messages are in each of these queues. You can replay the messages in either queue, which moves them to the internal work queue again.

Refresh Message Counts

Click this button to refresh the displayed count of how many messages are on the hold and retention queues.

Replay Hold Queue

Click this button to move all the messages from the hold queue to the internal work queue.

Replay Retention Queue

Click this button to move all the messages from the retention queue to the internal work queue.

Hold Queue Messages

This field indicates how many messages are on the hold queue. To update this field click **Refresh Message Counts**.

Data type Integer

Retention Queue Messages

This field indicates how many messages are on the retention queue. To update this field click **Refresh Message Counts**.

Data type Integer

Message Exceptions

This field displays any exceptions that are thrown while messages are being replayed.

Data type String

Process choreographer: Failed message handling and quiesce mode

Process choreographer provides a facility for handling temporary infrastructure failures. This facility consists of two numerical limits, two queues, quiesce mode, and the message retry behavior.

Retry Limit

The Retry Limit defines the maximum number of times that a message can fail before being put on the hold queue. In a performance critical application running

in a reliable infrastructure, the retry limit should be small, for example, one or two. This parameter can be found in the administration console, on the Business Process Container page.

Retention Queue Message Limit

The Retention Queue Message Limit defines the maximum number of messages that can be on the retention queue. If the retention queue overflows, the system goes into quiesce mode. To make the system go into quiesce mode as soon as one message fails, set the value to zero. To make the system more tolerant of infrastructure failures, increase the value. This parameter can be found in the administration console, on the Business Process Container page.

Retention Queue

The retention queue holds failed messages, that will be replayed by moving them back to the business process container's internal work queue. Any messages that have failed a number of times equal to the retry limit the message does not return to the retention queue, but instead, is moved to the hold queue. If the retention queue the retention queue is full to the limit defined by the retention queue message limit, and another message fails, the queue has overflowed, and the system goes into quiesce mode. The administrator can move the messages in this queue back to the internal queue performing the task Querying and replaying failed messages.

Hold Queue

The hold queue is where messages end up when the number of times that they have failed is equal to the retry limit. The administrator can move the messages in this queue back to the internal queue performing the task Querying and replaying failed messages.

Replay Messages

The administrator can move the messages from the hold or retention queues back to the internal queue. This is done using the Replay Messages panel for the business process container.

Quiesce Mode

Quiesce mode is entered when the retention queue overflows. When this happens, it is assumed that there is a serious, though possibly temporary, infrastructure failure. The purpose of quiesce mode is to prevent the system from using a lot of resources while an infrastructure failure means that most messages will probably fail anyway. In quiesce mode, the system sleeps for two seconds before attempting to process the next message. As soon as a message is successfully processed, the system resumes normal message processing.

Using scripts to query and replay failed messages

When a problem occurs processing an internal message, this message ends up on the retention queue or hold queue. To determine if any failed messages exist, and to send them to the internal queue again:

1. Verify that you are using a user ID that has administrative rights.

2. Change to the process choreographer utilities directory where the scripts are located: On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\util
   ```

   On UNIX systems and z/OS, enter:

   ```
   cd install_root/ProcessChoreographer/util
   ```

3. Query the number of failed messages on both the retention and hold queues. Enter one of the following commands:

   ```
   install_root\bin\wsadmin -f queryNumberOfFailedMessages.jacl
                            -cluster clusterName

   install_root\bin\wsadmin -f queryNumberOfFailedMessages.jacl
                            -node nodeName
                            -server serverName
   ```

   Where:

   clusterName
   The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

   nodeName
   Optional when specifying the server name. This name identifies the node. The default is the local node.

   serverName
   The name of the server. Required if the cluster name is not specified.

   If you want to check for a server on the local node, enter:

   ```
   wsadmin -f queryNumberOfFailedMessages.jacl -server serverName
   ```

4. Replay all failed messages on the hold queue, retention queue, or both queues by entering one of the following commands:

   ```
   install_root\bin\wsadmin -f replayFailedMessages.jacl
                        -cluster clusterName
                        -queue replayQueue

   install_root\bin\wsadmin -f replayFailedMessages.jacl
                        -node nodeName
                        -server serverName
                        -queue replayQueue

   install_root\bin\wsadmin -f replayFailedMessages.jacl
                        -server serverName
                        -queue replayQueue
   ```

   Where:

   replayQueue
   Must have one of the following values: `holdQueue`, `retentionQueue`, or `both`.

   clusterName
   The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

   nodeName
   Optional when specifying the server name. This name identifies the node. The default is the local node.

   serverName
   The name of the server. Required if the cluster name is not specified.

Deleting audit log entries

You can use the `deleteAuditLog.jacl` script to delete audit log entries from the database.

1. Change to the process choreographer utilities directory where the scripts are located. On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\util
   ```

   On UNIX systems and z/OS, enter:

   ```
   cd install_root/ProcessChoreographer/util
   ```

2. Delete the entries in the audit log table. Enter one of the following commands:

   ```
   install_root\bin\wsadmin -f deleteAuditLog.jacl
                            -server serverName [options]

   install_root\bin\wsadmin -f deleteAuditLog.jacl
                            -node nodeName
                            -server serverName [options]

   install_root\bin\wsadmin -f deleteAuditLog.jacl
                            -cluster clusterName [options]
   ```

   Where:

   clusterName
   The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

   nodeName
   Optional when specifying the server name. This name identifies the node. The default is the local node.

   serverName
   The name of the server. Required if the cluster name is not specified.

   The available options are:

   -all Deletes all the audit log entries in the database. The deletion is done in multiple transactions. Each transaction deletes the number of entries specified in the slice parameter.

   -slice Used with the **-all** parameter to specify the number of entries included in each transaction. The value depends on the available log size for your database system. Higher values require fewer transactions but you might exceed the database log space. Lower values might cause the script to take longer to complete the deletion. The default size for the **-slice** parameter is 250.

   -time Deletes all the audit log entries that are older than the time you specify for this parameter. The time format must be: `YYYY-MM-DD['T'HH:MM:SS]`. If you specify only the year, month, and day, the hour, minutes, and seconds are set to `00:00:00`.

   -processtime
   Deletes all the audit log entries that belong to a process that finished before the time you specify for this parameter. Use the same time format as for the **-time** parameter.

Refreshing staff entries that are cached in the database

Verify that your user ID has administrative rights.

Process choreographer caches the results of staff assignments evaluated against a staff directory, such as an Lightweight Directory Access Protocol (LDAP) server, in the run-time database. If the staff directory changes, you can force the staff assignments to be evaluated again.

1. Change to the process choreographer utilities directory where the scripts are located. On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\util
   ```

   On UNIX systems and z/OS, enter:

   ```
   cd install_root/ProcessChoreographer/util
   ```

2. Force the staff assignment to be evaluated again. Enter one of the following commands:

   ```
   install_root\bin\wsadmin -f refreshStaffQuery.jacl
                            -server serverName
                            [-template templateName]

   install_root\bin\wsadmin -f refreshStaffQuery.jacl
                            -node nodeName
                            -server serverName
                            [-template templateName]

   install_root\bin\wsadmin -f refreshStaffQuery.jacl
                            -cluster clusterName
                            [-template templateName]
   ```

   Where:

   clusterName
   The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

   nodeName
   Optional when specifying the server name. This name identifies the node. The default is the local node.

   serverName
   The name of the server. Required if the cluster name is not specified.

   templateName
   Optional. The name of the process template. Staff assignments that belong to this process template are refreshed.

Removing unused staff entries that are cached in the database

Verify that your user ID has administrative rights.

Process choreographer maintains lists of user names in the run-time database for staff expressions that have been evaluated. Although these processes instances are finished, the lists of user names are maintained in the database until the corresponding business process application is uninstalled.

If the size of the database is affecting performance, you can remove the unused staff lists that are cached in the database tables.

1. Change to the process choreographer utilities directory where the scripts are located. On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\util
   ```

   On UNIX systems and z/OS, enter:

   ```
   cd install_root/ProcessChoreographer/util
   ```

2. Remove the unused staff lists. Enter one of the following commands:

   ```
   install_root\bin\wsadmin -f cleanupUnusedStaffQueryInstances.jacl
                            -server serverName

   install_root\bin\wsadmin -f cleanupUnusedStaffQueryInstances.jacl
                            -node nodeName
   ```

```
                        -server serverName

install_root\bin\wsadmin -f cleanupUnusedStaffQueryInstances.jacl
                        -cluster clusterName
```

Where:

clusterName
The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

nodeName
Optional when specifying the server name. This name identifies the node. The default is the local node.

serverName
The name of the server. Required if the cluster name is not specified.

The number of entries deleted from the database is displayed.

Adding tables to the run-time database

Verify the following:

- Your user ID has administrative rights.
- The user ID you configured to access the database has the rights to run the statements in the DDL file.
- The DDL file is accessible from the application server.

If you need to add tables to the process choreographer run-time database, you can run a script that runs a DDL file on the database.

1. Change to the process choreographer utilities directory where the scripts are located. On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\util
   ```

   On UNIX systems and z/OS, enter:

   ```
   cd install_root/ProcessChoreographer/util
   ```

2. Add the tables to the database. Enter one of the following commands:

   ```
   install_root\bin\wsadmin -f runDdlFile.jacl
                           -server serverName
                           -filename ddlFile

   install_root\bin\wsadmin -f runDdlFile.jacl
                           -node nodeName
                           -server serverName
                           -filename ddlFile

   install_root\bin\wsadmin -f runDdlFile.jacl
                           -cluster clusterName
                           -filename ddlFile
   ```

   Where:

   clusterName
   The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

   nodeName
   Optional when specifying the server name. This name identifies the node. The default is the local node.

   serverName
   The name of the server. Required if the cluster name is not specified.

ddlFile

The name of a DDL file that is local to the application server. Staff assignments that belong to this process template are refreshed.

Chapter 7. Managing processes

Verify that the business process container is installed and configured for each application server that uses process choreographer.

The way in which you install and uninstall business process application depends on whether the process is a process based on the Business Process Execution Language (BPEL) or a V5.0-style process.

Managing BPEL-based processes involves the following actions:

- 5.1+ Installing BPEL-based process applications.
- 5.1+ Installing BPEL-based process applications WebSphere Studio did not generate.
- Stopping and starting BPEL-based process templates.
- 5.1+ Uninstalling BPEL-based process applications.

Managing V5.0-style processes involves the following actions:

- Installing V5.0-style process applications.
- Stopping and starting process V5.0-style templates.
- Uninstalling V5.0-style process applications.

Installing BPEL-based process applications

To install an enterprise application based on the Business Process Execution Language (BPEL):

1. Verify that the business process container of the application server is installed and configured for each application server that uses process choreographer.
2. Make sure that the application server is running. In this respect, process applications are different from other Java 2 Platform Enterprise Edition (J2EE) applications.
   - In a Network Deployment (ND) environment, the ND application server, all ND-managed stand-alone application servers, and at least one application server must be running for each cluster where you install the application.
   - If you use the wsadmin tool to install applications, make sure that you are connected to a process. In an ND environment, the process is the deployment manager; in a stand-alone environment, it is the application server. When you run the wsadmin tool, do not use the `-conntype NONE` option as one of the installation options.
3. Install your application on your application server.
   a. Click **Applications > Install New Applications** in the administrative console navigation pane.
   b. Click **Browse** and select the enterprise archive (EAR) file that you want to install.
   c. Click **Next**.
   d. In the **Provide options to perform EJB deploy** step, make sure that you select the database system that is used for the business process container where the application is to be installed. This database system must be the

same as the one that is used for the data sources for the business process container. If you omit this step, it can result in errors during installation of the business process application.

If you use WebSphere Studio to build your business process application, database mapping codes for the Cloudscape database system are created automatically. To run this application outside the Unit Test Environment with a database system other than Cloudscape, you must create additional database mappings before you export the EAR file. If you do not create these mappings, this step offers you only the Cloudscape database system for the Enterprise JavaBeans (EJB) deploy step.

e. In an ND environment, when installing a new application on a cluster named *clusterName*, you must insert the cluster name into the default JNDI names in two steps:
   1) In the **Provide default datasource mapping for modules containing 2.0 entity beans** step, change the **JNDI Name** field for your EJB module from the default value `eis/jdbc/BPEDB_CMP` to `eis/jdbc/BPEDB_`*`clusterName`*`_CMP`.
   2) In the **Map resource references to resources** step, change the **JNDI Name** field for your EJB module from the default value `jdbc/BPEDB` to `jdbc/BPEDB_`*`clusterName`*.

f. Optional: In the **Automatically create tables for Process Entity Beans** step, you can choose to automatically create database tables. This option is available only if the application contains an interruptible process. You must also have the authority to create new tables.

   Make sure that the EAR file contains the required relational database (RDB) mapping for process entity beans for the chosen database system. If you use WebSphere Studio to build your business process application, you can generate these mappings in the J2EE perspective of the EJB module (**Generate > EJB to RDB mappings**). For more information on generating EJB mappings, see the WebSphere Studio information center.

   Restriction for DB2 z/OS V7. The table names and index names created for the process entity bean are truncated to 18 characters. Because the process entity bean name comprises the name and a valid-from time stamp, the length restriction can result in name clashes. When you create the RDB mapping in WebSphere Studio, make sure that the resulting table and index names do not clash with names of existing business processes. The table generated during the mapping specifies neither a database nor a storage group. For a production environment, it is recommended that you export the DDL file, add the appropriate qualifiers, and run the file manually using the command-line tools of the database system.

g. Go to the **Summary** step and click **Finish**.

All process templates are put into the start state. To create process template instances, you must start the process application. If you did not create a database table for the process entity bean in this step, the creation of the process instances fails if the process is an interruptible process.

4. If you did not create tables automatically in the previous step, export the DDL file of the installed application and apply it to the process choreographer database. A `table.dll` file is generated for each module. This file is located in the backend directory of the chosen database type. Depending on where the module is installed, the `table.dll` file must be applied to each of the databases on the server or cluster.
   a. Click **Applications > Enterprise Applications** in the administrative console navigation pane.

b. Select the check box next to the application for which you want to export the DDL file and click **Export DDL**.
c. Click the link to the data definition language (DDL) file and save it to a temporary directory.
d. Apply the DDL files to the database, to the server, or to the cluster where you want to install the module.

Installing BPEL-based process applications that WebSphere Studio did not generate

You can use WebSphere Studio to model business processes and then generate application modules from the process models. You can then use the administrative console to install these applications in process choreographer.

Process choreographer also supports BPEL-based processes that WebSphere Studio did not generate. You must map these process applications to constructs that can be used directly by process choreographer. A script is provided for you to specify options for creating and installing the application. You can provide an options file to use with the script to specify options for the individual processes in the application.

1. Change to the process choreographer directory utilities directory. On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\util
   ```

   On UNIX platforms and z/OS, enter:

   ```
   cd install_root/ProcessChoreographer/util
   ```

2. Run the deploy script. On Windows, enter:

   ```
   install_root\bin\wsadmin -f deployBPEL.jacl [options]
   ```

   On UNIX platforms and z/OS, enter:

   ```
   install_root/bin/wsadmin -f deployBPEL.jacl [options]
   ```

Parameter	Description	Value range	Default
Options for creating EAR files			
-scrdir directory	Source directory that contains the BPEL processs definitions and the associated WSDL and XSD definitions.	(directory)	"." (period)
-options file	BPEL partner link to WSDL port binding table and process deployment settings.	(file name: *.options)	(none)
-earFileName name	Fully-qualified or relative file name of the output EAR file.	(file name: *.ear)	
Options for installing applications in process choreographer			
-activateProcess indicator	Install application in WebSphere Application Server.	`Yes` or `No`	`Yes`

Parameter	Description	Value range	Default
-createTables indicator	Automatically create container-managed persistence (CMP) tables for interruptible processes.	`Yes` or `No` If the database system is DB2 UDB OS/390 Version 7, the only allowed value is `No`.	`Yes`

-database value	Type of database.	`DB2UDB_V81`
		`CLOUDSCAPE_V5`
		`DB2UDBOS390_V7`
		`DB2UDBISERIES`
		`INFORMIX_V93`
		`MSSQLSERVER_2000`
		`ORACLE_V8`
		`ORACLE_V9I`
		`SYBASEV1200`
		`SYBASEV1250`
			`DB2UDB_V81`

-target (name, ...)	Name of the WebSphere deployment target.	server, node, or cluster name	Not needed for a single server.
Binding parameters for the application			
-genericMdbConnection Factory name	Generic process choreographer MDB queue connection factory name.		`jms/BPECF`
-genericMdbQueue name	Generic process choreographer MDB queue name.		`jms/BPEIntQueue`
-jndiInitialContextFactory name	Initial context factory.		(none)
-jndiProviderURL url	JNDI provider URL		(none)

To generate a process application EAR file, for example, enter:

```
wsadmin -conntype NONE
        -f install_root/ProcessChoreographer/util/deployBPEL.jacl
        -srcdir <mySamples>\SimpleProcess
        -earFileName SimpleProcess.ear
        -options interfaces4Partner.opt
        -activateProcess No
```

For this example, the `interfaces4Partner.opt` options file might look like this:

```
*.deployAsEJB=Yes
```

To generate and install a process application, for example, enter:

```
wsadmin -conntype NONE
        -f install_root/ProcessChoreographer/util/deployBPEL.jacl
        -srcdir <mySamples>\SimpleProcess
        -earFileName SimpleProcess.ear
        -options interfaces4Partner.opt
        -database CLOUDSCAPE_V5
```

The `interfaces4Partner.opt` options file is the same as in the previous example.

If you create a process, you can use the administrative console to install and manage the process. If you create and install the process, you can use the administrative console to manage the process template instances.

Options file for deployBPEL.jacl script

When you install BPEL-based process applications that were not generated by WebSphere Studio, you specify options with the `deployBEPL.jacl` script for creating the EJB module and installing the process application.

You can also specify properties for the individual processes in the process application and the relationships between BPEL partner links and Web service end points (WSDL ports). These properties are contained in an options file that you specify with the **-options** parameter when you run the `deployBPEL.jacl` script.

An entry for a process setting in the options file has the following syntax:

```
targetNameSpaceURI(processNCName).Keyword=Value
```

To specify that a property applies to all processes in the process application, you can use an asterisk (*) as a wildcard character, for example:

```
*.Keyword=Value
```

Process properties

Keyword=Value	Description	Value range	Default
Properties for generating the process			
allowOptimizations= indicator	Allow run-time optimizations.	`Yes` or `No`	`No`
deployAsEJB= indicator	Generate deploy code for EJB	`Yes` or `No`	`No`
deployAsJMS= indicator	Generate deploy code for JMS	`Yes` or `No`	`No`
BPEL file properties that you can overwrite for process choreographer			
validFrom= value	Process valid from timestamp.	yyyymmddThhmmss	(none)
staffProvider	Process staff provider name.	(global JNDI name)	`bpe/staff/ userregistryconfiguration`

Syntax of relationships between partner links and end points

- For inbound services:

```
targetNameSpaceURI(processNCName).partnerLinkNCName.myEndpoint=
      serviceQName(portNCName1)
```

- For outbound services:

```
targetNameSpaceURI(processNCName).partnerLinkNCName.partnerEndpoint=
      serviceQName(portNCName2)
```

Stopping and starting BPEL-based process templates

Verify that you are using a console user ID that has either the `operator` or the `administrator` role. If you are stopping a process, verify that no process instances are running. If necessary, a process administrator can use the process choreographer Web client to stop running process instances.

When an enterprise application that contains process modules is installed and deployed, the process templates that are contained in the process modules are put into the start state.

1. In the navigation pane of the administrative console, click **Applications > Enterprise Applications**.
2. Click the application that you want to manage.
3. Under Related Items, click **EJB Modules** and select an EJB module.
4. Under Additional Properties, click **Business Processes**.
5. Select the process template that you want to manage.
   a. To start the process template, click **Start**.
   b. To stop the process template, click **Stop**.

Uninstalling BPEL-based process applications

To uninstall an enterprise application that contains processes based on the Business Process Execution Language (BPEL):

1. Verify that all the stand-alone servers, the database, and the application server (or at least one application server per cluster) are running.
   - In a Network Deployment (ND) environment, the ND application server, all ND-managed stand-alone application servers, and at least one application server for each cluster where the application is installed must be running.
   - If you use the wsadmin tool to install applications, make sure that you are connected to a process. In an ND environment, the process is the deployment manager; in a stand-alone environment, it is the application server.
2. Stop all process templates in the application. This action prevents the creation of process instances.
   a. Click **Applications > Enterprise Applications** in the administrative console navigation pane.
   b. Select the application that you want to stop.
   c. Under Related Items, click **EJB Modules** and select an Enterprise JavaBeans (EJB) module.
   d. Under Additional Properties, click **Business Processes**.
   e. Select all process templates by clicking the appropriate check box.
   f. Click **Stop**.

   Repeat this step for all EJB modules that contain business processes.
3. Verify that no process instances are running. If necessary, a process administrator can use the process choreographer Web client to stop running process instances.
4. To stop and uninstall the application:
   a. Click **Applications > Enterprise Applications** in the administrative console navigation pane.
   b. Select the application that you want to uninstall and click **Stop**. This step fails if any process instances belonging to the application are still running
   c. Select the application that you want to uninstall **again**.
   d. Click **Uninstall**.

The process application is uninstalled. If the application contains an interruptible process, the corresponding tables are deleted from the databases on the deployment targets. If the tables cannot be deleted, for example, because you do not have the database access rights to delete tables, an exception is written to the `SystemOut.log` file.

Installing V5.0-style process applications

To install an enterprise application that contains V5.0-style process modules:

1. Verify that the business process container of the application server is installed and configured for each application server that uses process choreographer.
2. Verify that the application server is running. In this respect, process applications are different from other Java 2 Platform Enterprise Edition (J2EE) applications.
   - In a Network Deployment (ND) environment, the ND application server, all ND-managed stand-alone application servers, and at least one application server must be running for each cluster where you install the application.
   - If you use the wsadmin tool to install applications, make sure that you are connected to a process. In an ND environment, the process is the deployment manager, in a stand-alone environment it is the application server. When you run the wsadmin tool, do not use the `-conntype NONE` option as one of the installation options.
3. Install your application on your application server.

All process templates are put into the start state. Instances of the process templates are created as soon as requests arrive if they are issued using the process choreographer generic API. For other types of requests, you must start the application before process template instances can be created.

Stopping and starting V5.0-style process templates

Verify that you are using a console user ID that has either the `operator` or the `administrator` role.

When an enterprise application that contains process modules is installed and deployed, the process templates that are contained in the process modules are started automatically. You can stop these process templates if required, and restart them.

1. In the navigation pane of the administrative console, click **Applications > Enterprise Applications**.
2. Click the application that you want to manage.
3. Under Related Items, click **Business Process Modules**.
4. Select the process module that you want to manage.
5. In the Additional Properties section, click **Templates**. The Process Templates page is displayed. It lists the process templates that are contained in the selected process module.

   You can stop a running process template and start it again.
6. To stop a process template, select the check box next to the process template and click **Stop**.
7. To start a process template, select the check box next to the process template and click **Start**. As long as the started process exists, do not delete the user ID that you used to start the process from the user registry. The navigation of a process relies on the Java 2 Platform Enterprise Edition (J2EE) security context of the process starter, and the process cannot continue if the user ID does not exist in the user registry. If you delete such a user ID by accident, you must recreate it to continue navigating this process.

Uninstalling V5.0-style process applications

To uninstall an enterprise application that contains process modules:

1. Verify that all the stand-alone servers, the database, and the application server (or at least one application server per cluster) are running.
   - In a Network Deployment (ND) environment, the ND application server, all ND-managed stand-alone application servers, and at least one application server for each cluster where the application is to be installed must be running.
   - If you use the wsadmin tool to install applications, make sure that you are connected to a process. In an ND environment, the process is the deployment manager, in a stand-alone environment, it is the application server.
2. Stop all process templates in the application. This action prevents the creation of process instances.
   a. Click **Applications > Enterprise Applications** in the administrative console navigation pane.
   b. Select the application that you want to stop.
   c. Under Related Items, click **Business Process Modules**.
   d. Select a business process module.
   e. Under Additional Properties, click **Templates**.
   f. Select all process templates by clicking the appropriate check box.
   g. Click **Stop**.

   Repeat this step for all business process modules.
3. Verify that no process instances are running. If necessary, a process administrator can use the process choreographer Web client to stop running process instances.
4. To stop and uninstall the application:
   a. Click **Applications > Enterprise Applications** from the navigation pane.
   b. Select the application that you want to stop.
   c. Click **Stop**.
   d. Click **Uninstall**.

The process application is uninstalled.

Business Process collection

Use this page to manage business processes associated with a process application.

To view this administrative console page, click **Applications > Enterprise Applications >** *application_name* **> EJB Modules >***ejbmodule_name* **> Business Processes**.

Name

Name of the business process.

Data type String

Valid From Time

Specifies the time from which processes of this type can be started.

You can stop a business process and start it again later, depending on the Valid From time that is specified.

Data type String
Units Coordinated Universal Time (UTC)

Current State

Specifies the current state of the business process.

Range

Started The selected business process is started, which means that new instances of it this process are created when request messages arrive.

Stopped The selected business process is stopped. No new instances are created.

Business Process settings

Use this page to modify business process settings.

To view this administrative console page, click **Applications > Enterprise Applications >** *application_name* **> EJB Modules >***ejbmodule_name* **> Business Processes** > *businessprocess_name*.

Name

Name of the business process.

Data type String

Valid From Time

Specifies the time from which processes of this type can be started.

You can stop a business process and start it again later, depending on the Valid From time that is specified.

Data type String
Units Coordinated Universal Time (UTC)

Current State

Specifies the current state of the business process.

Range

Started The selected business process is started, which means that new instances of it this process are created when request messages arrive.

Stopped The selected business process is stopped. No new instances are created.

Process modules collection

Use this page to view a list of the process modules defined for this application.

A process module contains process templates. When a process module is started, all its templates are enabled and instances of the process templates can be created.

To view this administrative console page, click **Servers > Applications >** *application_name* **> Business Process Modules**.

Process module settings

Use this page to manage process modules.

A business process module contains one or more business process templates.

To view this administrative console page, click **Applications > Enterprise Applications >** *application_name* **> Process Modules >***processmodule_name*.

Process templates collection

Use this page to manage business process templates associated with a process module.

To view this administrative console page, click **Applications > Enterprise Applications >** *application_name* **> Process Modules >***processmodule_name* **> Templates**.

Name:

Name of the process template.

Data type String

Valid From:

Specifies the time from which the process can start.

You can stop a running process template and start it again later, depending on the Valid From time that is specified.

Data type String
Units Coordinated Universal Time (UTC)

Current State:

Specifies the current state of the process template.

Range

Started The selected process template is started, which means that new instances of it this template are created when request messages arrive.

Stopped The selected process template is finished and then stopped. No new instances are created.

Process template settings:

Use this page to modify business process template settings.

A process template is the definition of one particular business process.

To view this administrative console page, click **Applications > Enterprise Applications >** *application_name* **> Process Modules >***processmodule_name* **>** *processtemplate_name*.

Name:

Name of the process template.

Data type String

Valid From:

Specifies the point in time from which the process can be started.

You can stop a running process template and start it again later, depending on the Valid From time that is specified.

Data type String
Units Coordinated Universal Time (UTC)

Current State:

Specifies the current state of the process template.

Range

Started The selected process template is started, which means that new instances of this template are created when request messages arrive.

Stopped The selected process template is finished and then stopped. No new instances are created.

Chapter 8. Using the process choreographer Web client

This set of topics describes how you can use the process choreographer Web client to work with deployed business processes.

Before you can use the process choreographer Web client to work with processes and their associated activities, you must start the Web client. If you are using the Web client in a remote installation, for example, in a cluster, you must also configure the Web client.

You can use the process choreographer Web client to display information about process templates, process instances, services, and work items. You can also act on processes, services, and work items; for example, to start new process instances, start services, or to claim and complete a work item in your To Do list.

The process templates, process instances, and work items that you can see depend on the authority that is assigned to your user ID, or the role under which you are working.

You can use the Web client to work with business processes that you deployed. If you encounter problems when using the Web client, see troubleshooting the Web client, for more information on how to solve these problems.

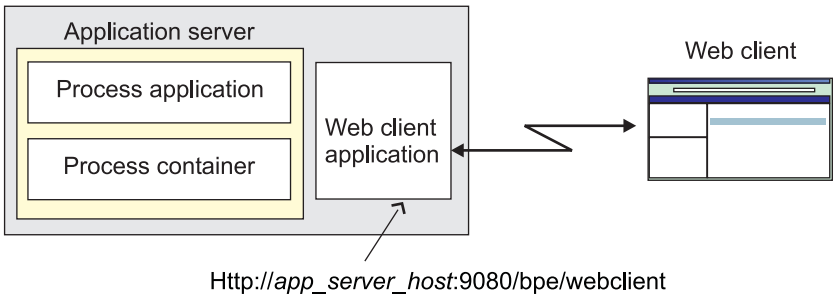
- Work with work items
- 5.1+ Manage work items
- Work with process instances
- Work with process templates
- Customize the Web client

About the process choreographer Web client

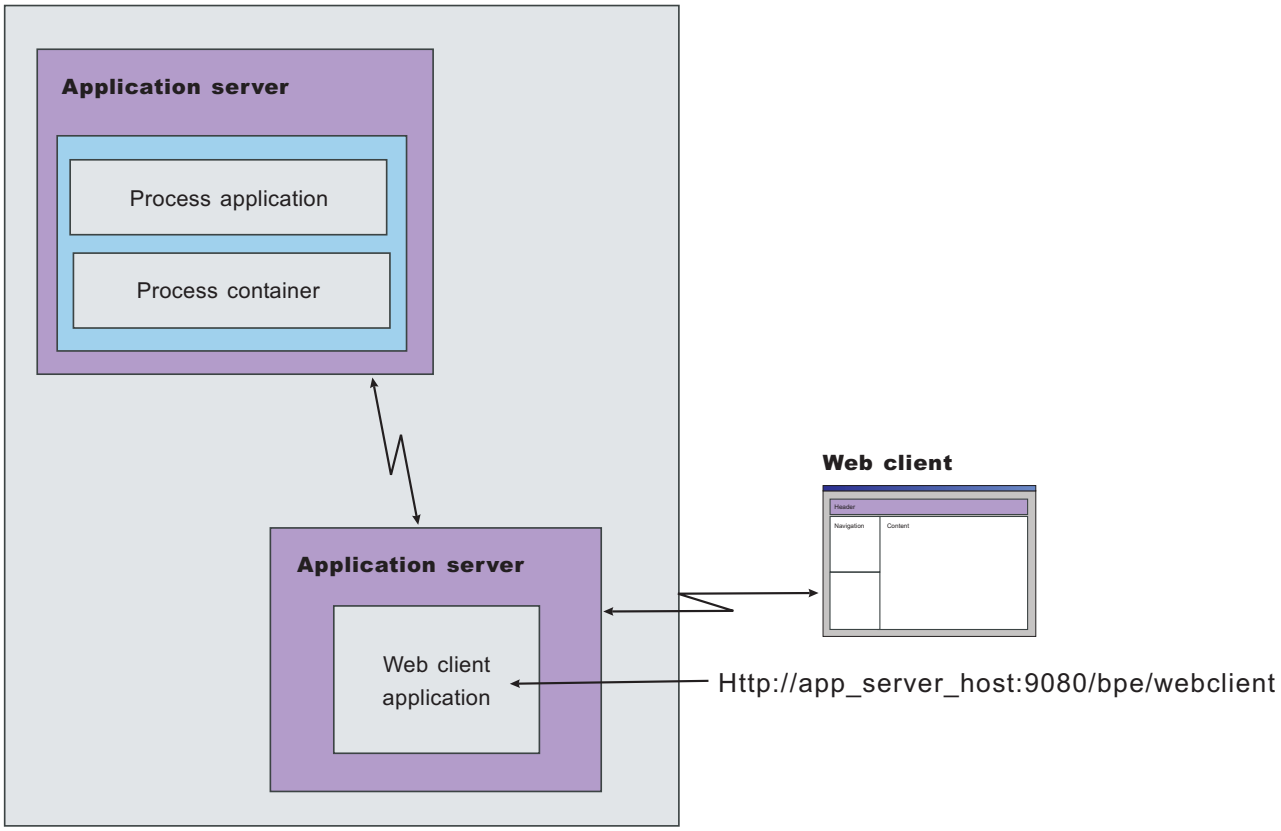
You can use the process choreographer Web client to work with business processes that you deployed as applications.

The Web client can run as:

- A Web application on the host where your business process application is deployed.



- A Web application on a host other than the one where your business process application is deployed.



If you want to work with business process applications on several application servers at the same time, you need to start a process choreographer Web client for each application server.

You can use the Web client to display information about work items, services, process instances, and process templates. You can also act on process instances and work items; for example, you can start new process instances or claim and complete an activity from your To Do list. When you complete an activity, its output message is saved. The data is available to subsequent activities in the process.

Configuring the process choreographer Web client manually

You have configured process choreographer but you have not yet configured the Web client or you want to add Web clients to an existing process choreographer installation.

With most types of installations, the process choreographer Web client is already installed on your system. However, in the following situations you have to install and configure the Web client manually.

- Connecting to a clustered process choreographer configuration
- Connecting to a process choreographer configuration that is running on a remote WebSphere Application Server installation (that is, process choreographer is not running on the same server or cluster as the Web client)

- Configuring the Web client on a WebSphere Application Server base node in a Network Deployment (ND) environment

Go to the process choreographer directory and invoke the `bpeportal.jacl` script. On Windows systems, run:

```
cd install_root\ProcessChoreographer
..\bin\wsadmin.bat -f bpeportal.jacl
```

On UNIX systems and z/OS, run:

```
cd install_root/ProcessChoreographer
../bin/wsadmin.sh -f bpeportal.jacl
```

1. If WebSphere security is enabled, add the `-user` and `-password` command-line options with the appropriate parameters for the user ID and password.
2. If you are configuring the Web client on an unmanaged node and the application server is not running, add the `-conntype NONE` command-line option.
3. If you are configuring the Web client on a WebSphere Application Server base node in an ND environment, make sure that the deployment manager is running.
4. If you are configuring the Web client on a WebSphere Application Server enterprise node in an ND environment, the deployment manager does need not to run. Run the command on the deployment manager node and add the `-conntype NONE` command-line option.

The `bpeportal.jacl` script prompts you for the required information.

The Web client is configured and ready to use. If you have problems with the configuration, check the log file written by the `bpeportal.jacl` script. This log is located in the `logs/bpeportal.log` file. This directory also contains a `wsadmin.traceout` file that might contain more information about the problem.

Example: Configuring the process choreographer Web client on Windows platforms

This example configures the process choreographer Web client on Windows platforms using the `bpeportal.jacl` script. The script prompts you for the required information. To select the first in a list of options, press the Enter key at the prompt.

- Invoke the `bpeportal.jacl` script from the `%WAS_HOME%\ProcessChoreographer` directory

```
d:\WebSphere\DeploymentManager\ProcessChoreographer>
..\bin\wsadmin -f bpeportal.jacl
WASX7209I: Connected to process "dmgr" on node viennaManager
using SOAP connector;
The type of process is: DeploymentManager
************************************************************
Process Choreographer Web client configuration started.
```

- If clusters are configured, specify whether to install the Web client on a cluster or on a standalone server. In this example, the Web client is installed on a cluster.

```
Install the Web client on a standalone server or in a cluster
[Standalone/cluster]? c
==> cluster
```

- Specify the name of the cluster.

```
Name of the cluster where to install the Web client [cluster1]:
==> cluster1
```

- Specify the name of the virtual host for the Web client.

  ```
  Virtual Host for Web client [default_host]:
  ==> default_host
  ```

- Specify how the Web client communicates with process choreographer.

  ```
  Connect the Web client to a clustered or a standalone Process Choreographer
  [Clustered/standalone]?
  ==> Clustered
  ```

- Specify the location of the process choreographer configuration.

  ```
  Cluster of Process Choreographer to connect to [cluster1]:
  ==> cluster1
  ```

  The installation starts.

  Example output from the scripting command issued by the `bpeportal.jacl` script:

```
WASX7327I: Contents of was.policy file:
 grant codeBase "file:${application}"
{
 permission java.lang.RuntimePermission "getClassLoader";
 permission java.lang.RuntimePermission "createClassLoader";
  permission java.lang.RuntimePermission "setContextClassLoader";
  permission java.lang.RuntimePermission "accessDeclaredMembers";
  permission java.lang.RuntimePermission "accessClassInPackage.*";
  permission java.lang.RuntimePermission "charsetProvider";
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
};

ADMA5016I: Installation of BPEWebClient_cluster1 started.
ADMA5005I: Application BPEWebClient_cluster1 configured in WebSphere repository
ADMA5001I: Application binaries saved in
d:\WebSphere\DeploymentManager\wstemp\Scriptfa85d589c3\workspace\cells\viennaNet
work\applications\BPEWebClient_cluster1 installed successfully.
Done installing BPEWebClient_cluster1.
```

Starting the process choreographer Web client

With most types of installations, the process choreographer Web client is already installed on your system. Before you can start the process choreographer Web client, you must complete the following prerequisite steps:

1. Configure the business process container. This step enables the process choreographer service and the process choreographer Web client. Depending on the type of installation you perform, your business process container might already be configured.

   To check whether you have a business process container configured on your server, click **Applications > Enterprise Applications** in the navigation bar of the administrative console. Your process container is available if the list of enterprise applications contains an application named `BPEContainer_`*`identifier`*. The Web client is already installed if an application named `BPEWebClient_`*`identifier`* appears in the list of applications. The identifier for both the business process container and the Web client depends on where you installed the business process container.

Business process container installed on:	**`BPEContainer_`*`identifier`***
Application server	`BPEContainer_`*`nodeName_serverName`*
Cluster	`BPEContainer_`*`clusterName`*

   If a business process container is not available, you must configure one on your server before you can proceed.

2. Configure the WebSphere Application Server security environment for secured applications, including assigning users and groups to roles defined in business process applications and configuring authentication mechanisms. For more information about configuring the WebSphere Application Server security environment for secured applications, see the *WebSphere Business Integration Server Foundation Applications* PDF.
3. Deploy the applications that use the business processes. These can be V5.0-style business processes or processes based on the Business Process Execution Language (BPEL).

If you want to work with business process applications on several application servers at the same time, you need to start a process choreographer Web client for each application server.

To start the process choreographer Web client, complete the following steps:

1. Use a Web browser to open the following Web page:

   `http://app_server_host:port_no/bpe/webclient`

   app_server_host
   The network name for the host of the application server that provides the business process application with which you want to work.

   port_no
   The port number used by the Web client. The port number depends on your configuration.

   The Web client prompts you for a user ID and password.
2. Type a user ID and password then click **OK**.

This task displays the initial page of the process choreographer Web client, which shows the work items in your To Do list.

Working with work items

This set of topics describes how you can use the process choreographer Web client with work items belonging to interruptible business processes.

5.1 + No human interaction is involved in a non-interruptible process. Depending on the type of process, a process output message is displayed immediately after the process finishes. Not all processes based on the Business Process Execution Language (BPEL) have output messages, for example, if the process implements a one-way operation, an output message is not displayed.

For an interruptible process, you can use the process choreographer Web client to work with activities that require human interaction. If you are authorized as a potential owner, editor, or reader of an activity, the work item is added automatically to your To Do list which is displayed in the My To Dos page of the Web client.

The My To Dos page is not refreshed automatically, for example, when you click **Back**. To see the latest version of your To Do list, you must refresh the My To Dos page manually.

You can use the Web client to work with work items. You can:
- Display work items in your To Do list
- Claim a work item

- Complete a work item
- Query work items
- Display information about an activity

Displaying work items in your To Do list

If you are authorized to work with a new work item, it is added automatically to your To Do list. You can view your To Do list by displaying the My To Dos page. The My To Dos page is displayed when you first start the Web client, and you can return to that page at any time from other Web client pages. The My To Dos page is not refreshed automatically. If you change a work item, or want to check the latest contents of your To Do list, you must refresh the My To Dos page manually.

From the My To Dos page, you can select a work item on which to work; for example, to claim a work item that you want to complete.

To display the My To Dos page, click **My To Dos** under Work Item Lists in the navigation pane of the Web client.

This task displays the My To Dos page of the process choreographer Web client.

Claiming a work item

To work with a work item, you must claim the work item and then perform the actions needed to complete it. You can claim a work item that is in the ready state if you are a potential owner or the administrator of that work item. If you claim a work item, you become the owner of that work item and are responsible for completing it.

You can claim a ready work item from its Activity page; for example, by completing the following steps:

1. Click **My To Dos** under Work Item Lists in the navigation pane of the Web client. This action displays the My To Dos page, which lists the work items with which you can work.
2. Click the work item name.

   This action displays the Activity page, with **Claim** displayed as an available action. This page also displays information about the process to which the work item belongs and the output message for the work item. You cannot change the fields displayed until you claim the work item.
3. Optional: For more detailed information about the activity associated with the work item, click **View more details about this activity**. However, you need to return to the Activity page to claim the work item.
4. Claim the activity. Click **Claim**.

When you claim a work item, the My To Dos page is displayed.

To work on the work item, click its **To Do Name**. The Activity page is displayed with the output message fields enabled for you to work with.

Completing a work item

Before you can complete a work item, you must claim the work item, as described in Claiming a work item.

After you have claimed a work item, you can perform the actions needed to complete it.

You do not have to complete a work item in one step. While working with a work item, you can save changes you make to the work item and leave the work item to do other things. For example, you might want to save your changes if your work is interrupted or if you need to get more information before you can complete the work item.

1. Display the Activity page, for example, by clicking the name of the activity on your My To Dos page.
2. Complete the input for the work item. The content of the Activity Output Message section, including the number and types of input fields, depends on the design of the activity in the process template.
3. Optional: If you want to save your changes, click **Save Changes**.
4. If you have finished all the required input, you can complete the work item. Click **Complete**. When you click **Complete**, you cannot make any more changes to the work item. If required input is missing, you cannot complete the work item. The Activity page indicates the input you still need to supply to complete the work item.

When you complete the work item, the Web client window refreshes to display your My To Dos page. Information about the completed work item is saved. This information is now available to subsequent activities in the process.

Querying work items

You can define queries to find work items that have certain characteristics. For example, you can define a query to find all the work items that have been claimed by a particular user or all the work items that you have claimed.

1. Click **Define Work Item List** under Work Item Lists in the navigation pane. This action displays the Define Work Item List page.
2. Specify criteria for querying work items and click **Show List**. This action displays the User-Defined Work Item List page. This page displays only those work items that meet all of your search criteria.
3. Optional: Work with the work items found by the query.
4. Optional: If you are authorized as the business process administrator, you can save the query. The saved query appears under Work Item Lists in the navigation pane.

Displaying information about an activity

You can use the process choreographer Web client in the following ways to view information about an activity associated with a work item:

- On the My To Dos page, click the name of the work item.

  This action displays the Activity page, which provides the set of information and actions needed to work with the activity in its current state.
- On the Process Instance page, click the name of the activity listed under Activities.

  This action displays the Activity page, which provides the set of information and actions needed to work with the activity in its current state.
- On the Activity page, click **View more details about this activity**.

This action displays the Activity Information page, which provides more detailed information about the activity with which the work item is associated. The Activity Information page also provides information about the process to which the activity belongs.

Managing work items for BPEL-based processes

This set of topics describes how you can use the process choreographer Web client to manage work items for processes based on the Business Process Execution Language (BPEL).

Work items are the means by which ″pieces of work″ are assigned to people. A work item represents the relationship between a person or group of people and an activity instance or process instance. These relationships are set up in the process model.

However, sometimes, it can be necessary to change a relationship after a process instance has started, for example, to transfer a work item from the original owner to someone else. You might also need to create additional work items or delete work items that are not needed anymore.

For BPEL-based processes, you can manage work items for activity instances and process instances. You can:

- Create work items
- Delete work items
- Transfer work items

Creating work items for BPEL-based processes

To create a work item for either an activity instance or a process instance, you must be the administrator of the process instance or the business process administrator. Also, the activity instance or the process instance must be in one of the following states:

Activity instance
States: claimed, waiting, ready, stopped

Process instance
States: running, failing, terminating

For BPEL-based processes, you can create work items for activities and process instances. You might want to create work items for new potential owners, for example, when none of the current potential owners can accept any additional work. You might also want to create work items if the query against the staff repository does not return any potential owners. This might happen, for example, in long-running, interruptible processes if there have been organizational changes since the process started.

1. For work items for process instances, select a process instance.
   a. Click **Manage Work Items for Process Instances** under Administration in the navigation pane.
   b. Select the process instance for which you want to create work items and click **Create**.
2. For work items for activity instances, select an activity instance.
   a. Click **Manage Activities for Process Instances** under Administration in the navigation pane.

b. Select the process instances to which the activity instances belong and click **Show Activities**.
c. Select the activity instance for which you want to create work items and click **Create**.

3. Create the work items.
   a. In the **Create for** field, specify the user ID of the new work-item owner.
   b. Select one or more roles from the **Reason** list. These roles determine the actions the assigned person can perform on the new work item.
   c. Click **Create**.

A work item is created for each role specified for the new work-item owner.

Deleting work items for BPEL-based processes

To delete a work item for either an activity instance or a process instance, you must be the administrator of the process instance. Also, the activity instance or the process instance must be in one of the following states:

Activity instance
States: claimed, waiting, ready, stopped

Process instance
States: running, failing, terminating

For processes based on the Business Process Execution Language (BPEL), you can delete work items for activities and process instances. You might want to delete work items, for example, if you created work items in error or if work items are generated for someone who no longer works with the company.

1. For work items for process instances, select the process instances.
   a. Click **Manage Work Items for Process Instances** under Administration in the navigation pane.
   b. Select the process instances for which you want to delete work items and click **Delete**.
2. For work items for activity instances, select the activity instances.
   a. Click **Manage Activities for Process Instances** under Administration in the navigation pane.
   b. Select the process instances to which the activity instances belong and click **Show Activities**.
   c. Select the activity instances for which you want to delete work items and click **Delete**.
3. Delete the work items.
   a. Select the work items.
   b. Click **Delete**.

The work items are deleted.

Transferring work items for BPEL-based processes

To transfer a work item for an activity instance, you must either be the owner of the work item or the administrator of the process instance. To transfer a work item for a process instance, you must be the administrator of the process instance. Also, the activity instance or the process instance must be in one of the following states:

Activity instance
States: claimed, waiting, ready, stopped

Process instance
States: running, failing, terminating

For processes based on the Business Process Execution Language, you can transfer work items for activities and process instances for one owner to another owner. You might want to do this, for example, if you have started working on a work item but you realize that some of the information needed to complete it can be provided only by one of your colleagues. You might also want to transfer a work item if the work-item owner is on vacation and the work item must be completed before he returns.

1. For work items for process instances, select the process instances.
   a. Click **Manage Work Items for Process Instances** under Administration in the navigation pane.
   b. Select the process instances for which you want to transfer work items and click **Transfer**.
2. For work items for activity instances, select the activity instances.
   a. Click **Manage Activities for Process Instances** under Administration in the navigation pane.
   b. Select the process instances to which the activity instances belong and click **Show Activities**.
   c. Select the activity instances for which you want to transfer work items and click **Transfer**.
3. Transfer the work items.
   a. In the **Transfer to** field, specify the user ID of the new work-item owner.
      - If the current work-item owner is the process administrator, you can specify any user ID.
      - If the current work-item owner is not the process administrator, you can specify only a user ID that is defined in the process template.
   b. Select the work items.
   c. Click **Transfer**.

The transferred work items appear on the My To Dos page of the new work-item owner.

Working with process instances

This set of topics describes how you can use the process choreographer Web client to work with business processes instances.

You can use the Web client to display information about process instances or act on process instances. The Web client displays only those process instances with which you are authorized to work, and displays only those actions that you are authorized to perform on the process instances.

You use the Web client to:
- Display information about a process instance
- Work with process instances you administer
- Work with process instances you started
- Monitor a process instance
- Administer compensation for a process instance

- Terminate a process instance
- Query process instances

Displaying information about a process instance

You can use the process choreographer Web client in the following ways to view information about any process instance:

- To display information about a process instance that you started:
  1. Click **Created By Me** under Process Instance Lists in the navigation pane.
  2. In the Created By Me page, click the name of the process instance.
- To display information about a process instance that you administer:
  1. Click **Administered By Me** under Process Instance Lists in the navigation pane.
  2. In the Administered By Me page, click the name of the process instance.
- On an Activity page, click **View more details about this process**.

5.1 + The Web client displays a Process Instance page with information and available actions for the process instance. The information displayed includes a list of the work items in the process. If the process based on the Business Process Execution Language (BPEL), this page also includes a list of the events in the process. You can click an activity name or event name to display information about that work item or service. If the process instance produced an output message, this page includes this message.

Working with process instances you administer

To administer processes, you need to have administrator authority for the business process.

The Administered By Me page of the process choreographer Web client displays a list of only those process instances for which you are the process administrator.

1. Display the Administered By Me page. Click **Administered By Me** under Process Instance Lists in the navigation pane.
2. Click the name of the process instance with which you want to work.

   5.1 + The Web client displays a Process Instance page with information and available actions for the process instance. The information displayed includes a list of the work items in the process. If the process based on the Business Process Execution Language (BPEL), this page also includes a list of the events in the process. You can click an activity name or event name to display information about that work item or service. If the process instance produced an output message, this page includes this message.
3. Optional: To act on the process instance, click an action displayed under Available Actions. The actions that are available depend on the current state of the process instance. For example, if a process instance is in the running state, **Terminate** is displayed.

Working with process instances you started

The Created By Me page of the process choreographer Web client displays a list of the process instances that you started.

1. Display the Created By Me page. Click **Created By Me** under Process Instance Lists in the navigation pane.
2. Click the name of the process instance with which you want to work.

   5.1+ The Web client displays a Process Instance page with information and available actions for the process instance. The information displayed includes a list of the work items in the process. If the process based on the Business Process Execution Language (BPEL), this page also includes a list of the events in the process. You can click an activity name or event name to display information about that work item or service. If the process instance produced an output message, this page includes this message.
3. Optional: To act on the process instance, click an action displayed under Available Actions. The actions that are available depend on your authorization and the current state of the process instance. For example, if the process instance is in the running state, **Terminate** is displayed.

Monitoring a process instance

You can view the progress of the activities in a process instance.

1. On any of the process instance list pages, for example, Administered By Me, select the process instance you want to monitor, then click **Monitor**. The Process Instance Monitor page is displayed. This page gives a brief description of the process, shows its activities and the status of each of the activities.
2. Optional: Click an activity to see more information about it.

Administering compensation for a process instance

If compensation fails for a process instance, there are a number of administrative actions you can take.

1. 5.1+ Do one of the following:
   - Under Process Instance Lists, click **Undo Actions in Error**.
   - Click **Repair Compensation** on any of the pages that display process instances.

   The Undo Actions in Error page is displayed. Each failed compensation action is named. This page also contains information as to why the compensation action failed. It can help you to decide what actions to take to correct the failed compensation action.
2. Select a process instance and then click one of the available actions. The following administrative actions are available:

   Skip Compensating Action
   Skip the current compensating action and continue with compensating the process instance.

   Retry Compensating Action
   If you have taken action to correct the failed compensation action, click **Retry Compensating Action** to try the compensation action again.

   Stop Compensation
   Stop the compensation process.

   If you selected **Skip Compensating Action**, this action might result in a non-compensated activity.

Terminating a process instance

To terminate a process instance, you must have appropriate access rights as a process administrator.

You might want to terminate a process instance if the work or documents it represents are no longer needed, if no one is available to complete the process instance, if you have encountered problems with the process template and it needs to be redesigned, and so on.

To terminate a process instance, do one of the following:
- Click **Terminate** on the Process Instance page.
- On any of the process instance list pages, for example, Administered By Me, select the check box next to the process instance, then click **Terminate**.

This action stops the process instance immediately without waiting for any outstanding activities. Process instances that are terminated are not compensated.

Querying process instances

You can define queries to find process instances that have certain characteristics. For example, you can define a query to find all the process instances that are currently in a failed state.

1. Click **Define Process Instance List** under Process Instance Lists in the navigation pane. This action displays the Define Process Instance List page.
2. Specify criteria for querying process instances and click **Show List**. This action displays the User-Defined Process Instance List page.

   This page displays only those process instances that meet all of your search criteria.
3. Optional: Work with the process instances found by the query.
4. Optional: If you are authorized as the business process administrator, you can save the query. The saved query appears under Process Instance Lists in the navigation pane.

Working with process templates

This set of topics describes how you can use the process choreographer Web client to work with templates for business processes.

You can use the Web client to display information about process templates. If you are authorized to do so, you can start a process instance.

You use the Web client to:
- View a business process template
- Start a business process instance
- Query process templates

Displaying information about a business process template

To display information about a process template in the process choreographer Web client, complete the following steps:

1. Select a process template from the list of process templates. Click **My Templates** under Process Template Lists in the navigation pane and select a template from the list.
2. Click **View**.

5.1 +

This task displays a Process Template page with information about the process template.

If this is a process based on the Business Process Execution Language (BPEL), a list of the services for which you are authorized is also shown. To start a process instance, select one of the services.

If this is a V5.0-style process, and you are authorized to start a process from the template, a **Start Instance** action is displayed.

Starting a new process instance

You can start a new process instance from any of the process templates that you are authorized to use.

To start a new process instance, complete the following steps:

1. Click **My Templates** under Process Templates Lists in the navigation pane. This action displays the My Templates page.
2. 5.1 + Select a process template and click **Start Instance**. This action displays the Process Input Message page, which provides a description of the process template. You can complete starting the process instance from this page. For a process based on the Business Process Execution Language (BPEL) that contains more than one operation that you are authorized to start, this action displays the Process Template page. On this page, select a service and click **Start Instance**. This action displays the Process Input Message page.
3. Optional: If the process is an interruptible process, you can type in a process instance name.
4. Complete the input for the process input message.
5. To start the process, click **Start Instance**.

If the business process contains an activity that requires human interaction, a work item is added to the To Do lists of all the potential owners. If you are one of these potential owners, you can display your My To Dos page to work with the work item.

5.1 + If the process instance is a non-interruptible process, it displays an output message when the process ends. If the process is a BPEL-based process and it implements a one-way operation, an output message is not displayed.

Querying process templates

You can define queries to find process templates that meet a certain set of characteristics. For example, you can define a query to find all the process templates that are valid from a certain date.

1. Click **Define Template List** under Process Template Lists in the navigation pane. This action displays the Define Template List page.
2. Specify criteria for querying process templates and click **Show List**. This action displays the User-Defined Process Template List page.

   This page displays only those process templates that meet all of your search criteria.
3. Optional: Work with the process templates found by the query.

Customizing the process choreographer Web client

Process choreographer provides a ready-to-use Web user interface based on JavaServer Pages (JSP) files and servlets. You can use the Web interface as it is or adapt it to fit your needs. Typically, user interfaces for business process applications often require customization, for example, to adapt the user interface to fit a certain look and feel.

The following tasks describe the different ways in which you can customize the process choreographer Web client:
- Adapt the look and feel
- Create user-defined JSP files

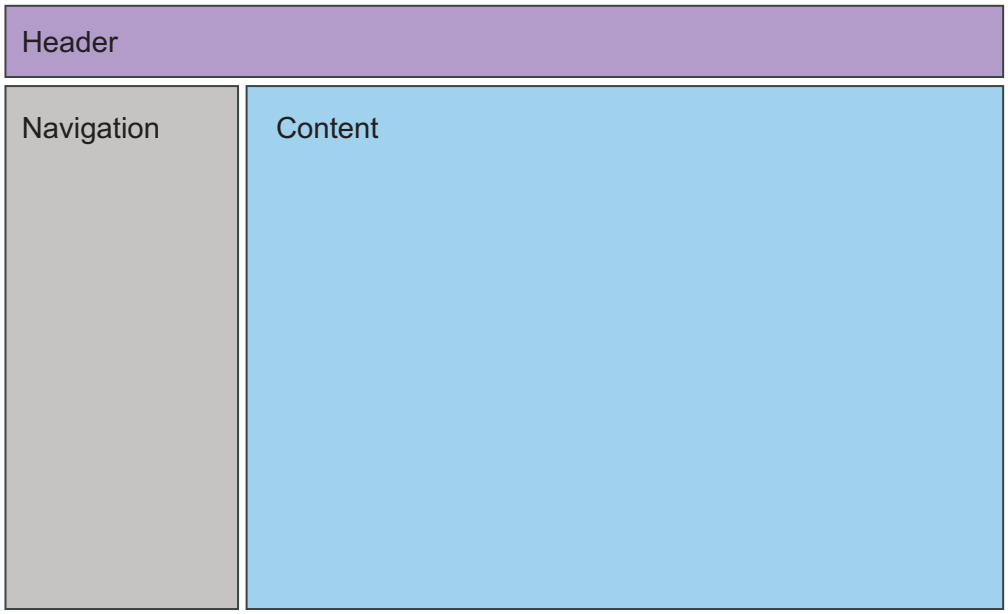
Adapting the look and feel

Process choreographer provides a ready-to-use Web user interface based on JavaServer Pages (JSP) files and servlets. You can adapt the user interface to fit a certain look and feel without writing any new JSP files. The Web client interface consists of a header, a navigation pane, and a content pane. A style sheet controls how the Web interface is rendered.

1. Optional: Modify the header. The `Header.jsp` file is always displayed in the Web client. The default `Header.jsp` file contains logos, images, and a link to the information center.
2. Optional: Modify the style sheet. The default style sheet, `style.css`, contains styles for headings, paragraphs, and tables. It also contains classes, mainly for table and table-cell elements.

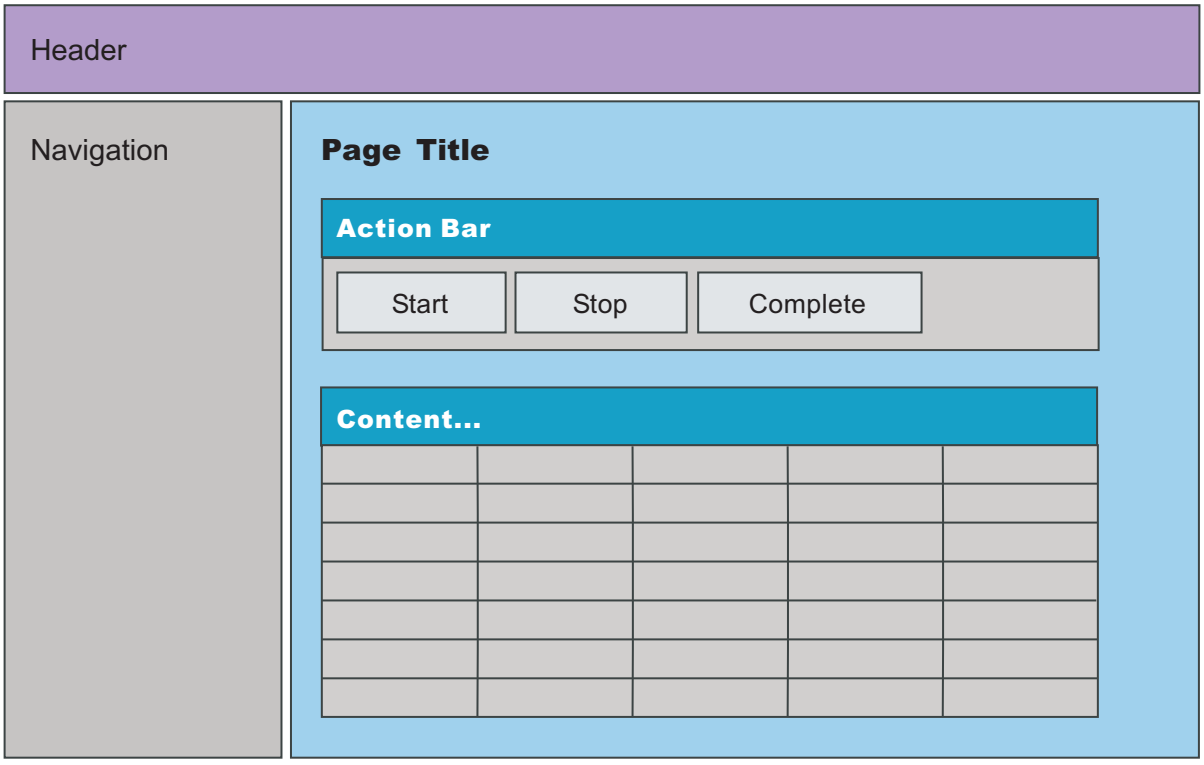
Layout of the process choreographer Web client user interface

The Web client interface consists of a header, a navigation pane, and a content pane.



The header and the navigation panes are always displayed. They are generated by the `Header.jsp` and `Navigation.jsp` files. Other JavaServer Pages (JSP) files use the `<jsp:include page=xxx>` tag to reference them. The information shown in the content pane depends on the JSP file that is used to generate the page.

The layout of the Web client is implemented with HTML tables. Each page consists of the main table and the header:



The main table has one row and one column and includes the `Navigation.jsp` pane. It also provides a table cell for the page content. Depending on the content,

this cell can also contain tables, forms, and labels. The HTML template for the page layout looks similar to the following example:

```
<body>
 <jsp:include page="Header.jsp" flush="true"/>
 <table class="page">
  <tr>
   <jsp:include page="Navigation.jsp" flush="true"/>
   <td class="content">
     ...
    </td>
  </tr>
 </table>
</body>
```

Creating user-defined JSP files

When you model a process in WebSphere Studio Application Developer Integration Edition, you can define business-process specific JavaServer Pages files (or user-defined JSP files) for the process and for staff activities. The Web client uses these JSP files to display input and output messages for the process and for the activities for which user-defined JSP files have been defined. For example, if a message has non-primitive parts, it is recommended that you use user-defined JSP files to enhance the usability of the message for the user.

You can also use user-defined JSP files to extend the Web client, for example, these files can check user input for correctness before it is sent to the business process container. If you have not defined any user-defined JSP files, the Web client generates a default rendering to display message data.

A message-mapping JSP file is required for each user-defined JSP file that receives user input data (either a JSP file that displays a process input message or an activity output message). The message-mapping JSP file receives the user data, validates the input data, wraps it in an appropriate message object, and then forwards the message to the business process container.

User-defined JSP files are not self-contained Web pages. To include user-defined JSP files in the Web client, you must specify them when you model a business process in WebSphere Studio.

1. Optional: Create JSP files for displaying messages
2. Optional: Create JSP files for processing user input
3. Integrate user-defined JSP files in a V5.0-style process model

Creating JSP files for displaying messages

The process choreographer Web client displays read-only messages for various purposes. For example, the process input message (after the process has started) and the process output message cannot be edited by users. Some of these messages require more information to enhance their usability. You can create user-defined JavaServer Pages (JSP) files to display this additional information.

A user-defined JSP that displays messages and more information about these messages receives the message data with the help of the BusinessProcess session bean.

1. Make the BusinessProcess bean available to the JSP file.

```
BusinessProcessService process =
        MessageUtilities.getBusinessProcessService(request);
```

You can use the BusinessProcessService interface to develop applications that work with both the remote and the local object of the bean.

2. Receive the input or output messages of the process or activity using either the getInputMessage() or the getOutputMessage() method. When user-defined JSP files are called, they receive the ID of the object they are displaying. Call the getParameter() method to receive the ID from the HttpServletRequest object. If the JSP file is related to an activity, the activity instance ID (AIID) comes as a string with the HttpServletRequest. Similarly, JSP files that work with processes receive the process instance ID (PIID) of the process as a string. For example, to receive the process output message of a process as an `org.apache.wsif.base.WSIFDefaultMessage` message:

   ```
   String piid = request.getParameter(Constants.WF_PIID);
   WSIFDefaultMessage outMsg =
       (WSIFDefaultMessage)process.getOutputMessage(piid).getObject();
   ```

   The ID of the process is received using the getParameter() method. Use the getOutputMessage(String piid) method in the com.ibm.bpe.api.BusinessProcess class to return a com.ibm.bpe.api.ClientObjectWrapper object. This ClientObjectWrapper object contains a WSIFDefaultMessage object. To receive this message object, the getObject() method is called.

3. Access the message parts. For example, a process output message might have the following structure:

outputMessage	java.lang.String
flowID	int

   You can access the parts as shown in the following code snippet:

   ```
   String outputMessage = (String)msg.getObjectPart("outputMessage");
   int flowID = msg.getIntPart("flowID");
   ```

Creating JSP files for processing user input

User input fields often require additional information to enable users to understand the purpose of these fields. You can use user-defined JavaServer Pages (JSP) files to provide additional information about message parts in the standard Web client. A typical user-defined JSP file for displaying user input data can provide a detailed description for each type of input field including entry fields, check boxes, and radio buttons. General information about the process or the activity to which the input data is related can also be displayed.

1. Make the BusinessProcess bean available to the JSP file.

   ```
   BusinessProcessService process =
           MessageUtilities.getBusinessProcessService(request);
   ```

   You can use the BusinessProcessService interface to develop applications that work with both the remote and the local object of the bean.

2. Display additional information about the process or activity. The following code snippet is part of a user-defined JSP file that uses information about the current activity to display the activity output message. Depending on the state of the activity, the JSP file displays the information in different ways:
   - When the activity is in the ready state, general text about the purpose of the activity is displayed.
   - When the activity is claimed, two radio buttons that correspond to true and false, respectively, are also displayed.
   - After the activity finishes, the chosen option is presented as text.

   The highlighted text shows the lines of code that provide this functionality:

```
<%
String aiid = request.getParameter("WF_AIID");
BusinessProcessService process =
        MessageUtilities.getBusinessProcessService(request);
ActivityInstanceData activity = process.getActivityInstance(aiid);
%>
<p>
      A user has placed a stock order with a total estimated purchase price
      of more than $100000.
</p>
<%}if(activity.getExecutionState() == ActivityInstanceData.STATE_READY){%>
<p>
      This order must be approved.
</p>
<%}if(activity.getExecutionState() == ActivityInstanceData.STATE_CLAIMED){%>
<p>
 This order must be approved. What do you want to do?
 <label>
  <input type="radio" name="Approved" value="true">
  Approve the order.
 </label>
 <label>
  <input type="radio" name="Approved" value="false" checked>
  Reject the order.
 </label>
</p>
<%}if(activity.getExecutionState() == ActivityInstanceData.STATE_FINISHED){>
  if (outMsg.getBooleanPart("approved")){%>
<p>
      This order has been approved.
</p>
<%      } else {%>
<p>
      This order has not been approved.
</p>
<%      }
  } %>
```

Create a message-mapping JSP file for each user-defined JSP file that processes user input.

Integrating user-defined JSP files in a V5.0-style process model

To include user-defined JavaServer Pages (JSP) files in the Web client, you must specify them when you model a business process in WebSphere Studio Application Developer Integration Edition.

1. Include user-defined JSP files for a process. If you have created user-defined JSP files for a process:
   a. In the process editor, select the Client tab, then click **Add** to add user-defined settings to the process definition. The Add Definition dialog box appears.
   b. Select the action for which you want to specify user-defined settings. Fields for the user-defined JSP files appear.
   c. Select the JSP files that you want to use for the specified action. The specified JSP files must be either part of the service project that holds the current process or part of another Web project in WebSphere Studio. If these files are part of another project, this Web project must be part of the Enterprise application project that holds the service project.

      You can specify a subset of JSP files. For example, you can specify only an output-message JSP. In this case, the default Web client settings are used for

the input message. If you specify a user-defined JSP file that accepts user input, you must also specify a message-mapping JSP file.

2. Include user-defined JSP files for activities. If you have created user-defined JSP files for activities:
   a. Open the process containing the activities in the Process Editor and select the **Process** tab.
   b. Double-click the activity for which you want to specify user-defined JSP files. The Properties dialog box for the activity appears.
   c. In the navigation pane, click **Client**.
   d. Click **Add** to add user-defined Web client settings to the activity. The Add Definition dialog box appears.
   e. Select **ActivityDisplay** and specify the appropriate JSP files. For activities involving human interaction, you can specify JSP files only for ActivityDisplay.

Message-mapping JSP files

Message-mapping JavaServer Pages (JSP) files are required whenever user input fields are displayed by user-defined JSP files. Message-mapping JSP files:

- Receive the data that is provided by the user
- Process data-type checks and validation
- Wrap the data in an appropriate message
- Forward the data

Receiving data

Message data that is provided by the user in the input fields of a user-defined JSP file can be received by the javax.servlet.http.HttpServletRequest object in the subsequent message-mapping JSP file. The following code snippet shows how to access user input data in a user-defined JSP file:

```
String symbol = request.getParameter("Symbol_Input");
String number = request.getParameter("Number_Input");
String limit = request.getParameter("Limit_Input");
[...]
```

Processing data

When all the user input is available, it is appropriate to perform type checks and validate the input data. For example, some input fields might be mandatory, or some input fields might be required to accept values in a certain range or format only. You can use a message-mapping JSP file to do these checks.

Examples of type checking and data validation are given in the following code snippet:

```
if (number.equals("")) {
 throw new ServletException("The required field 'number' has not been completed.
                            Please specify a number.");
}


try {
 int intValueOfNumber = Integer.valueOf(number)).intValue();
} catch (NumberFormatException ex) {
 throw new ServletException("The specified number '"+number+"' is not valid. ");
}
```

When you check and validate input data, it is recommended that you use the built-in error feature of JSP files to display errors that occur. Define a user-defined error page, `error.jsp` in the page header of the message-mapping JSP file:

```
<%@ page errorPage="error.jsp" %>
```

Wrap the exceptions that occur when the data is checked in a ServletException exception. Whenever a ServletException exception (wrapping an error that occurred during data validation and checking) occurs, the error page is displayed. Detailed information about the error can be displayed on the error page, for example, `The required field 'user id' has not been completed`. Users can use the browser back button to return to the Web client. They can correct their entries and then send a message (an input or output message of a process or an activity) with the corrected data.

Wrapping data

After the user input is received and validated, the data must be wrapped in a message object that can be processed by the business process container.

You can use an org.apache.wsif.base.WSIFDefaultMessage class for all messages. If message data is to be retrieved, for example, it is always wrapped in a WSIFDefaultMessage class.

Suppose a process input message has the following structure:

symbol	java.lang.String
number	int
limit	double
id	java.lang.String

The following code snippet shows how to create and fill a WSIFDefaultMessage class that wraps these message parts.

```
WSIFDefaultMessage wsifProcessInMessage = new WSIFDefaultMessage();
wsifProcessInMessage.setObjectPart("symbol", symbol);
wsifProcessInMessage.setIntPart("number", intValueOfNumber);
wsifProcessInMessage.setDoublePart("limit", doubleValueOfLimit);
wsifProcessInMessage.setObjectPart("id", id);
```

Forwarding data

To send a message to the business process container, you can use the forwardMessageToController() static method in the com.ibm.bpe.client.MessageUtilities class. The signature of this method is:

```
public static void forwardMessageToController(
          javax.servlet.http.HttpServletRequest request,
          javax.servlet.http.HttpServletResponse response,
          java.io.Serializable message,
          java.util.Map furtherParameters,
          java.util.Vector excludeRequestParameters)

          throws javax.servlet.ServletException,
                 java.io.IOException
```

- For the request and response parameters, use the implicit HttpServletRequest and HttpServletResponse objects that are available in the current JSP file.
- For the message parameter, use a com.ibm.bpe.api.ClientObjectWrapper object that contains a WSIFDefaultMessage object.

- 5.1 + furtherParameters and excludeRequestParameters parameters are not supported in Version 5.1.

The following code snippet shows how a process input message might be forwarded:

```
ClientObjectWrapper messageWrapper = new ClientObjectWrapper(message);

  //where message is the WSIFMessage instance that was previously populated

MessageUtilities.forwardMessageToController(request, response, messageWrapper,
                                            null, null);
```

Troubleshooting the process choreographer Web client

You have encountered an error while using the process choreographer Web client. Further information might be available to help you resolve the problem you encountered. This information is available in English only.

The error page contains a link to the technical support search page.

1. Search for the error code on the IBM technical support pages.
   - Click the **Search for more information** link. This starts a search for the error code on the IBM technical support site.
   - Copy the error message code that is shown on the Web client Error page to the clipboard. The error code has the format `BPEcnnnnc`, where each `c` is a character and `nnnn` is a 4-digit number. Go to the technical support search page. Paste the error code into the **Limit by adding search terms** field and click **Go**.

   If the search does not return any documents or if the documents do not help you solve your problem, you can provide IBM with information about your error situation so that the technical support information can be updated for future reference.
2. To provide information on the error, go to the technical support search page and open the first document in the list. This information is used to update the technical support information only. To receive a response to your problem, submit a service request.

Process choreographer Web client page directory

The process choreographer Web client comprises pages that you can use to work with business processes and work items.

All Web client pages contain the following common features:

- Navigation pane

  This pane contains links to lists of work items, process instances, and process templates:
  - Work Item Lists

    Contains links for working with work items in your To Do list and for specifying criteria for queries on work items.
  - Process Instance Lists

    Contains links for working with process instances that you administer or created, and for specifying criteria for queries on process instances.
  - Process Template Lists

Contains links for working with process templates and for specifying criteria for queries on process templates.

– 5.1 + Administration

Contains links for administering work items.

- Content pane

  You can use this pane to work with process instances and work items. The content depends on the actions that you have taken with the Web client. When you first start the Web client, this page displays the work items in your To Do list.

Activity page

Use this page to work with a work item and to display more information about its corresponding activity or its business process.

This page is displayed if you click a work item in the My To Dos page or the Process Instance page.

The Activity page has the following sections:

Available Actions

Actions that you can perform on the work item. An action is available only if you have the authority to act on the work item in its current state. If you are not authorized to act on the work item in its current state, this section is empty. For example:

Claim Displayed if the work item is in a Ready state and you are a potential owner of the work item. If you click **Claim**, you become the work item owner (its state is changed to Claimed) and are responsible for its completion.

Complete

Displayed if the work item is in a Claimed state and you are the owner of the work item. If you specified the required input, you can click **Complete** to complete the work item.

Save Changes

Displayed if the work item is in a Claimed state and you are the owner of the work item. If you change the output message properties, you can click **Save Changes**, to save the changes without completing the work item. For example, you might save changes to a work item if your work is interrupted or if you need more information.

If you click **Save Changes**, the Activity page is replaced by the My To Dos page. You can display the Activity page later to complete the work item.

Cancel

Displayed if the work item is in a Claimed state and you are the owner of the work item. You can click **Cancel** to leave the work item without saving any changes that you might have made.

Process Context

This section is displayed only if you are authorized to view information about the business process. It shows information about the work item and its associated process instance and process template. It also provides the following links for additional information about the corresponding activity and its process:

- View more details about this activity
- View more details about this process

Activity Input Message
This section displays information about the activity. The content of this section, including the number and types of input fields, depends on the design of the activity in the process template.

Activity Output Message
This section displays fields in the output message. The content of this section, including the number and types of input fields, depends on the design of the activity within the process template.

You can change the fields in this section only if you are the owner or an editor of the work item.

When you click **Complete** to complete this work item, the information in the activity output message is stored. This information can be used by any of the subsequent activities in the process.

Activity Information page

Use this page to view details about the activity and its process.

This page is displayed if you click a link for more information about an activity; for example, on the Activity page.

The Activity Information page has the following sections:

Activity Description
This section shows detailed information about an activity. The content of this section depends on the design of the activity in the process template, the state of the activity, and your authority for the activity. For example, this section can display the following information:

Activity Name
The name of the activity.

Description
A short description of the activity.

Potential Owners
A list of the user IDs for potential owners of the work item associated with this activity.

This section also provides a link for you to work with this activity.

Process Description
This section is displayed only if you are authorized to view information about the business process. This section provides information about the properties of the business process to which the activity belongs. The content of this section depends on the design of the activity within the process template. For example, this section can display the following information:

Process Instance Name
The identifier for the process.

Description
A brief description of the process.

Template Name
The name of the business process template from which the process was started.

Starter
The user ID that started the process.

State The current state of the process.

Started
The date and time when the process was started.

This section also provides a link for you to view more details about this process.

Administered By Me page

Use this page to display information about each process instance for which you are the process administrator. You can select a process instance to work on.

To administer a process instance, select the check box next to the process instance and click one of the Available Actions.

This page is displayed if you click **Administered By Me** under Process Instance Lists in the navigation pane.

Available Actions
Actions that you can perform on selected process instances. An action is available only if you have the authority to act on the selected processes in their current states.

View Click **View** to display additional information about the process instance.

Compensate
Click **Compensate** to terminate the process instance and start compensation.

Terminate
Click **Terminate** to discontinue the navigation of the process instance.

Delete Click **Delete** to explicitly delete a finished process instance that is not automatically deleted when it completes, or a process that was terminated by a user.

Monitor
Click **Monitor** to view the progress of the activities in the process instance.

Repair Compensation
Click **Repair Compensation** to view failed compensation actions for the process instance.

Process
The business process identifier for the process. Click the link to display the Process Instance page, which provides additional information about the process instance.

Template Name
The name of the process template to which the process instance belongs.

State The current state of the process instance.

Started
The date and time that the process instance started.

Process Starter
The user ID of the person who started the process instance.

Reason
The role of the person who started the process instance.

Create Work Items (for an activity) page

Use this page to create work items for an activity.

To create a work item for an activity, provide the required information and click **Create**.

This page is displayed if you click **Create** on the Manage Work Items for Activities page. This page is available for BPEL processes only.

Available Actions

Actions that you can perform on the activity instance. An action is available only if you have the authority to act on the selected activity in its current state.

Create Click **Create** to create work items for the activity instance.

Create for

Specify the user ID of the owner of the new work item:

- If the current work-item owner is the process administrator, you can specify any user ID.
- If the current work-item owner is not the process administrator, you can specify only a user ID that is defined in the process template.

Reason

Select one or more roles from the list. These roles determine the actions the assigned person can perform on the new work item.

Create Work Items (for process instances) page

Use this page to create work items for a process instance.

To create a work item for a process instance, provide the required information and click **Create**.

This page is displayed if you click **Create** on the Manage Work Items for Process Instances page. This page is available for BPEL processes only.

Available Actions

Actions that you can perform on the process instance. An action is available only if you have the authority to act on the process in its current state.

Create Click **Create** to create work items for the specified work-item owner.

Create for

Specify the user ID of the owner of the new work item:

- If the current work-item owner is the process administrator, you can specify any user ID.
- If the current work-item owner is not the process administrator, you can specify only a user ID that is defined in the process template.

Reason

Select one or more roles from the list. These roles determine the actions the assigned person can perform on the new work item.

Created By Me page

Use this page to display information about each process instance that you start and to select a process on which to work.

To work on a process instance, select the check box next to the process instance and click one of the Available Actions.

This page is displayed if you click **Created By Me** under Process Instance Lists in the navigation pane.

Available Actions

Actions that you can perform on selected process instances. An action is available only if you have the authority to act on the selected processes in their current states.

View Click **View** to display additional information about the process instance.

Compensate
Click **Compensate** to terminate the process instance and start compensation.

Terminate
Click **Terminate** to discontinue the navigation of the process instance.

Delete Click **Delete** to explicitly delete a finished process instance that is not automatically deleted when it completes, or a process that was terminated by a user.

Monitor
Click **Monitor** to view the progress of the activities in the process instance.

Repair Compensation
Click **Repair Compensation** to view failed compensation actions for the process instance.

Process
The identifier for the process instance. Click the link to display the Process Instance page, which provides additional information about the process instance.

Template Name
The name of the process template to which the process instance belongs.

State The current state of the process.

Started
The date and time that the process started.

Reason
The role of the person who started the process instance.

Define Process Instance List page

Use this page to specify criteria for querying process instances.

This page is displayed if you click **Define Process Instance List** under Process Instance Lists in the navigation pane.

To see the list of process instances that match all of your selection criteria, click **Show List**. This list shows only those process instances that you are authorized to see.

State

Specifies the current state of the process instance. To select more than one state, hold down the Ctrl key and click the states that you want to include in your query.

Running
The process instance has started.

Finished
The process instance has completed normally.

Failed The running process instance has encountered an unhandled fault or a fault node. If compensation is not defined for the process, navigation of the process stops.

Terminated
The process instance is stopped because either its parent process has failed or it has received an explicit terminate request.

Compensated
The failed process instance has been compensated and is now in its end state.

Process instances that have an activity in state stopped: Select this check box to include process instances with an activity in stopped state in the query.

Starter
The principal ID of the starter of the process instance. You can use the wildcard character, percent (%), to represent one or more characters. For example, to find all the users whose IDs start with B, you can type `B%` here.

Template Name
The name of the process template associated with the process instance. You can use the wildcard character, percent (%), to represent one or more characters. For example, to find all the process templates that start with P, you can type `P%` here.

Started
Specifies when the process instances started. You can specify that the process started before or after a certain date or within a given range of dates.

Define Template List page

Use this page to specify criteria for querying process templates.

This page is displayed if you click **Define Template List** under Process Template Lists in the navigation pane.

To see the list of process templates that match all of your selection criteria, click **Show List**. This list shows only those process templates that you are authorized to see.

State
Specifies the current state of the process template. To select more than one state, hold down the Ctrl key and click the states that you want to include in your query.

Started
The process template is started.

Stopped
The process template is stopped.

Template Name
The name of the process template. You can use the wildcard character, percent (%), to represent one or more characters. For example, to find all the process templates that start with P, you can type `P%` here.

Valid From
Specifies when the process template started. You can specify that the process template started before or after a certain date or within a given range of dates.

Define Work Item List page

Use this page to specify criteria for querying work items.

This page is displayed if you click **Define Work Item List** under Work Item Lists in the navigation pane.

To see the list of work items that match all of your selection criteria, click **Show List**. This list shows only those work items that you are authorized to see.

State

Specifies the current state of the work item. To select more than one state, hold down the Ctrl key and click the states that you want to include in your query.

Ready The work item is ready to be claimed.

Claimed
The work item has been accepted by one of its potential owners.

Stopped
The work item has produced an unhandled fault.

Expired
The work item was not completed within the time allocated.

Finished
The work item has been completed.

Terminated
The process to which this work item belongs has been terminated.

Failed The process to which this work item belongs has failed.

Owner

The principal ID of the owner of the work item; only claimed or finished work items have owners. You can use the wildcard character, percent (%), to represent one or more characters. For example, to find all the users whose IDs start with B, you can type `B%` here.

Activated

Specifies when the work item was activated. You can specify that the work item was activated before or after a certain date or within a given range of dates.

Delete Work Items (for activities) page

Use this page to delete work items for activities.

You might want to delete work items, for example, if you have created a work item in error or if a work item has been generated for someone who no longer works in a particular department. To delete work items for activity instances, select one or more work items, and click **Delete**.

This page is displayed if you click **Delete** in the Manage Work Items for Activities page. This page is available for BPEL processes only.

Available Actions
Actions that you can perform on the selected work items. An action is available only if you have the authority to act on the selected activity instances in their current states.
Delete Click **Delete** to delete the selected work items.

To Do Name
The name of the work item.

Work Item Owner
The current work-item owner.

Reason
The reason why the work-item was assigned to the current owner.

Activity ID
The ID of the activity instance with which the work item is associated.

Process
The identifier of the process.

Template Name
The name of the process template to which the process instance belongs.

Delete Work Items (for process instances) page

Use this page to delete all of the work items for a process instance.

You might want to delete work items, for example, if you have created work items in error or if work items have been generated for someone who no longer works in a particular department. To delete work items for a process instance, select one or more process instances, then click **Delete**.

This page is displayed if you click **Delete** in the Manage Work Items for Process Instances page. This page is available for BPEL processes only.

Available Actions

Actions that you can perform on the selected process instances. An action is available only if you have the authority to act on the selected process instances in their current states.

Delete Click **Delete** to delete the selected work items.

Process

The business process identifier for the process instance.

Work Item Owner

The current work-item owner.

Reason

The reason why the work-item was assigned to the current owner.

Template Name

The name of the process template to which the process instance belongs.

Manage Activities for Process Instances page

Use this page to select process instances for which you want to manage activity work items, for example, to transfer a work item to another work-item owner.

Select the check box next to one or more process instances that you want to work with and click **Show Activities**.

This page is displayed if you click **Manage Activities for Process Instances** under Administration in the navigation pane. This page displays only processes instances that belong to BPEL processes.

Available Actions

Actions that you can perform on selected process instances. An action is available only if you have the authority to act on the selected process instances in their current states.

Show Activities

Click **Show Activities** to show the activities for the selected process instances.

Process

The business process identifier for the process. Click the link to display the Process Instance page, which provides additional information about the process instance.

Template Name

The name of the process template to which the process instance belongs.

State The current state of the process.

Started

The date and time that the process was started.

Process Starter

The user ID of the person who started the process instance.

Manage Work Items for Activities page

Use this page to select activity instances for which you want to create, transfer, or delete work items.

To manage the work items for activity instances, select the check box next to one or more activities and click one of the Available Actions.

This page is displayed if you click **Show Activities** on the Manage Activities for Process Instances list page. This page displays only activity instances that belong to BPEL processes.

Available Actions
Actions that you can perform on the selected activity instances. An action is available only if you have the authority to act on the selected activities in their current states.

Create Click **Create** to create work items for the selected activity instance. This action is available when only one activity instance is selected.

Transfer
Click **Transfer** to transfer work items to another work-item owner.

Delete Click **Delete** to delete work items.

To Do Name
The work item name. Click the link to display the Activity page, which provides additional information about the associated activity.

Process
The business process identifier for the process. Click the link to display the Process Instance page, which provides additional information about the process instance.

Template Name
The name of the process template to which the process instance belongs.

Manage Work Items for Process Instances page

Use this page to select process instances for which you want to create, transfer, or delete work items.

To manage the work items for a process instance, select the check box next to one or more process instances and click one of the Available Actions.

This page is displayed if you click **Manage Work Items for Process Instances** under Administration in the navigation pane. This page displays only processes instances that belong to BPEL processes.

Available Actions
Actions that you can perform on selected process instances. An action is available only if you have the authority to act on the selected process instances in their current states.

Create Click **Create** to create work items for the selected process instance. This action is available when only one process instance is selected.

Transfer
Click **Transfer** to transfer work items to another work item owner.

Delete Click **Delete** to delete work items.

Process
The business process identifier for the process. Click the link to display the Process Instance page, which provides additional information about the process instance.

Template Name
The name of the process template to which the process instance belongs.

State The current state of the process.

Started
The date and time that the process started.
Process Starter
The user ID of the person who started the process instance.

My Templates page

Use this page to view process templates with which you can work. If you are authorized, you can start process instances from this page. To act on a process template, select the check box next to the process template and click one of the Available Actions.

This page is displayed if you click **My Templates** under Process Templates Lists in the navigation pane.

Available Actions
Select this option to work on selected process templates. This action is available only if you have the authority to act on the selected templates in their current states.

View Click **View** to display additional information about the process template.

Start Instance
Click **Start Instance** to start a process instance from the selected process template. This action displays the Process Input Message page, where you can specify the properties of the input message before you start the process instance.

5.1 + For a process based on the Business Process Execution Language (BPEL) that contains more than one operation that you are authorized to start, this action displays the Process Template page. On this page, select a service and click **Start Instance**. This action displays the Process Input Message page.

Template Name
The name of the process template. To view additional information about the process template, click the link to display the Process Template page.
Valid From
The date and time when the template was first used.
Can Run Interrupted
Specifies whether process instances derived from this template are interruptible.
Delete on completion
Specifies whether process instances derived from this template are automatically deleted when they complete.
State The state of the process template.
Description
A brief description of the process.

My To Dos page

Use this page to work on work items in your To Do list. You can also view additional information on work items and their processes from this page.

To work on one or more work items, select the check box next to the work item and click one of the Available Actions.

This page is displayed if you click **My To Dos** under Work Item Lists in the navigation pane.

Available Actions

Actions that you can perform on selected work items. An action is available only if you have the authority to act on the selected work items in their current states.

View Click **View** to display additional information about the work item.

Claim Available if the work item is in a Ready state and you are a potential owner of the work item. If you click **Claim**, you become the work item owner and are responsible for its completion. The state of the work item changes to Claimed.

Complete

Available if the work item is in a Claimed state and you are the owner of the work item. If you have specified the required input, you can click **Complete** to complete the activity.

Cancel

Available if the work item is in a Claimed state and you are the owner of the work item. You can click **Cancel** to leave the work item without saving any changes you might have made.

Restart

Available if the work item is in a Stopped state and you are the process administrator for the process instance.

Force Complete

Available if the work item is in a Stopped state and you are the process administrator for the process instance.

Transfer

Available only for work items that belong to BPEL processes. In addition, the work item must be in a Claimed state and you must be either the owner of the work item or the process administrator for the process instance.

To Do Name

The name of the work item. You can click this link to work with the work item.

State The current state of the work item.

Work items in the Ready state appear on the My To Dos page of all potential work item owners. If you claim such a work item, you become the work item owner (its state is changed to Claimed) and are responsible for its completion.

Owner

If the work item is in a Claimed state, the owner of the work item is displayed here.

Reason

The reason why the work item is assigned to you.

Activated

The date and time that the work item was activated.

Process

The process instance to which the work item belongs. Click this link to work with the process instance.

Template Name

The name of the process template from which the process instance was started. Click this link to work with the process template.

Process Input Message page

Use this page to change the input message before you complete the actions to start the business process.

This page is displayed when you start a process that requires an input message from the Process Template page.

This page has the following sections:

- Available Actions

  Click **Start Instance** to start a process instance with the information shown on this page.
- Process Template Description

  This section provides information about the properties of the process template from which the process is started.
- 5.1+ Service

  For BPEL-based processes, this section provides information about the starting service.
- Process Instance Name (optional)

  The name of the new process instance. This field is displayed for interruptible processes only.
- Process Input Message

  This section provides entry fields for the process input message. The number and types of fields depend on the process template.

Process Instance page

Use this page to display information about the process and, optionally, to act on the process instance. You can also view the activities for the process instance and select an activity with which to work.

This page is displayed if you click a process instance link, for example, on the Administered By Me page or the Created By Me page, or a link for more information about a process.

This page contains the following sections:

Available Actions

Actions that you can perform on the process instance. An action is available only if you have the authority to act on the process instance in its current state. If you are not authorized to act on the process instance in its current state, this section is empty.

Compensate

Click **Compensate** to terminate the process instance and start compensation.

Terminate

Click **Terminate** to discontinue the navigation of the process instance.

Delete Click **Delete** to explicitly delete a finished process instance that is not automatically deleted when it completes, or a process that was terminated by a user.

Monitor

Click **Monitor** to view the progress of the activities in the process instance.

Repair Compensation

Click **Repair Compensation** to view failed compensation actions for the process instance.

Process Description

Information about the process, including:

Process Instance Name

The identifier of the process.

Template Name
The name of the business process template from which the process instance was started.
Description
A brief description of the process.
Starter
The user ID that started the process instance.
State The current state of the process instance.
Started
The date and time when the process instance started.

Process Input Message
Input provided when the process instance was started.
Process Output Message
If the process has finished, its output message is displayed here.

5.1 + If the process is a BPEL-based process and it implements a one-way operation, an output message is not displayed.

Activities
Information about the activities that make up the process:
Name The name of the activity. Click this link to work with the activity.
State The current state of the activity.
Owner
The owner of the activity.
Activated
The date and time when the activity was started.
Completed
The date and time when the activity was completed.

Events
For BPEL-based processes, information about the events that make up the process

Process Instance Monitor page

Use this page to view the status of activities in a process instance. You can also view more detailed information on activities from this page.

To work on an activity, click its activity name in the **To Do Name** list. This action displays the Activity page, which provides information about the activity and, if appropriate, provides options for you to act on the activity.

This page is displayed if you click **Monitor** on the Administered By Me page, the Created By Me page, the Process Instance page, or the User-Defined Process Instance List page.

This page contains the following sections:
Process Description
Information about the process instance, including the starter of the process, the process readers, and the process administrators.
Activities
The activities that the process instance contains with information about each activity. Click an activity to see additional information.

Process Output Message page

5.1 + Use this page to view the results of a business process that you started. This page is displayed when a non-interruptible process ends. Not all BPEL-based

processes have output messages, for example, if the process implements a one-way operation, an output message is not displayed.

The information displayed by the output message depends on the process template that defines the model for the process.

Process Template page

Use this page to view information about a process template, which provides the design for business processes.

This page is displayed if you click a process template on the My Templates page.

Process Template Description
Provides information about the process template. This information includes when the template was first used, whether process instances based on the template are interruptible, and if these process instances are automatically deleted when they complete.

Services
For processes based on the Business Process Execution Language (BPEL), services that you are authorized to run. To start a process instance, click one of the services and complete the information on the Process Input Message page.

Transfer Work Items (for activities) page

Use this page to transfer work items for selected activity instances to another work-item owner.

To transfer work items for an activity instance, select one or more work items, specify the new work-item owner in the **Transfer to** field, and click **Transfer**.

This page is displayed if you click **Transfer** in the Manage Activities for Process Instances page. This page is available for BPEL processes only.

Available Actions
Actions that you can perform on the selected work items. An action is available only if you have the authority to act on the selected work items in their current states.

Transfer
Click **Transfer** to transfer work items to another work-item owner.

Transfer to
Specify the user ID of the new owner of the work item:
- If the current work-item owner is the process administrator, you can specify any user ID.
- If the current work-item owner is not the process administrator, you can specify only a user ID that is defined in the process template.

To Do Name
The name of the work item.

Work Item Owner
The current work-item owner.

Reason
The reason why the current work-item owner was assigned the work item.

Activity ID
The identifier for the activity instance.

Process
The business process identifier for the process.

Template Name
The name of the process template to which the process instance belongs.

Transfer Work Items (for process instances) page

Use this page to transfer work items for selected process instances to another work-item owner.

To transfer work items, select one or more process instances, specify the new work-item owner in the **Transfer to** field, then click **Transfer**.

This page is displayed if you click **Transfer** in the Manage Work Items page. This page is available for BPEL processes only.

Available Actions

Actions you can perform on the selected work items. An action is available only if you have the authority to act on the selected work items in their current states.

Transfer

Select **Transfer** to transfer work items to another work-item owner.

Transfer to

Specify the user ID of the new owner of the work item:

- If the current work-item owner is the process administrator, you can specify any user ID.
- If the current work-item owner is not the process administrator, you can specify only a user ID that is defined in the process template.

Process

The business process identifier for the process instance.

Work Item Owner

The current work-item owner.

Reason

The reason why the current work-item owner was assigned the work item.

Template Name

The name of the process template to which the process instance belongs.

Undo Actions in Error page

Use this page to act on compensation actions that failed.

To administer compensation, select the check box next to the process instance and click one of the Available Actions.

5.1+ This page is displayed if you click **Undo Actions in Error** under Process Instance Lists in the navigation pane or **Repair Compensation** on any of the pages that display process instances.

Available Actions

The compensation actions that you can perform. An action is available only if you have the authority to act on the selected process instance in its current state.

Skip Compensating Action

Skip the current compensating action and continue with compensating the process instance.

This action might result in a non-compensated activity.

Retry Compensating Action

If you acted to correct the failed compensation action, click **Retry Compensating Action** to try the compensation action again.

Stop Compensation

Stop the compensation process.

Compensation Action for Activity

The activity for which the failed compensation action is defined.

Reason
The reason the compensation action failed. This information can help you decide what actions to take to correct the failed compensation action.

Created
The date and time that the compensation action was created.

Started
The date and time that the compensation action started.

Finished
The date and time that the compensation action finished.

Process Context
Information that identifies the process instance to which the activity belongs.

User-Defined Process Instance List page

Use this page to work with process instances found by a query. If you are authorized as the process choreographer system administrator, you can also save the query so that it appears under Process Instance Lists. To save the query, type a unique name for the query in the **Save List As** field and click **Save**.

This page is displayed if you click **Show List** in the Define Process Instance List page.

This page has the following sections:

Available Actions
Actions that you can perform on the process. An action is available only if you have the authority to act on the process in its current state.

View Click **View** to see additional information about the selected process instance.

Compensate
Click **Compensate** to terminate the process instance and start compensation.

Terminate
Click **Terminate** to discontinue the navigation of the process instance and stop any activities that are running.

Delete Click **Delete** to explicitly delete a finished process instance that is not automatically deleted when it completes, or a process that was terminated by a user.

Monitor
Click **Monitor** to view the progress of the activities in the process instance.

Repair Compensation
Click **Repair Compensation** to view failed compensation actions for the process instance.

Process
The identifier of the process.

Template Name
The name of the business process template from which the process was started.

State The current state of the process.

Started
The date and time when the process started.

Process Starter
The user ID that started the process.

Reason
The role of the process starter.

User-Defined Process Template List page

Use this page to work with process templates found by a query.

This page is displayed if you click **Show List** in the Define Template List page.

Available Actions
Actions that you can perform on the selected process template. An action is available only if you have the authority to act on the selected process template in its current state.

View Click **View** to see additional information about the selected process template.

Start Instance
If you are authorized to start a process from this template, the **Start Instance** action is available. To start a process, click **Start Instance**.

Template Name
The name of the process template.

Valid From
The date and time when the template was first used.

Can Run Interrupted
Whether processes derived from the template are interruptible.

Delete on completion
Whether processes derived from the template are automatically deleted on completion.

State The state of the process template.

Description
A short description of the process template.

User-Defined Work Item List page

Use this page to work with the work items found by a query. If you are authorized as the process choreographer system administrator, you can also save the query so that it appears under Work Item Lists.

From this page you can:

- Work with work items. To act on a work item, select the check box next to the work item and click one of the Available Actions.
- Work with a process instance. Click the process name under **Process** in the list. This action displays the Process Instance page, which provides additional information about the process and, if appropriate, provides options for you to act on the process instance.
- Work with a process template. Click the process template name under **Template Name** in the list. This action displays the Process Template page, which provides additional information about the process template and, if appropriate, provides options for you to act on the process template.
- Save the work item list. Type a unique name for the list in the **Save List As** field, then click **Save**. The work item list then appears under My Lists.

This page is displayed if you click **Show List** in the Define Work Item List page.

This page has the following sections:

Available Actions
Actions that you can perform on the selected work items. An action is available only if you have the authority to act on the selected work items in their current states.

View Click **View** to display additional information about the work item.

Claim Available if the work item is in a Ready state and you are a

potential owner of the work item. If you click **Claim**, you become the work item owner (its state is changed to Claimed) and are responsible for its completion.

Complete
Available if the work item is in a Claimed state and you are the owner of the work item. If you have already specified the required user input, you can click **Complete** to complete the work item.

Cancel
Available if the work item is in a Claimed state and you are the owner of the work item. You can click **Cancel** to leave the work item without saving any changes you might have made.

Restart
Available if the work item is in a Stopped state and you are the process administrator for the process instance.

Force Complete
Available if the work item is in a Stopped state and you are the process administrator for the process instance.

Transfer
Available only for work items that belong to BPEL processes. In addition, the work item must be in a Claimed state and you must be either the owner of the work item or the process administrator for the process instance.

To Do Name
The name of the work item. You can click this link to work with the work item.

State The current state of the work item.

Owner
If the work item is in a Claimed state, the owner of the work item is displayed here.

Reason
The reason why the work item was assigned to the owner.

Activated
The date and time that the work item was activated.

Process
The name of the process instance to which the work item belongs. You can click this link to work with the process instance.

Template Name
The name of the template that contains the process.

Process choreographer Web client roles and actions

The actions you can take with the Web client depend on the role that you have been assigned; there are roles for processes and roles for work items. You cannot delete the user ID of the process starter from your user registry while the process instance exists. If you do so, the navigation of this process cannot continue. You receive the following exception in the system log file:

```
no unique ID for: <user ID>
```

5.1 +

Table 1. Actions for process roles

Action	Business process administrator	Process administrator	Starter	Potential owner	Reader
Start process	Yes		See note	Yes	

Table 1. Actions for process roles (continued)

Action	Business process administrator	Process administrator	Starter	Potential owner	Reader
Terminate process	Yes	Yes			
Monitor process (5.0.2 and later)	Yes	Yes			Yes
Delete process	Yes	Yes			
Display process	Yes	Yes	Yes		Yes
Save user-defined lists (5.0.2 and later)	Yes				
Force complete stopped activity	Yes	Yes			
Force restart activity	Yes	Yes			
Claim work item	Yes	Yes			
Save work item	Yes	Yes			
Display work item	Yes	Yes			
Complete work item	Yes	Yes			
Transfer work item (5.1 and later)	Yes	Yes			
Create work item (5.1 and later)	Yes	Yes			
Delete work item (5.1 and later)	Yes	Yes			
Note: If a potential owner of a starting activity starts a process, this person becomes the *starter* of the process. If a potential owner claims a work item, the potential owner becomes the *owner* of the work item.					

5.1 +

Table 2. Actions for work-item roles

Action	Business process administrator	Potential owner	Owner	Reader	Editor
Claim work item	Yes	See note 1	See note 2		
Transfer work item (5.1 and later)	Yes		Yes		
Save work item	Yes	Yes	Yes		Yes
Display work item	Yes	Yes	Yes	Yes	Yes

Table 2. Actions for work-item roles (continued)

Action	Business process administrator	Potential owner	Owner	Reader	Editor
Complete work item	Yes		Yes		
Force restart work item	Yes				
Force complete stopped work item	Yes				
Create work item (5.1 and later)	Yes				
Delete work item (5.1 and later)	Yes				
Note: 1. If a potential owner claims a work item, the potential owner becomes the owner of the work item. 2. An owner of a work item can transfer the work item only to a user that is a potential owner of the associated activity.					

Chapter 9. Developing applications for BPEL-based processes

The process choreographer API provides the following renderings for developing applications for processes based on the Business Process Execution Language (BPEL):

- An Enterprise JavaBeans (EJB) rendering that allows the API to be called locally or remotely. There is a stateless session bean for each type of call (LocalBusinessProcess and BusinessProcess) that exposes the functions that an application program can call. The BusinessProcessService interface provides a common interface for these session beans.
- A Java Message Service (JMS) rendering that allows a subset of the API functions to be called remotely using JMS.

You can use the API to develop the following types of applications:

- Business process applications for non-interruptible processes
- Business process applications for interruptible processes
- Administration applications for interruptible processes

For information on how to handle faults, see handling exceptions and faults.

For more information, see the Javadoc.

Accessing the process choreographer remote EJB interface

An application accesses the BusinessProcess session bean through the home and remote interfaces of the bean.

1. Add a reference to the BusinessProcess session bean in the application deployment descriptor. Add the reference to:
   - The `application-client.xml` file, for a Java 2 Platform Enterprise Edition (J2EE) client application
   - The `web.xml` file, for a Web application
   - The `ejb-jar.xml` file, for an Enterprise JavaBeans (EJB) application

   Add the reference to the remote home interface as in the following example:

   ```
   <ejb-ref>
    <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.bpe.api.BusinessProcessHome</home>
    <remote>com.ibm.bpe.api.BusinessProcess</remote>
   </ejb-ref>
   ```

2. Package the generated stubs with your application. If your application runs on a different Java Virtual Machine (JVM) from the one where the BPEContainer application runs.
   a. Package the files contained in the `WebSphere/AppServer/ProcessChoreographer/client/bpe137650.jar` file with the enterprise archive (EAR) file of your application.
   b. Set the **Classs-Path** parameter in the manifest file of the application module to include the `bpe137650.jar` file. The application module can be a J2EE application, a Web application, or an EJB application.
3. Make the home interface of the BusinessProcess session bean available to the application using Java Naming and Directory Interface (JNDI) lookup mechanisms.

```
// Obtain the default initial JNDI context
Context initialContext = new InitialContext();

  // Lookup the remote home interface of the BusinessProcess bean
  Object result =
         initialContext.lookup("java:comp/env/ejb/BusinessProcessHome");

// Convert the lookup result to the proper type
   BusinessProcessHome processHome =
           (BusinessProcessHome)javax.rmi.PortableRemoteObject.narrow
           (result,BusinessProcessHome.class);
```

The home interface of the BusinessProcess session bean contains a create() method for EJB objects. The method returns the remote interface of the session bean.

4. Access the remote interface of the BusinessProcess session bean.

   ```
   BusinessProcess process = processHome.create();
   ```

5. Call the business functions exposed by the BusinessProcessService interface, for example:

   ```
   process.initiate("MyProcessModel",input);
   ```

   Calls from applications are run as transactions. A transaction is established and ended in one of the following ways:
   - Automatically by WebSphere Application Server (the deployment descriptor specifies TX_REQUIRED).
   - Explicitly by the application. You can bundle application calls into one transaction:

     ```
     // Obtain user transaction interface
        UserTransaction transaction=
            (UserTransaction)initialContext.lookup("jta/usertransaction");

        // Begin a transaction
        transaction.begin();

          // Applications calls ...

        // On successful return, commit the transaction
        transaction.commit();
     ```

Accessing the process choreographer local EJB interface

An application accesses the LocalBusinessProcess session bean through the home and local interfaces of the bean.

1. Add a reference to the LocalBusinessProcess session bean in the application deployment descriptor. Add the reference to:
   - The `application-client.xml` file, for a Java 2 Platform Enterprise Edition (J2EE) client application
   - The `web.xml` file, for a Web application
   - The `ejb-jar.xml` file, for an Enterprise JavaBeans (EJB) application

   Add the reference to the local home interface as in the following example:

   ```
   <ejb-local-ref>
    <ejb-ref-name>ejb/LocalBusinessProcessHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local-home>com.ibm.bpe.api.LocalBusinessProcessHome</local-home>
    <local>com.ibm.bpe.api.LocalBusinessProcess</local>
   </ejb-local-ref>
   ```

2. Make the local home interface of the LocalBusinessProcess session bean available to the application, using Java Naming and Directory Interface (JNDI) lookup mechanisms.

```
// Obtain the default initial JNDI context
Context initialContext = new InitialContext();

  // Lookup the local home interface of the LocalBusinessProcess bean

   LocalBusinessProcessHome processHome =
        (LocalBusinessProcessHome)initialContext.lookup
        ("java:comp/env/ejb/LocalBusinessProcessHome");
```

   The home interface of the LocalBusinessProcess session bean contains a create() method for EJB objects. The method returns the local interface of the session bean.

3. Access the local interface of the LocalBusinessProcess session bean.

```
LocalBusinessProcess process = processHome.create();
```

4. Call the business functions exposed by the BusinessProcessService interface, for example:

```
process.initiate("MyProcessModel",input);
```

   Calls from applications are run as transactions. A transaction is established and ended in one of the following ways:
   - Automatically by WebSphere Application Server (the deployment descriptor specifies TX_REQUIRED).
   - Explicitly by the application. You can bundle application calls into one transaction:

```
// Obtain user transaction interface
   UserTransaction transaction=
       (UserTransaction)initialContext.lookup("jta/usertransaction");

   // Begin a transaction
   transaction.begin();

     // Applications calls ...

   // On successful return, commit the transaction
   transaction.commit();
```

Accessing the process choreographer JMS interface

Process choreographer accepts Java Message Service (JMS) messages that follow the point-to-point paradigm. An application that sends or receives JMS messages must:

1. Create a connection to process choreographer. Use Java Naming and Directory Interface (JNDI) lookup to retrieve the connection factory. The JNDI-lookup name must be the same as the name specified when the process choreographer external request queue is configured.

```
//Obtain the default initial JNDI context
Context initialContext = new InitialContext();

// Look up the connection factory
QueueConnectionFactory queueConnectionFactory =
     (QueueConnectionFactory) initialContext.lookup("jms/BPECF");

// Create connection
QueueConnection queueConnection =
     queueConnectionFactory.createQueueConnection();
```

2. Create a session so that message producers and consumers can be created.

```
//Create a nontransacted session using autoacknowledgement
QueueSession queueSession =
     queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
```

3. Create a message producer to send messages. The JNDI-lookup name must be the same as the name specified when the process choreographer external request queue is configured.

```
// Look up the destination of the process choreographer queue
// to send messages to
Queue bpeQueue = (Queue) initialContext.lookup("jms/BPEApiQueue");
// Create message producer
QueueSender queueSender = queueSession.createSender(bpeQueue);
```

4. Create a message consumer to receive replies. The JNDI-lookup name must specify a user-defined destination to which replies are sent.

```
// Look up the destination of the reply to queue
Queue replyToQueue = (Queue) initialContext.lookup("MyReplyToQueue");
// Create message consumer
QueueReceiver queueReceiver = queueSession.createReceiver(replyToQueue);
```

5. Send requests and receive replies.

```
// Start sending and receiving messages
queueConnection.start();

// Create message - see the task descriptions for examples - and send
ObjectMessage message = queueSession.createObjectMessage();
// message.SetStringProperty(..);
// message.setObject(...);

queueSender.send(message);

// Receive message and analyze reply
// See the detailed task descriptions for examples
Message reply = queueReceiver.receive();
```

6. Close the connection to the free resources.

```
// Final housekeeping: free resources
queueConnection.close();
```

Developing applications for BPEL-based non-interruptible processes

You can develop the following applications for non-interruptible processes based on the Business Process Execution Language (BPEL):

- Run a non-interruptible process using the Enterprise JavaBeans (EJB) interface. The application depends on whether the process contains a unique starting service or a non-unique starting service.
- Run a non-interruptible process using the Java Message Service (JMS) interface

Also see the Javadoc.

Running a non-interruptible process that contains a unique starting service

You can use the Enterprise JavaBeans (EJB) interface to develop applications that run non-interruptible processes based on the Business Process Execution Language (BPEL). A non-interruptible process can be started by a receive activity or a pick activity. If the non-interruptible process implements a request-response operation, that is, the process contains a reply, you can use the call() method to execute the non-interruptible process.

If the starting service is unique, you can use the call() method and pass the process template name as a parameter. The starting service is unique if the non-interruptible process starts with a receive activity or when the pick activity has only one onMessage definition.

1. Optional: List the process templates to find the name of the non-interruptible process you want to run. This step is optional if you already know the name of the process.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.CAN_RUN_SYNC=TRUE",
    "PROCESS_TEMPLATE_NAME",
    newInteger(50),
    null);
   ```

   The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started by the call() method.

2. Start the process with an input message of the appropriate type. When you create the message, you must specify its message type name so that the message definition is contained.

   ```
   ProcessTemplateData template = processTemplates[0];
   //create a message for the single starting receive activity
   ClientObjectWrapper input = process.createMessage
                              (template.getID(),
                               template.getInputMessageTypeName());
   WSIFMessage myMessage = input.getObject();

   //set the parts in the message, for example, a customer name
   myMessage.setObjectPart("CustomerName", "Smith");

   //run the process
   ClientObjectWrapper output = process.call(template.getName(), input);
   WSIFMessage myOutput = (WSIFMessage)output.getObject();
   int order = myOutput.getIntPart("OrderNo");
   ```

   This action creates an instance of the process template, CustomerTemplate, and passes some customer data. The operation returns only when the process is complete. The result of the process, OrderNo, is returned to the caller.

Running a non-interruptible process that contains a non-unique starting service

You can use the Enterprise JavaBeans (EJB) interface to develop applications that run non-interruptible processes based on the Business Process Execution Language (BPEL). A non-interruptible can be started by a receive activity or a pick activity. If the non-interruptible process implements a request-response operation, that is, the process contains a reply, you can use the call() method to execute the non-interruptible process.

If the starting service is not unique, that is, the non-interruptible process starts with a pick activity that has multiple onMessage definitions, then you must identify the service to be called.

1. Optional: List the process templates to find the name of the non-interruptible process you want to run. This step is optional if you already know the name of the process.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_MICROFLOW",
    "PROCESS_TEMPLATE_NAME",
    newInteger(50),
    null);
   ```

The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as non-interruptible processes.

2. Determine the starting service to be called.

```
ProcessTemplateData template = processTemplates[0];
ActivityServiceTemplateData startActivities =
        process.getStartActivities(template.getID());
```

3. Start the process with an input message of the appropriate type. When you create the message, you must specify its message type name so that the message definition is contained.

```
ActivityServiceTemplateData activity = startActivities[0];
//create a message for the service to be called
ClientObjectWrapper input =
      process.createMessage(activity.getServiceTemplateID(),
                            activity.getActivityTemplateID(),
                            activity.getInputMessageTypeName());
WSIFMessage myMessage = input.getObject();

//set the parts in the message, for example, a customer name
myMessage.setObjectPart("CustomerName", "Smith");

//run the process
ClientObjectWrapper output = process.call(activity.getServiceTemplateID(),
                                          activity.getActivityTemplateID(),
                                          input);
//check the output of the process, for example, an order number
WSIFMessage myOutput = (WSIFMessage)output.getObject();
int order = myOutput.getIntPart("OrderNo");
```

This action creates an instance of the process template, CustomerTemplate, and passes some customer data. The operation returns only when the process is complete. The result of the process, OrderNo, is returned to the caller.

Executing a non-interruptible process using the JMS interface

You can use the Java Message Service (JMS) interface to run non-interruptible processes.

1. Create a message, for example, an ObjectMessage message.

```
ObjectMessage message = queueSession.createObjectMessage();
```

2. Optional: Set the JMSReplyToQueue. If you do not want to receive a reply, this step is optional.

```
//Specify the destination object replies are to be sent to
message.SetJMSReplyTo(replyToQueue);
```

3. Optional: Specify the JMS properties. If you do not specify any properties, the process template name Dispatch is assumed. The JMSReplyToQueue value must be set.

```
message.SetStringProperty("wf$verb", "call");
message.SetStringProperty("wf$processTemplateName", "CustomerTemplate");
```

4. Start the process with an input message of the appropriate type. Specify the input message as the payload of the message. When you have an array containing basic types, it is recommended that you use the Enterprise JavaBeans (EJB) interface to specify the message. This technique guarantees that the message definition is contained in the input message and that process choreographer can analyze the message.

```
//Create Customer input message
WSIFDefaultMessage input = new WSIFDefaultMessage();
input.setObjecPart("CustomerName", "Smith");
```

```
message.setObject(new ClientObjectWrapper(input));

//Send message
queueSender.send(message);
```

This action creates an instance of the process template, CustomerTemplate, and passes some customer data. The operation returns only when the flow is complete and when a JMSReplyToQueue is specified. The result of the process, OrderNo, is returned as the payload of the reply message. Because an ObjectMessage message was passed, an object message is returned.

5. Optional: Get the result of the process. You can get the results of the process only if you specified a queue in step 2. In the example, the result of the process, OrderNo, is returned.

```
Message m = queueReceiver.receive();
if (m instanceof ObjectMessage)
{
 ClientObjectWrapper wrapper = (ClientObjectWrapper)m.getObject();
 WSIFMessage output = (WSIFMessage)wrapper.getObject();
  int order = output.getIntPart("OrderNo");
}
```

Developing applications for BPEL-based interruptible processes

You can use the Enterprise JavaBeans (EJB) interface to develop the following services for interruptible processes based on the Business Process Execution Language (BPEL):

- Start an interruptible process that contains a unique starting service
- Start an interruptible process that contains a non-unique starting service
- Process a staff activity
- Send a message to a waiting activity
- Analyze the result of a process
- Use work lists to query information

For more information, see the Javadoc.

Starting an interruptible process that contains a unique starting service

You can use the Enterprise JavaBeans (EJB) interface to develop applications that run interruptible processes based on the Business Process Execution Language (BPEL). An interruptible process can be started through multiple initiating receive or pick activities. If the starting service is unique, you can use the initiate() method and pass the process template name as a parameter. This is the case when the long-running process starts with either a single receive or pick activity and when the single pick activity has only one onMessage definition.

1. Optional: List the process templates to find the name of the interruptible process you want to start. This step is optional if you already know the name of the process.

```
ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.CAN_RUN_INTERRUP=TRUE"
 "PROCESS_TEMPLATE.NAME",
 newInteger(50),
 null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started by the initiate() method.

2. Start the process with an input message of the appropriate type. When you create the message, you must specify its message type name so that the message definition is contained. If you specify a process-instance name, it must not start with an underscore. If a process-instance name is not specified, the process instance ID (PIID) in String format is used as the name.

```
ProcessTemplateData template = processTemplates[0];
//create a message for the single starting receive activity
ClientObjectWrapper input = process.createMessage
                           (template.getID(),
                            template.getInputMessageTypeName());
WSIFMessage myMessage = input.getObject();

//set the message parts, for example, the customer name
myMessage.setObjectPart("CustomerName", "Smith");

//start the process
PIID piid = process.initiate(template.getName(), "CustomerOrder", input);
```

   This action creates an instance, CustomerOrder, and passes some customer data. When the process starts, the operation returns the object ID of the new process instance to the caller.

   The starter of the process instance is set to the caller of the request and receives a work item for the process instance. The process administrators, readers, and editors of the process instance are determined and receive work items for the process instance. The follow-on activity instances are determined. These are started automatically or, if they are staff, receive, or pick activities, work items are created for the potential owners.

Starting an interruptible process that contains a non-unique starting service

You can use the Enterprise JavaBeans (EJB) interface to develop applications that run interruptible processes based on the Business Process Execution Language (BPEL). An interruptible process can be started through multiple initiating receive or pick activities. You can use the initiate() method to start the process. If the starting service is not unique, for example, the process starts with multiple receive activities or a pick activity that has multiple onMessage definitions, then you must identify the service to be called.

1. Optional: List the process templates to find the name of the interruptible process you want to start. This step is optional if you already know the name of the process.

```
ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_LONG_RUNNING"
 "PROCESS_TEMPLATE.NAME",
 newInteger(50),
 null);
```

   The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as interruptible processes.

2. Determine the starting service to be called.

```
ProcessTemplateData template = processTemplates[0];
ActivityServiceTemplateData startActivities =
        process.getStartActivities(template.getID());
```

3. Start the process with an input message of the appropriate type. When you create the message, you must specify its message type name so that the message definition is contained. If you specify a process-instance name, it must

not start with an underscore. If a process-instance name is not specified, the process instance ID (PIID) in String format is used as the name.

```
ActivityServiceTemplateData activity = startActivities[0];
//create a message for the service to be called
ClientObjectWrapper input = process.createMessage
                           (activity.getServiceTemplateID(),
                            activity.getActivityTemplateID(),
                            activity.getInputMessageTypeName());
WSIFMessage myMessage = input.getObject();

//set the message parts, for example, the customer name
myMessage.setObjectPart("CustomerName", "Smith");

//start the process
PIID piid = process.initiate(activity.getServiceTemplateID(),
                            activity.getActivityTemplateID(),
                            "CustomerOrder",
                            input);
```

This action creates an instance, CustomerOrder, and passes some customer data. When the process starts, the operation returns the object ID of the new process instance to the caller.

The starter of the process instance is set to the caller of the request and receives a work item for the process instance. The process administrators, readers, and editors of the process instance are determined and receive work items for the process instance. The follow-on activity instances are determined. These are started automatically or, if they are staff, receive, or pick activities, work items are created for the potential owners.

Processing staff activities in BPEL-based processes

Staff activities are assigned to various people in your organization through work items. When a process is started, work items are created for the potential owners. One of these owners claims the activity. This person is responsible for providing the relevant information and completing the activity.

1. List the activities belonging to a logged-on person that are ready to be worked on:

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
                    ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
                    WORK_ITEM.REASON =
                         WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                    null, null, null);
```

   This action returns a query result set that contains the activities that can be worked on by the logged-on person.

2. Claim the activity to be worked on:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 WSIFMessage activityInput = (WSIFMessage) input.getObject();
  // read the values from WSIFMessage
    ...
}
```

   When the activity is claimed, the input message of the activity is returned.

3. When work on the activity is finished, complete the activity. The activity can be completed either successfully or with a fault message. If the activity is successful, an output message is passed. If the activity is unsuccessful, the activity is put into the failed or stopped state and a fault message is passed. You must create the appropriate messages to do this. When you create the message, you must specify the message type name so that the message definition is contained.
   a. To complete the activity successfully, create an output message.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
WSIFMessage myMessage = output.getObject();

//set the parts in your message, for example, an order number
myMessage.setIntPart("OrderNo", 4711);

//complete the activity
process.complete(aiid, output);
```

      This sets an output message containing the order number.
   b. To complete the activity when a fault occurs, create a fault message.

```
//retrieve the faults modeled for the staff activity
List faultNames = process.getFaultNames(aiid);

//create a message of the appropriate type
ClientObjectWrapper myFault =
      process.createMessage(aiid, faultNames.get(0));

// set the parts in your fault message, for example, an error number
WSIFMessage myMessage = (WSIFMessage)myFault.getObject();
myMessage.setIntPart("error",1304);

process.complete(aiid,myFault,faultNames.get(0));
```

      This sets the activity in either the failed or the stopped state. If the **continue-on-error** parameter for the activity in the process model is set to true, the activity is put into the failed state and the navigation continues. If the **continue-on-error** parameter is set to false, the activity is put into the stopped state. In this state the activity can be repaired using force terminate or force retry.

Sending a message to a waiting activity

Events are used to synchronize a running process with the ″outside world″. For example, the receipt of an e-mail from a customer in response to a request for information can be viewed as an event occurrence. Activities, such as pick activities and receive activities allow the process to wait for outside events.

1. List the activities that are waiting for a message from the logged-on user.

```
QueryResultSet result =
     query("ACTIVITY_SERVICE.VTID,ACTIVITY.ATID",
           "ACTIVITY.STATE=ACTIVITY.STATE.STATE_WAITING AND
            WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
            null, null, null);
```

2. Send a message. The caller must be a potential owner of the awaited event or an administrator of the process instance.

```
if ( result.size > 0 )
   {
     result.first();
```

```
        VTID vtid = (VTID)result.getOID(1);
        ATID atid = (ATID)result.getOID(2);
        ActivityServiceTemplateData activity =
                process.getActivityServiceTemplate(vtid,atid);

        // create a message for the service to be called
        ClientObjectWrapper message =
              process.createMessage(vtid,atid,activity.getInputMessageTypeName());
        WSIFMessage myMessage = message.getObject();

        // set the parts in your message - for example, chocolate is to be ordered
        myMessage.setObjectPart("Order", "chocolate");

        // send the message to the waiting activity
        process.sendMessage(vtid, atid, message);
      }
```

This sends the specified message to the waiting activity service and passes some order data.

You can also specify the process instance ID to ensure that the message is sent to the specified process instance. If the process instance ID is not specified, the message is sent to the activity service and process instance identified by the correlation values in the message. If the process instance ID is specified, the process instance found using the correlation values is checked to ensure that it has the specified process instance ID.

Analyzing results of a process

An interruptible process runs asynchronously. Its output message is not automatically returned when the process completes. The message must be retrieved explicitly. The results of the process are stored in the database only if the process template from which the process instance was derived does not specify automatic deletion of the derived process instances.
Analyze the results of the process, for example, check the order number.

```
QueryResultSet result =
     process.query("PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                    PROCESS_INSTANCE.STATE = PROCESS_INSTANCE.STATE.STATE_FINISHED",
                    null, null, null);
if (result.size() > 0)
{
 result.first();
 PIID piid = (PIID) result.getOID(1);
 ClientObjectWrapper output = process.getOutputMessage(piid);
 WSIFMessage message = (WSIFMessage)output.getObject();
  int orderNo = message.getIntPart("OrderNo");
}
```

Using worklists to query information

A worklist is a query that is persistently stored in the database. This worklist represents a set of items that have the same characteristics. Although worklist definitions are stored persistently, items contained in the worklist are assembled dynamically when they are queried. All worklists are publicly accessible.

1. Optional: List the available worklists.

   ```
   String[] worklists = process.getWorklistNames();
   ```

2. Optional: Check the query defined by a specific worklist.

```
WorkListData worklist = process.getWorklist("CustomerOrdersStartingWithA");
String selectClause = worklist.getSelectClause();
String whereClause = worklist.getWhereClause();
String orderByClause = worklist.getOrderByClause();
Integer threshold = worklist.getThreshold();
```

3. Run the query defined by the worklist.

```
QueryResultSet result = process.executeWorklist("CustomerOrdersStartingWithA");
```

Developing administration applications for BPEL-based interruptible processes

You can develop the following administration applications for interruptible processes based on the Business Process Execution Language (BPEL). You can use the Enterprise JavaBeans (EJB) interface to render all of these applications. The Java Message Service (JMS) interface can be used only to terminate a process instance.

- Cancel a claimed activity
- Force the completion of an activity
- Retry the execution of a stopped activity
- Delete a process instance
- Terminate a process instance using the EJB interface
- Terminate a process instance using the JMS interface
- Manage worklists
- Manage work items

For more information, see the Javadoc.

Canceling a claimed activity

Sometimes it is necessary for someone with process administrator rights to cancel an activity that is claimed by someone else. This situation might occur, for example, when an activity must be completed but the owner of the activity is absent.

1. List the claimed activities owned by a specific person to find the ID of the activity in question.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_CLAIMED AND
                    ACTIVITY.OWNER = 'Smith'
                    AND ACTIVITY.TEMPLATE_NAME = 'CustomerTemplate'",
                    null, null, null);
```

   This action returns a query result set that lists the activities claimed by the specified person, Smith.

2. Cancel the claimed activity.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 process.cancelClaim(aiid);
}
```

   This action returns the activity to the ready state so that it can be claimed by one of the other potential owners.

Forcing the completion of an activity

If an activity in an interruptible process encounters an exception or an uncaught fault in the enclosing scope and the associated activity template specifies that the activity stops when an error occurs, the activity is put into the stopped state so that it can be repaired. In this state, you can force completion of the activity. For staff activities, you can also pass parameters in the force-complete call, such as the message that should have been sent or the fault that should have been raised. For invoke and script activities, you cannot pass parameters in the force-complete call. However, you must set the variables that need to be repaired.

1. List the stopped activities.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                    null, null, null);
```

   This action returns the stopped activities for the CustomerOrder process instance.

2. Complete the activity. In this example, an output message is passed:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper output =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  WSIFMessage myMessage = output.getObject();

  //set the parts in your message, for example, an order number
  myMessage.setIntPart("OrderNo", 4711);
  process.forceComplete(aiid, output, true);
}
```

   For more information, see the Javadoc. This completes the activity. If an error occurs, the **continueOnError** parameter determines whether the activity stays in the stopped state. In the example, `continueOnError=true`. This value means that if an error occurs during processing of the forceComplete request, the activity is put into the failed execution state. The fault is propagated to the enclosing scopes of the activity until it is either handled or the process scope is reached. The process is then put into the failing state and eventually reaches the failed state.

Retrying the execution of a stopped activity

If an activity in an interruptible process encounters an exception or an uncaught fault in the enclosing scope and if the associated activity template specifies that the activity stops when an error occurs, the activity is put into the stopped state so that it can be repaired. You can retry the execution of the activity.

You can set variables that are used by the activity. You can also pass parameters in the force-retry call, such as the message that was expected by the activity.

1. List the stopped activities.

```
QueryResultSet result =
      process.query("DISTINCT ACTIVITY.AIID",
                    "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                     PROCESS_INSTANCE.NAME='CustomerOrder'",
                     null, null, null);
```

This action returns the stopped activities for the CustomerOrder process instance.

2. Retry the execution of the activity.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper input =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  WSIFMessage myMessage = input.getObject();

  //set the parts in your message, for example, chocolate is to be ordered
  myMessage.setObjectPart("OrderNo", "chocolate");
  process.forceRetry(aiid, input, true);
}
```

For more information, see the Javadoc. This action retries the activity. If an error occurs, the **continueOnError** parameter determines whether the activity stays in the stopped state. In the example, **continueOnError** is true. This means that if an error occurs during processing of the forceRetry request, the activity is put into the failed state. The fault is propagated to the enclosing scopes of the activity until it is either handled or the process scope is reached. The process is then put into the failing state and eventually reaches the failed state.

Deleting a process instance

Process instances are only automatically deleted when they complete if this is specified in the process template from which the process instances are derived. To delete all finished process instances:

1. List the process instances that are finished.

```
QueryResultSet result =
     process.query("DISTINCT PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                           PROCESS_INSTANCE.STATE.STATE_FINISHED",
                    null, null, null);
```

This action returns a query result set that lists finished process instances.

2. Delete the finished process instances.

```
while (result.next() )
{
 PIID piid = (PIID) result.getOID(1);
 process.delete(piid);
}
```

Terminating a process instance using the EJB interface

Sometimes it is necessary for someone with process administrator rights to terminate a process instance that is known to be in an unrecoverable state. For example, when an application is invoked and fails and does not return to a dormant state.

It is recommended that you terminate a process instance only in exceptional situations. The process instance is terminated immediately without waiting for any outstanding subprocesses or activities. If the process template from which the process instances are derived does not specify compensation, process instances that are terminated are not compensated.

1. Retrieve the process instance to be terminated.

```
ProcessInstanceData processInstance =
       process.getProcessInstance("CustomerOrder");
```

2. Terminate the process instance. If you terminate a process instance using the Enterprise JavaBeans (EJB) interface, you can terminate the process instance with or without compensation.

   To terminate the process instance with compensation:

   ```
   PIID piid = processInstance.getID();
   process.forceTerminate(piid, CompensationBehavior.INVOKE_COMPENSATION);
   ```

   To terminate the process instance without compensation:

   ```
   PIID piid = processInstance.getID();
   process.forceTerminate(piid);
   ```

   If compensation is defined for the process instance, the process instance is terminated and compensation is started. If the process instance does not have compensation defined, the process instance is terminated immediately without waiting for any outstanding activities.

Terminating a process instance using the JMS interface

Sometimes it is necessary for someone with process administrator rights to terminate a process instance that is known to be in an unrecoverable state. For example, when an application is invoked and fails and does not return to a dormant state.

It is recommended that you terminate a process instance only in exceptional situations. The process instance is terminated immediately without waiting for any outstanding subprocesses or activities. Process instances that are terminated using the Java Message Service (JMS) interface are not compensated.

1. Create a message, for example, an ObjectMessage message.

   ```
   ObjectMessage message= queueSession.createObjectMessage();
   ```

2. Optional: Set the JMSReplyToQueue. If you do not want to receive a reply, this step is optional.

   ```
   // Specify the destination object replies are to be sent to
   message.SetJMSReplyTo(replyToQueue);
   ```

3. Set the JMS properties.

   ```
   message.SetStringProperty("wf$verb", "forceTerminate");
   message.SetStringProperty("wf$processInstanceName", "CustomerOrder");
   ```

4. Terminate the CustomerOrder process instance.

   ```
   // Send message
   queueSender.send(message);
   ```

   The process instance is terminated immediately without waiting for any outstanding subprocesses or activities. If the JMSReplyToQueue value is set, this action returns an empty reply message if the process instance was terminated successfully. The JMSCorrelationID value is set to the JMSMessageID value of the forceTerminate request. Neither properties nor payload are set on the reply message.

Managing worklists

A worklist is a query that is stored persistently in the database. This worklist represents a set of items that have the same characteristics. Although worklist definitions are stored persistently, items contained in the worklist are assembled dynamically when they are queried. All worklists are publicly accessible.

1. Create a worklist. Create a query and save it with a specific name.

```
process.newWorklist("CustomerOrdersStartingWithA",
             "PROCESS_INSTANCE.NAME, DISTINCT PROCESS_INSTANCE.PIID",
             "PROCESS_INSTANCE.NAME LIKE 'A%'",
             "PROCESS_INSTANCE.NAME",
             null,
             null);
```

   This query returns a sorted list of all the process-instance names that begin with the letter A and their associated process instance IDs (PIID).

2. Delete a worklist.

```
process.deleteWorklist("CustomerOrdersStartingWithA");
```

Managing work items

A work item represents a relationship between a person, or group of people, and an object, typically an activity instance. The relationship is described by attributes such as the type of the associated object and the reason why the object is assigned.

Work items are created whenever a staff activity, an activity waiting for an event, or an erroneous activity is encountered during the navigation of a process instance. The associated staff resolution plug-in is invoked and returns a list of people. Each person on the list receives a work item for the activity instance. Work items are also created for the starter, process administrators, editors, and readers of a process instance.

During the lifetime of a process instance, the set of associated people can change, for example, because a person is ill or leaves the enterprise, new people are hired, or the workload needs to be distributed differently. To allow for these changes, you can develop applications to create, delete, or transfer work items.

- Create a work item.

```
// query the process instance for which an additional
// process administrator is to be specified
QueryResultSet result = process.query("PROCESS_INSTANCE.PIID",
                                      "PROCESS_INSTANCE.NAME='CustomerOrder'",
                                       null, null, null);
if ( result.size() > 0 )
{
  result.first();
  // create the work item
  process.createWorkItem((PIID)(result.getOID(1)),
                         WorkItemData.REASON_ADMINISTRATOR,"Smith");
}
```

  This creates a process instance administrator work item for the user Smith.

- Delete a work item.

```
// query the process instance for which a work item is to be deleted
QueryResultSet result = process.query("PROCESS_INSTANCE.PIID",
                                      "PROCESS_INSTANCE.NAME='CustomerOrder'",
                                       null, null, null);
if ( result.size() > 0 )
{
  result.first();
  // delete the work item
  process.deleteWorkItem((PIID)(result.getOID(1)),
                         WorkItemData.REASON_READER,"Smith");
}
```

  This deletes the process instance reader work item for the user Smith.

- Transfer a work item.

```
// query the activity that is to be rescheduled
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "PROCESS_INSTANCE.NAME='CustomerOrder' AND
                    ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY AND
                    WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER AND
                    WORK_ITEM.OWNER_ID='Miller'",
                    null, null, null);
if ( result.size() > 0 )
{
  result.first();
  // transfer the work item from user Miller to user Smith
  // so that Smith can work on the activity
  process.transferWorkItem((AIID)(result.getOID(1)),
                           WorkItemData.REASON_POTENTIAL_OWNER,"Miller","Smith");
}
```

  This transfers the work item to the user Smith so that he can work on it.

Handling exceptions and faults

Faults can occur when a process instance is created or when operations that are invoked as part of the navigation of a process instance fail. Mechanisms exist to handle these faults and they include:

- Passing control to the corresponding fault handlers
- Stopping the process and let someone repair the situation (force-retry, force-complete)
- Compensating the process
- Passing the fault to the client application as an API exception, for example, an exception is thrown when the process model from which an instance is to be created does not exist

The handling of faults and exceptions is described in the following tasks:

- Handling API exceptions
- Checking which fault is set for a staff activity using the EJB interface
- Checking which fault occurred for a stopped invoke activity using the EJB interface

Handling API exceptions

For the Enterprise JavaBeans (EJB) interface, if a method in the BusinessProcessService interface does not complete successfully, an exception is thrown that denotes the cause of the error. You can handle this exception specifically to provide guidance to the caller.

However, it is common practice to handle only a subset of the exceptions specifically and to provide general guidance for the other potential exceptions. All specific exceptions inherit from a generic ProcessException. It is a *best practice* to catch generic exceptions with a final `catch(ProcessException)` statement. This statement helps to ensure the upward compatibility of your application program because it takes account of all other exceptions that can occur.

For the Java Message Service (JMS) interface, the description of the message payload in a JMS message contains information on the error that occurred.

Checking which fault is set for a staff activity using the EJB interface

1. List the staff activities that are in a failed or stopped state.

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "(ACTIVITY.STATE = ACTIVITY.STATE.STATE_FAILED OR
                     ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED) AND
                     ACTIVITY.KIND=ACTIVITY.KIND.KIND_STAFF",
                     null, null, null);
```

   This action returns a query result set that contains failed or stopped staff activities.

2. Read the name of the fault. This is the local part of the fault QName.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper faultMessage = process.getFaultMessage(aiid);
 WSIFMessage fault = (WSIFMessage) faultMessage.getObject();
  String faultName = fault.getName();
 }
```

   This returns the fault name. You can also analyze the unhandled exception for a stopped activity instead of retrieving the fault name.

Checking which fault occurred for a stopped invoke activity using the EJB interface

1. List the staff activities that are in a stopped state.

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    ACTIVITY.KIND=ACTIVITY.KIND.KIND_INVOKE",
                    null, null, null);
```

   This action returns a query result set that contains stopped invoke activities.

2. Read the name of the fault. This is the local part of the fault QName.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);

  ProcessException excp = activity.getUnhandledException();
  if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException fault = (ApplicationFaultException)excp;
   String faultName = fault.getFaultName();
  }
}
```

   This returns the fault name. You can also analyze the unhandled exception for a stopped activity instead of retrieving the fault name.

Authorization for EJB renderings for BPEL-based processes

It is recommended that you enable security in WebSphere Application Server.

When an instance of the LocalBusinessProcess or the BusinessProcess session bean is created, WebSphere Application Server associates a session context with the instance. The session context contains the caller's principal. This information is used by both the business process container and the process engine to check the caller's authorization for each call.

The following reasons for a work-item assignment are used:
- For processes: reader, starter, administrator
- For activities: reader, editor, potential owner, owner

These assignment reasons are mapped to authorization authorities:
- Activity reader authority: can see properties of the associated activity instance, and its input and output messages.
- Activity editor authority: has the authority of the activity reader, and has write access to messages and other data associated with the activity.
- Potential activity owner authority: has the authority of the activity editor, and has the right to claim the activity or to send a message to the activity.
- Activity owner authority: has the authority of the potential activity owner, and has the right to complete the activity. Has the authority to transfer owned work items to an administrator or potential owner.
- Process starter authority: can see properties of the associated process instance, its input and output messages, and write other data associated with the process.
- Process reader authority: can see properties of the associated process instance, its input and output messages, and everything that the activity reader supports for all contained activities but not those of the independent subprocesses.
- Process administrator authority: has the authority of the process reader and the process starter, and the right to intervene in a process that has started. Has the authority to create, delete, transfer work items.

Special authority is granted to a person with the role of business process administrator. This role is different from the process administrator of a process instance; a business process administrator has all privileges.

You cannot delete the user ID of the process starter from your user registry while the process instance exists. If you do, the navigation of this process cannot continue. You receive the following exception in the system log file:

```
no unique ID for: <user ID>
```

Required authorizations for BPEL-based process requests

Access to the LocalBusinessProcess or the BusinessProcess interface does not guarantee that the caller can perform all actions on a process; the caller must also be authorized to perform the request. The following minimum authorization authorities are needed for BPEL-based process requests:

Request	Required authorization
createMessage	process reader
createWorkItem	process administrator
getAllActivities	process reader
getAllWorkItems	process reader
geClientUISettings	process reader
getCustomProperty	process reader
getFaultMessage	process reader

Request	Required authorization
getInputClientUISettings	process reader
getInputMessage	process reader
getOutputClientUISettings	process reader
getOutputMessage	process reader
getProcessInstance	process reader
getVariable	process reader
getWorkItems	process reader
setCustomProperty	process starter
setVariable	process administrator
delete	process administrator
deleteWorkItem	process administrator
transferWorkItem	process administrator
forceTerminate	process administrator
forceTerminateAndCompensate	process administrator

Required authorizations for BPEL-based activity requests

Access to the LocalBusinessProcess or the BusinessProcess interface does not guarantee that the caller can perform all actions on an activity; the caller must also be authorized to perform the request. The following minimum authorization authorities are needed for BPEL-based activity requests:

Request	Required authorization
createMessage	activity reader or process reader
createWorkItem	process administrator
getActivityInstance	activity reader or process reader
getCustomProperty	activity reader or process reader
getFaultMessage	activity reader or process reader
getFaultNames	activity reader or process reader
getInputMessage	activity reader or process reader
getOutputMessage	activity reader or process reader
getClientUISettings	activity reader or process reader
getWorkItems	activity reader or process administrator
getAllWorkItems	process reader or process administrator
setCustomProperty	activity editor or process administrator
setOutputMessage	activity editor or process administrator
setFaultMessage	activity editor or process administrator
call	potential activity owner or process administrator
initiate	potential activity owner or process administrator
claim	potential activity owner or process administrator
cancelClaim	activity owner or process administrator
complete	activity owner or process administrator
forceRetry	process administrator

Request	Required authorization
forceComplete	process administrator
deleteWorkItem	process administrator
transferWorkItem	activity owner or process administrator
sendMessage	potential activity owner or process administrator

Authorization for JMS renderings

It is recommended that you enable security in WebSphere Application Server.

When an instance of the BusinessProcess message-driven bean (MDB) is deployed, the role MDBUser role must map to a specific user ID. This user ID is used by both the business process container and the process engine to check the caller's authorization for each request.

The following authorization authorities are needed:

Request	Required authorization
forceTerminate	process administrator

Special authority is granted to a person with the role of business process administrator. This role is different from the process administrator of a process instance; a business process administrator has all privileges.

You cannot delete the user ID of the process starter from your user registry while the process instance exists. If you do, the navigation of this process cannot continue. You receive the following exception in the system log file:

```
no unique ID for: <user ID>
```

Structure of a process choreographer JMS message

A Java Message Service (JMS) message consists of:
- A message header for message identification and routing information.
- Properties for JMS-specific data, application-specific data, and provider-specific data (optional).
- The body (payload) of the message that holds the content (optional).

Process choreographer supports text, object, and bytes-message formats.

Message header

A process choreographer client application can set the following fields:
- JMSReplyTo

  The destination to send a reply to the request. If this field is not set, a reply is not returned.
- JMSMessageID

  Uniquely identifies a message. This value is set by process choreographer when the message sent returns. This identifier is used as the JMSCorrelationID in the reply message.
- JMSCorrelationID

Links messages. Do not set this field. A process choreographer reply message contains the JMSMessageID of the request message.

Data properties

This data is passed as name-value pairs. Process choreographer adds the following properties:

- wf$verb

  The function to call. Possible values are: call, forceTerminate.
- wf$processTemplateName

  The name of the process template to instantiate. This template name must be set for the call function. If the template name is not set, Dispatch is assumed.
- wf$processInstanceName

  The name of the process instance. You can use this name with the forceTerminate function to identify the process instance to process. It can be used with the call function to specify the process instance name of the process instance to create.
- wf$processState

  The state of the process instance. This value is set in reply messages.
- wf$exceptionText

  Specifies the message text of the exception that ended the process.

Message payload

Use to specify input, output, and fault messages.

- Request messages
  - TextMessage must contain an object of the type String. This object contains the input message to pass. The defined input message must also be of type String.
  - ObjectMessage must contain an object of the type ClientObjectWrapper. This object contains the input message to pass.
  - BytesMessage must contain a byte array that represents the streamed version of the input message to pass.
- Reply messages
  - If possible, the reply message is the same format as the request message. That is, if the request message has a TextMessage format, then the reply message also has a TextMessage format. If a TextMessage format cannot be used, for example, because the type of the process instance output message is not a String, then a message with ObjectMessage format is returned.
  - If an ObjectMessage message is returned, it contains an object of the type ClientObjectWrapper. This object contains the output message or the message of the fault terminal.
  - If a BytesMessage message is returned, it contains the streamed output message or the streamed message of the fault terminal.
  - If a process can be navigated until its output or fault terminal is reached, the wf$processInstanceName and the wf$processState properties are set.
  - If an exception occurs during the processing of a request or if an exception occurs during the execution of a process instance and the fault is not connected to a fault terminal of the process, the reply message does not contain a payload. The wf$processInstanceName, the wf$processState, and the wf$exceptionText properties are set. The wf$exceptionText property contains

the message text of the exception that ended the process. Nested exceptions follow the message text. These exceptions are separated by new-line characters.

Queries on business-process objects

You can use the process query interface to retrieve business-process information that is stored persistently. You use SQL-like syntax to query the following objects:
- Process templates
- Process instances
- Activity instances
- Activity services
- Work items

The query function is provided by the interface of the business process session bean. For process templates, the query function has the following syntax:

```
ProcessTemplateData[] queryProcessTemplates (java.lang.String whereClause,
                      java.lang.String orderByClause,
                      java.lang.Integer threshold,
                      java.util.Timezone timezone);
```

For the other business-process objects, the query function has the following syntax:

```
QueryResultSet query (java.lang.String selectClause,
                      java.lang.String whereClause,
                      java.lang.String orderByClause,
                      java.lang.Integer threshold,
                      java.util.Timezone timezone);
```

The query is made up of:
- Select clause
- Where clause
- Order-by clause
- Threshold parameter
- Timezone parameter

For example, a list of work items accessible to the caller of the function is retrieved by:

```
QueryResultSet result = process.query("WORK_ITEM.WIID",
                                      null, null,
                                      null, null);
```

The query function returns objects according to the caller's authorization. The query result set contains only those objects that the caller is authorized to see.

For more information, see the Javadoc.

Predefined views for queries on BPEL-based business process objects

Process choreographer provides the following predefined views for queries on business process objects in processes based on the Business Process Execution Language (BPEL):
- PROCESS_TEMPLATE
- PROCESS_INSTANCE
- PROCESS_ATTRIBUTE
- ACTIVITY

- ACTIVITY_ATTRIBUTE
- ACTIVITY_SERVICE
- WORK_ITEM

PROCESS_TEMPLATE view

Process choreographer provides the following predefined view for queries on BPEL-based process templates:

Column name	Type	Comments
PTID	ID	The process template ID.
NAME	String	The name of the process template.
APPLICATION_NAME	String	The name of the enterprise application to which the process template belongs.
VALID_FROM	Timestamp	The time from when the process template can be instantiated.
VERSION	String	User-defined version.
CREATED	Timestamp	The time the process template is created in the database.
STATE	Integer	Specifies whether the process template is available to create process instances. Possible values: STATE_STARTED STATE_STOPPED
DESCRIPTION	String	Description of the process template.
CATEGORY	String	The category to which the process template belongs.
CAN_RUN_SYNC	Boolean	Specifies if the process can be invoked by call().
CAN_RUN_INTERRUP	Boolean	Specifies if the process can be invoked by initiate().
EXECUTION_MODE	Integer	Specifies how process instances derived from this process template can be run. Possible values are: EXECUTION_MODE_MICROFLOW EXECUTION_MODE_LONG_RUNNING
COMP_SPHERE	Integer	Specifies the compensation behavior of process instances derived from this process template; either an existing compensation sphere is joined or a compensation sphere is created. Possible values for top-level processes are: COMP_SPHERE_NOT_SUPPORTED COMP_SPHERE_REQUIRES_NEW Possible values for subprocesses are: COMP_SPHERE_REQUIRED COMP_SPHERE_SUPPORTS

PROCESS_ATTRIBUTE view

Process choreographer provides the following predefined view for queries on process attributes in BPEL-based processes:

Column name	Type	Comments
PIID	ID	The ID of the process instance that has a customer property.
NAME	String	The name of the customer property.
VALUE	String	The value of the customer property.

PROCESS_INSTANCE view

Process choreographer provides the following predefined view for queries on BPEL-based process instances:

Column name	Type	Comments
PTID	ID	The process template ID.
PIID	ID	The process instance ID.
NAME	String	The name of the process instance.
STATE	Integer	The state of the process instance. Possible values: STATE_READY STATE_RUNNING STATE_FINISHED STATE_COMPENSATING STATE_INDOUBT STATE_FAILED STATE_TERMINATED STATE_COMPENSATED STATE_TERMINATING STATE_FAILING
CREATED	Timestamp	The time the process instance is created.
STARTED	Timestamp	The time the process instance is started.
COMPLETED	Timestamp	The time the process instance is completed.
PARENT_NAME	String	The name of the parent process instance.
TOP_LEVEL_NAME	String	The name of the top-level process instance. This name is the current process instance if there is no top level.
STARTER	String	The principal ID of the starter of the process instance.
TEMPLATE_NAME	String	The name of the associated process template.
TEMPLATE_DESCR	String	Description of the associated process template.

Column name	Type	Comments
TEMPLATE_CATEGORY	String	The category to which the associated process template belongs.
DESCRIPTION	String	If the description of the process template contains placeholders, this column contains the description of the process instance with the placeholders resolved.

ACTIVITY view

Process choreographer provides the following predefined view for queries on activities in BPEL-based processes:

Column name	Type	Comments
PIID	ID	The process instance ID.
AIID	ID	The activity instance ID.
PTID	ID	The process template ID.
ATID	ID	The activity template ID.
KIND	Integer	The kind of activity. Possible values: KIND_INVOKE KIND_RECEIVE KIND_REPLY KIND_THROW KIND_TERMINATE KIND_WAIT KIND_COMPENSATE KIND_SEQUENCE KIND_EMPTY KIND_SWITCH KIND_WHILE KIND_PICK KIND_FLOW KIND_SCOPE KIND_SCRIPT KIND_STAFF KIND_ASSIGN
RUN_MODE	Integer	Possible values: Zero for BPEL-based processes.
ACTIVATED	Timestamp	The time the activity is activated.
STARTED	Timestamp	The time the activity is started.
COMPLETED	Timestamp	The time the activity is completed.

Column name	Type	Comments
STATE	String	The state of the activity. Possible values: STATE_INACTIVE STATE_READY STATE_RUNNING STATE_SKIPPED STATE_FINISHED STATE_FAILED STATE_TERMINATED STATE_CLAIMED STATE_TERMINATING STATE_FAILING STATE_WAITING STATE_EXPIRED STATE_STOPPED
OWNER	String	Principal ID of the owner.
TEMPLATE_NAME	String	Name of the associated activity template.
TEMPLATE_DESCR	String	Description of the associated activity template.
DESCRIPTION	String	If the activity template description contains placeholders, this column contains the description of the activity instance with the placeholders resolved.
BUSINESS_RELEVANCE	Boolean	Specifies whether activities are audited. Possible values are: **TRUE** The activity is business relevant and it is audited. **FALSE** Auxiliary step.

ACTIVITY_ATTRIBUTE view

Process choreographer provides the following predefined view for queries on activity attributes in BPEL-based processes:

Column name	Type	Comments
AIID	ID	The ID of the activity instance that has a customer property.
NAME	String	The name of the customer property.
VALUE	String	The value of the customer property.

ACTIVITY_SERVICE view

Process choreographer provides the following predefined view for queries on activity services in BPEL-based processes:

Column name	Type	Comments
EIID	ID	The ID of the event instance.
AIID	ID	The ID of the activity waiting for the event.

Column name	Type	Comments
PIID	ID	The ID of the process instance that contains the event.
VTID	ID	The ID of the service template that describes the event.
PORT_TYPE	String	The name of the port type.
NAME_SPACE_URI	String	The URI of the name space.
OPERATION	String	The operation name of the service.

WORK_ITEM view

Process choreographer provides the following predefined view for queries on work items derived from BPEL-based processes:

Column name	Type	Comments
WIID	ID	The work item ID.
OWNER_ID	String	The principal ID of the owner.
EVERYBODY	Boolean	Specifies whether everybody owns this work item.
OBJECT_TYPE	Integer	The type of the associated object. Possible values: OBJECT_TYPE_ACTIVITY OBJECT_TYPE_PROCESS_INSTANCE OBJECT_TYPE_EVENT
OBJECT_ID	ID	The ID of the associated object, for example, the associated process or activity.
ASSOC_OBJECT_TYPE	Integer	The type of object associated with, or containing, the associated object of the work item. Possible values: OBJECT_TYPE_ACTIVITY OBJECT_TYPE_PROCESS_INSTANCE OBJECT_TYPE_EVENT
ASSOC_OID	ID	The ID of the object associated with, or containing, the associated object of the work item. For example, the process instance ID (PIID) of the process instance containing the activity instance for which this work item was created.
REASON	Integer	The reason for the assignment of the work item. Possible values: REASON_POTENTIAL_OWNER REASON_EDITOR REASON_READER REASON_OWNER REASON_STARTER REASON_ADMINISTRATOR

Select clause

The select clause describes the query result. It specifies a list of names that identify the object properties (columns of the result) to return. Its syntax is the same as an SQL select clause; use commas to separate parts of the clause. Each part of the clause must specify a property from one of the predefined views. The columns returned in the QueryResultSet object appear in the same order as the properties specified in the select clause.

The select clause does not support SQL aggregation functions, such as AVG(), SUM(), MIN(), MAX(), or COUNT().

Example select clauses

- "WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON"

  Gets the object types of the associated objects and the assignment reasons for the work items.
- "DISTINCT WORK_ITEM.OBJECT_ID"

  Gets all object IDs of objects for which the caller has a work item without duplicates.
- "ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON"

  Gets the names of the activities the caller has work items for and their assignment reason.
- "ACTIVITY.STATE, PROCESS_INSTANCE.STARTER"

  Gets the states of the activities and the starters of their associated process instances.

If an error occurs during the processing of the select clause, a QueryUnknownTable or a QueryUnknownColumn exception is thrown with the name of the property that is not recognized as a table or column name.

Where clause

The where clause describes the filter criteria to apply to the query domain. Its syntax is the same as an SQL where-clause. If you do not want to filter a query, you must specify `null` for the where clause.

The where-clause syntax supports:

- Keywords: AND, OR, NOT
- Comparison operators: =, <=, <, <>, >,>=, LIKE
- Set operation: IN

  The LIKE operation supports the wildcard characters defined for the queried database.

The following rules also apply:

- Specify object ID constants as `ID('string-rep-of-oid')`.
- Specify time-stamp constants as `TS('yyyy-mm-ddThh:mm:ss')`. To refer to the current date, specify `CURRENT_DATE` as the timestamp.

  You must specify at least a date or a time value in the timestamp:
  - If you specify a date only, the time value is set to zero.
  - If you specify a time only, the date is set to the current date.

- If you specify a date, the year must consist of four digits; the month and day values are optional. Missing month and day values are set to 01. For example, `TS('2003')` is the same as `TS('2003-01-01T00:00:00')`.
- If you specify a time, these values are expressed in the 24-hour system. For example, if the current date is 1 January 2003, `TS('T16:04')` or `TS('16:04')` is the same as `TS('2003-01-01T16:04:00')`.

- Specify binary constants as `BIN('UTF-8 string')`.
- It is recommended that you use symbolic constants instead of integer enumerations. For example, instead of specifying an activity state expression `ACTIVITY.STATE=2`, specify `ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY`.

Examples of where clauses

- Comparing an object ID with an existing ID

  ```
  "WORK_ITEM.WIID = ID('_WI:800c00ed.df8d7e7c.fefff80.38')"
  ```

  This type of where clause is usually created dynamically with an existing object ID from a previous call. If this object ID is stored in a *wiid1* variable, the clause can be constructed as:

  ```
  "WORK_ITEM.WIID = ID('" + wiid1.toString() + "')"
  ```

- Using time stamps

  ```
  "ACTIVITY.STARTED >= TS('2002-06-1T16.00.00')"
  ```

- Using symbolic constants

  ```
  "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER"
  ```

- Using boolean values true and false

  ```
  "PROCESS_TEMPLATE.CAN_RUN_SYNC = TRUE"
  ```

Order-by clause

Use the order-by clause to specify the sort criteria for the query result set. The order-by clause syntax is the same as an SQL order-by clause; use commas to separate each part of the clause. Each part of the clause must specify a property from one of the predefined views.

If you identify more than one property, the query result set is ordered by the values of the first property, then by the values of the second property, and so on.

If you do not want to sort the query result set, you must specify `null` for the order-by clause.

Examples of order-by clauses:

- ″PROCESS_TEMPLATE.NAME″

  Sorts the query result alphabetically by the process-template name.
- ″PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC″

  Sorts the query result by the creation date and, for a specific date, sorts the results alphabetically by the process-instance name in reverse order.
- ″ACTIVITY.OWNER, ACTIVITY_TEMPLATE.NAME, ACTIVITY.STATE″

  Sorts the query result by the activity owner, then the activity-template name, and then the state of the activity.

Threshold parameter

The threshold parameter in the query function restricts the number of objects returned in the query result set. This parameter can be useful, for example, in a

graphical user interface where only a small number of items should be displayed. If you set the threshold parameter accordingly, it improves performance. The database query is faster and less data needs to transfer from the server to the client.

If the parameter is set to `null`, a threshold is not applied and all the qualifying objects are returned.

Example of a threshold parameter:

- new Integer(50)

  Specifies to return 50 qualifying objects.

Timezone parameter

Timezones can differ between the client that starts the query and the process engine that processes the query. Use the timezone parameter to specify the timezone of the time-stamp constants used in the where clause, for example, to specify local times. The dates returned in the query result set have the same timezone as that specified in the query.

If the parameter is set to `null`, the timestamp constants are assumed to be Coordinated Universal Time (UTC) times.

Examples of timezone parameters:

- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS ('2002-01-01T17:40')",
                 null,
                 null,
                 java.util.Timezone.getDefault() );
  ```

  Specifies to return object IDs for activities that started later than 17:40 local time on 1 January 2002.

- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS ('2002-01-01T17:40')",
                 null, null, null;
  ```

  Specifies to return object IDs for activities that started later than 17:40 UTC on 1 January 2002. This specification is, for example, 6 hours earlier in Eastern Standard Time.

Query results

A query result set contains the results of a query. The elements of the set are objects that the caller is authorized to see. You can read elements in a relative fashion using the next() method or in an absolute fashion using the first() and last() methods. Because the implicit cursor of a query result set is initially positioned before the first element, you must call either the first() or next() methods before reading an element. You can use the size() method to determine the number of elements in the set.

An element of the query result set comprises the selected attributes of work items and their associated referenced objects, such as activity instances and process instances. The first attribute (column) of a QueryResultSet element specifies the value of the first attribute specified in the select clause of the query request. The second attribute (column) of a QueryResultSet element specifies the value of the second attribute specified in the select clause of the query request, and so on.

You can retrieve the values of the attributes by calling a method that is compatible with the attribute type and by specifying the appropriate column index. The numbering of the column indexes starts with 1.

Attribute type	Method
String	getString
ID	getOID
Timestamp	getTimestamp getString
Integer	getInteger getShort getLong getString getBoolean
Boolean	getBoolean getShort getInteger getLong getString

Example:

The following query is run:

```
QueryResultSet resultSet = process.query("ACTIVITY.STARTED,
                                         ACTIVITY.TEMPLATE_NAME AS NAME,
                                         WORK_ITEM.WIID, WORK_ITEM.REASON",
                                         null, null, null, null);
```

The returned query result set has four columns:
- Column 1 is a time stamp
- Column 2 is a string
- Column 3 is an object ID
- Column 4 is an integer

You can use the following methods to retrieve the attribute values:

```
while (resultSet.next())
{
 java.util.Calendar activityStarted = resultSet.getTimestamp(1);
 String templateName = resultSet.getString(2);
 WIID wiid = (WIID) resultSet.getOID(3);
 Integer reason = resultSet.getInteger(4);
}
```

You can use the display names of the result set, for example, as headings for a printed table. These names are the column names of the view or the name defined by the AS clause in the query. You can use the following methods to retrieve the display names in the example:

```
resultSet.getColumnDisplayName(1) returns "STARTED"
resultSet.getColumnDisplayName(2) returns "NAME"
resultSet.getColumnDisplayName(3) returns "WIID"
resultSet.getColumnDisplayName(4) returns "REASON"
```

Chapter 10. Developing applications for V5.0-style processes

Note: Several process choreographer API interfaces and methods used for business processes created with WebSphere Studio Application Developer Integration Edition Version 5.0 or earlier are deprecated in Version 5.1. A list of these interfaces and methods can be found in the Javadoc.

The process choreographer API provides the following renderings for developing applications:

- An Enterprise JavaBeans (EJB) rendering that allows the API to be called locally or remotely. There is a stateless session bean for each type of call (LocalBusinessProcess and BusinessProcess) that exposes the functions that an application program can call. The BusinessProcessService interface provides a common interface for these session beans.
- A Java Message Service (JMS) rendering that allows a subset of the API functions to be called remotely using JMS.

You can use the API to develop the following types of applications:

- Business process applications for non-interruptible processes
- Business process applications for interruptible processes
- Administration applications for interruptible processes

For more information, see the Javadoc.

Accessing the process choreographer EJB interface

An application accesses the BusinessProcess session bean through the home and remote interfaces of the bean.

1. Add a reference to the BusinessProcess session bean in the application deployment descriptor. Add the reference to:
   - The `application-client.xml` file, for a Java 2 Platform Enterprise Edition (J2EE) client application
   - The `web.xml` file, for a Web application
   - The `ejb-jar.xml` file, for an Enterprise JavaBeans (EJB) application

   Add the reference to the remote home interface as in the following example:

   ```
   <ejb-ref>
    <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.bpe.api.BusinessProcessHome</home>
    <remote>com.ibm.bpe.api.BusinessProcess</remote>
   </ejb-ref>
   ```

2. Package the generated stubs with your application. If your application runs on a different Java Virtual Machine (JVM) from the one where the BPEContainer application runs.
   a. Package the files contained in the `WebSphere/AppServer/ProcessChoreographer/client/bpe137650.jar` file with the enterprise archive (EAR) file of your application.
   b. Set the **Classs-Path** parameter in the manifest file of the application module to include the `bpe137650.jar` file. The application module can be a J2EE application, a Web application, or an EJB application.

3. Make the home interface of the BusinessProcess session bean available to the application using Java Naming and Directory Interface (JNDI) lookup mechanisms.

```
// Obtain the default initial JNDI context
Context initialContext = new InitialContext();

  // Lookup the remote home interface of the BusinessProcess bean
  Object result =
    initialContext.lookup("java:comp/env/ejb/BusinessProcessHome");

// Convert the lookup result to the proper type
   BusinessProcessHome processHome =
           (BusinessProcessHome)javax.rmi.PortableRemoteObject.narrow
            (result,BusinessProcessHome.class);
```

   The home interface of the BusinessProcess session bean contains a create() method for EJB objects. The method returns the remote interface of the session bean.

4. Access the remote interface of the BusinessProcess session bean.

```
BusinessProcess process = processHome.create();
```

5. Call the business functions exposed by the BusinessProcessService interface, for example:

```
process.initiate("MyProcessModel",input);
```

   Calls from applications are run as transactions. A transaction is established and ended in one of the following ways:
   - Automatically by WebSphere Application Server (the deployment descriptor specifies TX_REQUIRED).
   - Explicitly by the application. You can bundle application calls into one transaction:

```
// Obtain user transaction interface
   UserTransaction transaction=
       (UserTransaction)initialContext.lookup("jta/usertransaction");

   // Begin a transaction
   transaction.begin();

     // Applications calls ...

   // On successful return, commit the transaction
   transaction.commit();
```

Accessing the process choreographer local EJB interface

An application accesses the LocalBusinessProcess session bean through the home and local interfaces of the bean.

1. Add a reference to the LocalBusinessProcess session bean in the application deployment descriptor. Add the reference to:
   - The `application-client.xml` file, for a Java 2 Platform Enterprise Edition (J2EE) client application
   - The `web.xml` file, for a Web application
   - The `ejb-jar.xml` file, for an Enterprise JavaBeans (EJB) application

   Add the reference to the local home interface as in the following example:

```
<ejb-local-ref>
 <ejb-ref-name>ejb/LocalBusinessProcessHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.bpe.api.LocalBusinessProcessHome</local-home>
 <local>com.ibm.bpe.api.LocalBusinessProcess</local>
</ejb-local-ref>
```

2. Make the local home interface of the LocalBusinessProcess session bean available to the application, using Java Naming and Directory Interface (JNDI) lookup mechanisms.

```
// Obtain the default initial JNDI context
Context initialContext = new InitialContext();

  // Lookup the local home interface of the LocalBusinessProcess bean

   LocalBusinessProcessHome processHome =
        (LocalBusinessProcessHome)initialContext.lookup
         ("java:comp/env/ejb/LocalBusinessProcessHome");
```

   The home interface of the LocalBusinessProcess session bean contains a create() method for EJB objects. The method returns the local interface of the session bean.

3. Access the local interface of the LocalBusinessProcess session bean.

```
LocalBusinessProcess process = processHome.create();
```

4. Call the business functions exposed by the BusinessProcessService interface, for example:

```
process.initiate("MyProcessModel",input);
```

   Calls from applications are run as transactions. A transaction is established and ended in one of the following ways:
   - Automatically by WebSphere Application Server (the deployment descriptor specifies TX_REQUIRED).
   - Explicitly by the application. You can bundle application calls into one transaction:

```
// Obtain user transaction interface
   UserTransaction transaction=
       (UserTransaction)initialContext.lookup("jta/usertransaction");

   // Begin a transaction
   transaction.begin();

     // Applications calls ...

   // On successful return, commit the transaction
   transaction.commit();
```

Accessing the process choreographer JMS interface

Process choreographer accepts Java Message Service (JMS) messages that follow the point-to-point paradigm. An application that sends or receives JMS messages must:

1. Create a connection to process choreographer. Use Java Naming and Directory Interface (JNDI) lookup to retrieve the connection factory. The JNDI-lookup name must be the same as the name specified when the process choreographer external request queue is configured.

```
//Obtain the default initial JNDI context
Context initialContext = new InitialContext();

// Look up the connection factory
```

```
QueueConnectionFactory queueConnectionFactory =
     (QueueConnectionFactory) initialContext.lookup("jms/BPECF");

// Create connection
QueueConnection queueConnection =
     queueConnectionFactory.createQueueConnection();
```

2. Create a session so that message producers and consumers can be created.

```
//Create a nontransacted session using autoacknowledgement
QueueSession queueSession =
     queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
```

3. Create a message producer to send messages. The JNDI-lookup name must be the same as the name specified when the process choreographer external request queue is configured.

```
// Look up the destination of the process choreographer queue
// to send messages to
Queue bpeQueue = (Queue) initialContext.lookup("jms/BPEApiQueue");
// Create message producer
QueueSender queueSender = queueSession.createSender(bpeQueue);
```

4. Create a message consumer to receive replies. The JNDI-lookup name must specify a user-defined destination to which replies are sent.

```
// Look up the destination of the reply to queue
Queue replyToQueue = (Queue) initialContext.lookup("MyReplyToQueue");
// Create message consumer
QueueReceiver queueReceiver = queueSession.createReceiver(replyToQueue);
```

5. Send requests and receive replies.

```
// Start sending and receiving messages
queueConnection.start();

// Create message - see the task descriptions for examples - and send
ObjectMessage message = queueSession.createObjectMessage();
// message.SetStringProperty(..);
// message.setObject(...);

queueSender.send(message);

// Receive message and analyze reply
// See the detailed task descriptions for examples
Message reply = queueReceiver.receive();
```

6. Close the connection to the free resources.

```
// Final housekeeping: free resources
queueConnection.close();
```

Developing applications for non-interruptible processes

You can develop the following applications for non-interruptible processes:

- Execute a non-interruptible process using the Enterprise Java Bean (EJB) interface
- Execute a non-interruptible process using the Java Message Service (JMS) interface

Also see the Javadoc.

Executing a non-interruptible process using the EJB interface

You can use the Enterprise Java Bean (EJB) interface to run non-interruptible processes.

1. Optional: List the process templates to find the name of the non-interruptible process you want to execute. This step is optional if you already know the name of the process.

```
ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.CAN_RUN_SYNC=TRUE",
 "PROCESS_TEMPLATE_NAME",
 newInteger(50),
 null);
```

   The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as non-interruptible processes.

2. Start the process with an input message. In the following example, Customer and OrderNo are message types known to the system.

```
Customer input = new Customer("Smith");
...
ClientObjectWrapper output =
      process.call("CustomerTemplate", new ClientObjectWrapper(input));
OrderNo order = (OrderNo) output.getObject();
```

   This action creates an instance of the process template, CustomerTemplate, and passes some customer data. The operation returns only when the process is complete. The result of the process, OrderNo, is returned to the caller.

Executing a non-interruptible process using JMS

You can use the Java Message Service (JMS) interface to run non-interruptible processes.

1. Create a message, for example, an ObjectMessage message.

```
ObjectMessage message = queueSession.createObjectMessage();
```

2. Optional: Set the JMSReplyToQueue. If you do not want to receive a reply, this step is optional.

```
//Specify the destination object replies are to be sent to
message.SetJMSReplyTo(replyToQueue);
```

3. Optional: Specify the JMS properties. If you do not specify any properties, the process template name Dispatch is assumed. If JMSReplyToQueue is set, call is issued. If JMSReplyToQueue is not set, initiate is issued.

```
message.SetStringProperty("wf$verb", "call");
message.SetStringProperty("wf$processTemplateName", "CustomerTemplate");
```

4. Start the process with an input message. Specify the input message as the body, the payload, of the message. In the example, Customer is a message type known to the system.

```
//Create Customer input message
Customer input = new Customer();
input.setLastName("Smith");

message.setObject(new ClientObjectWrapper(input));

//Send message
queueSender.send(message);
```

   This action creates an instance of the process template, CustomerTemplate, and passes some customer data. The operation returns only when the flow is complete and when a JMSReplyToQueue is specified. The result of the process, OrderNo, is returned as the payload of the reply message. Because an ObjectMessage message was passed, an object message is returned.

5. Optional: Get the result of the process. You can get the results of the process only if you specified a queue in step 2. In the example, OrderNo is a message type known to the system.

```
Message m = queueReceiver.receive();
if (m instanceof ObjectMessage)
{
 ClientObjectWrapper wrapper = (ClientObjectWrapper)m.getObject();
 OrderNo output = (OrderNo)wrapper.getObject();
}
```

Characteristics of non-interruptible business processes

A non-interruptible business process has the following characteristics:
- Runs as one transaction.
- Consists of only synchronous services, enterprise beans, Java snippets, empty activities, and non-interruptible subprocesses.
- Usually started using the call method so that an output message is returned when the process is complete.
- Normally short running.
- Is not visible during execution because run-time values are not stored in the database.
- Cannot contain interruptible processes.

Developing applications for interruptible processes

You can use the Enterprise Java Bean (EJB) interface to develop the following services for interruptible processes:
- Start an interruptible process
- Process a person activity
- Send an event to a process instance
- Analyze the result of a process
- Use work lists to query information

You can use the Java Message Service (JMS) interface to develop the following services for interruptible processes:
- Start an interruptible process
- Send an event to a process instance
- Analyze the result of a process

For more information, see the Javadoc.

Characteristics of interruptible processes

An interruptible process has the following characteristics:
- Runs as several transactions.
- Can consist of services, enterprise beans, Java snippets, event, person, empty, and process activities.
- Usually started using the initiate method because the output message cannot be retrieved synchronously.
- Normally long running.
- Is visible during execution because run-time values are stored persistently.

Event activities

An event is an asynchronous notification that can be sent to a process instance. It is used to synchronize the execution of a process instance with external systems. An event activity waits for the occurrence of an external event; several event

activities might be waiting for the same event. All of these activities receive the external event when it is sent. The event is consumed; subsequent event activities waiting for the same external event require a new event with the same name to be sent.

Person activities

When a person activity is activated, the process engine creates work items and distributes them to the potential owners of the activity. One of these potential owners claims the associated activity, works with the associated data and components, and completes the activity. The completion of the activity triggers the next transaction and continues the process. You can only use the EJB interface to process person activities.

Starting an interruptible process using the EJB interface

1. Optional: List the process templates to find the name of the interruptible process you want to start. This step is optional if you already know the name of the process.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.CAN_RUN_INTERRUP=TRUE"
    "PROCESS_TEMPLATE.NAME",
    newInteger(50),
    null);
   ```

   The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as interruptible processes.
2. Start the process with an input message of the appropriate type. If you specify a process-instance name, it must not start with an underscore. If a process-instance name is not specified, the process instance ID (PIID) in String format is used as the name.

   ```
   Customer input = new Customer("Smith");
   PIID piid = process.initiate("CustomerTemplate", "CustomerOrder",
                                new ClientObjectWrapper(input));
   ```

   This action creates an instance, CustomerOrder, of the process template, CustomerTemplate, and passes some customer data. When the process starts, the operation returns the object ID of the new process instance to the caller.

   The starter of the process instance is set to the caller of the request and receives a work item for the process instance. The process administrators, readers, and editors of the process instance are determined and receive work items for the process instance. The starting activity instances are determined and either started automatically or, if they are person activities or receive events, work items are created for the potential owners.

Processing person activities using the EJB interface

1. List the activities belonging to a logged-on person that are ready to be worked on:

   ```
   QueryResultSet result =
        process.query("ACTIVITY.AIID",
                      "ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
                       WORK_ITEM.REASON =
                           WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                       null, null, null);
   ```

This action returns a query result set that contains the activities that can be started by the logged-on person.

2. Claim the activity to be worked on:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 {
  ClientObjectWrapper input = process.claim(aiid);
  Order activityInput = (Order) input.getObject();
 }
}
```

When the activity is claimed, the input message of the activity is returned.

3. When work on the activity is complete, complete the activity.

```
OrderNo output = new OrderNo(4711);
process.complete (aiid, new ClientObjectWrapper(output));
```

Sending an event to a process instance using the EJB interface

1. Optional: List the processes that are waiting for a specific event from the logged-on user.

```
QueryResultSet result =
     query("DISTINCT EVENT.PIID",
           "EVENT.NAME = 'OrderEvent'
            AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
            null, null, null);
```

2. Send an event. The caller must be a potential owner of the awaited OrderEvent event or an administrator of the process instance, CustomerOrder.

```
if (result.size() > 0)
{
 result.first();
 Order input = new Order("Chocolate");
 process.sendEvent ((PIID)result.getOID(1), "OrderEvent",
                      new ClientObjectWrapper(input));
}
```

Sends the specified OrderEvent event to the waiting process instance and passes some order data.

Analyzing results of a process using the EJB interface

An interruptible process runs asynchronously. Its output message is not automatically returned when the process completes. The message must be retrieved explicitly. The results of the process are stored in the database only if the process template from which the process instance was derived does not specify automatic deletion of the output message.
Analyze the results of the process:

```
QueryResultSet result =
     process.query("PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                    PROCESS_INSTANCE.STATE = PROCESS_INSTANCE.STATE.STATE_FINISHED",
                    null, null, null);
if (result.size() > 0)
{
 result.first();
```

```
 PIID piid = (PIID) result.getOID(1);
 ClientObjectWrapper output = process.getOutputMessage(piid);
 OrderNo order = (OrderNo) output.getObject();
}
```

Using worklists to query information

A worklist is a query that is persistently stored in the database. This worklist represents a set of items that have the same characteristics. Although worklist definitions are stored persistently, items contained in the worklist are assembled dynamically when they are queried. All worklists are publicly accessible.

1. Optional: List the available worklists:

   ```
   String[] worklists = process.getWorklistNames();
   ```

2. Optional: Check the query defined by a specific worklist:

   ```
   WorkListData worklist = process.getWorklist("CustomerOrdersStartingWithA");
   String selectClause = worklist.getSelectClause();
   String whereClause = worklist.getWhereClause();
   String orderByClause = worklist.getOrderByClause();
   Integer threshold = worklist.getThreshold();
   ```

3. Run the query defined by the worklist: QueryResultSet result = process.executeWorklist(″CustomerOrdersStartingWithA″);

Starting an interruptible process using the JMS interface

1. Create a message, for example, an ObjectMessage message.

   ```
   ObjectMessage message= queueSession.createObjectMessage();
   ```

2. Optional: Set the JMSReplyToQueue. If you do not want to receive a reply, this step is optional. You cannot specify a temporary queue as a reply-to queue.

   ```
   // Specify the destination object replies are to be sent to
   message.SetJMSReplyTo(replyToQueue);
   ```

3. Set the Java Message Service (JMS) properties. If you do not specify any properties, the process-template name, Dispatch is assumed. If JMSReplyToQueue is set, call is issued. If JMSReplyToQueue is not set, initiate is issued. If the process-instance name is set, it must not start with an underscore. If the process-instance name is not set, the process instance ID (PIID) in String format is used as the name.

   ```
   message.SetStringProperty("wf$verb", "initiate");
   message.SetStringProperty("wf$processTemplateName",  "CustomerTemplate");
   message.SetStringProperty("wf$processInstanceName", "CustomerOrder");
   ```

4. Start the process with an input message. Specify the input message as the body, the payload, of the message. In the following, Customer is a message type known to the system.

   ```
   // Create Customer input message
   Customer input= new Customer();
   input.setLastName("Smith");

   message.setObject(new ClientObjectWrapper(input));

   // Send message
   queueSender.send(message);
   ```

   This action creates an instance, CustomerOrder, of the process template, CustomerTemplate, and passes some customer data. The operation returns the object ID of the newly created instance as the value of the JMS property, wf$piid, if a JMSReplyToQueue is specified. The reply message does not contain a payload.

5. Get the result of the process initiation.

```
Message m = queueReceiver.receive();
String fiid = m.getStringProperty("wf$piid");
```

Sending an event to a process instance using the JMS interface

1. Create a message, for example, an ObjectMessage message.

```
ObjectMessage message= queueSession.createObjectMessage();
```

2. Set the JMSReplyToQueue. If you do not want to receive a reply, this step is optional.

```
// Specify the destination object replies are to be sent to
message.SetJMSReplyTo(replyToQueue);
```

3. Set the JMS properties. You can specify the process instance ID (PIID) of the process instance instead of the process-instance name. If you specify both properties, the processInstanceName is used.

```
message.SetStringProperty("wf$verb", "sendEvent");
message.SetStringProperty("wf$processInstanceName",  "CustomerOrder");
message.SetStringProperty("wf$event",  "OrderEvent");
```

4. Send the specified OrderEvent event to the process instance, CustomerOrder.

```
// Create event input message
Order input = new Order("Chocolate");

message.setObject(new ClientObjectWrapper(input));

// Send message
queueSender.send(message);
```

   If JMSReplyToQueue is set, an empty reply message returns if the event was sent successfully. The JMSCorrelationID is set to the JMSMessageID of the sendEvent request. Neither properties nor payload are set on the reply message.

Analyzing results of a process using the JMS interface

An interruptible process runs asynchronously. Its output message is not automatically returned when the process completes. The message must be retrieved explicitly. You can get the results of the process only if you specified a JMSReplyToQueue and used the call verb to instantiate the process.
Analyze the results of the process: In the example, OrderNo is a message type known to the system. When an ObjectMessage is passed in the request, an object message is returned.

```
Message m = queueReceiver.receive();
if (m instanceof ObjectMessage)
{
 ClientObjectWrapper wrapper = (ClientObjectWrapper)m.getObject();
 OrderNo output = (OrderNo)wrapper.getObject();
}
```

Developing administration applications for interruptible processes

You can develop the following administration applications for interruptible processes. You can use the Enterprise Java Bean (EJB) interface to render all of these applications. The Java Message Service (JMS) interface can be used only to terminate a process instance.

- Cancel a claimed activity

- Force the completion of an activity
- Retry the execution of a stopped activity
- Delete a process instance
- Terminate a process instance using the EJB interface
- Terminate a process instance using the JMS interface
- Manage work lists

For more information, see the Javadoc.

Canceling a claimed activity

Sometimes it is necessary for someone with process administrator rights to cancel an activity that is claimed by someone else. This situation might occur, for example, when an activity must be completed but the owner of the activity is absent.

1. List the claimed activities owned by a specific person to find the ID of the activity in question.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_CLAIMED AND
                    ACTIVITY.OWNER = 'Smith'
                    AND ACTIVITY.TEMPLATE_NAME = 'CustomerTemplate'",
                    null, null, null);
```

   This action returns a query result set that lists the activities claimed by the specified person, Smith.

2. Cancel the claimed activity.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 process.cancelClaim(aiid);
}
```

   This action returns the activity to the ready state so that it can be claimed by one of the other potential owners.

Forcing the completion of an activity

If an activity in an interruptible process encounters a system exception or an unconnected fault terminal and the associated activity template specifies that the activity stops when an error occurs, the activity is put into the stopped state so that it can be repaired. In this state, you can force completion of the activity. You can also pass parameters in the force-complete call, such as the message that should have been computed or the fault that should have been raised.

1. List the stopped activities.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                    null, null, null);
```

   This action returns the stopped activities for the CustomerOrder process instance.

2. Complete the activity. In this example, an output message is passed:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 OrderNo output = new OrderNo(4711);
 process.forceComplete(aiid, new ClientObjectWrapper(output), true);
}
```

For more information, see the Javadoc. The activity is complete. If a system error occurs, the **continueOnError** parameter determines whether the activity stays in the stopped state. In the example, `continueOnError=true`. This value means that if an error occurs during processing of the forceComplete request, the activity is put into the failed execution state. Navigation continues and the process is put into the failing state.

Retrying the execution of a stopped activity

If an activity in an interruptible process encounters a system exception or an unconnected fault terminal and the associated activity template specifies that the activity stops when an error occurs, the activity is put into the stopped state so that it can be repaired. You can retry the execution of the activity by passing a new input message.

1. List the stopped activities.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                    null, null, null);
```

   This action returns the stopped activities for the CustomerOrder process instance.

2. Retry the execution of the activity.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 Order input = new Order("Chocolate");
 process.forceRetry(aiid, new ClientObjectWrapper(input), true);
}
```

   For more information, see the Javadoc. This action retries the activity. If a system error occurs, the **continueOnError** parameter determines whether the activity stays in the stopped state. In the example, `continueOnError=true`. If an error occurs during processing of the forceRetry request, the activity is put into the failed execution state. Navigation continues and the process is put into the failing state.

Deleting a process instance

Process instances are only automatically deleted when they complete if this is specified in the process template from which the process instances are derived. To delete all finished process instances:

1. List the process instances that are finished.

```
QueryResultSet result =
     process.query("DISTINCT PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                            PROCESS_INSTANCE.STATE.STATE_FINISHED",
                    null, null, null);
```

This action returns a query result set that lists finished process instances.

2. Delete the finished process instances.

```
PIID piid;
while (result.next() )
{
 piid = (PIID) result.getOID(1);
 process.delete(piid);
}
```

Terminating a process instance using the EJB interface

Sometimes it is necessary for someone with process administrator rights to terminate a process instance that is known to be in an unrecoverable state. For example, when an application is invoked and fails and does not return to a dormant state.

It is recommended that you terminate a process instance only in exceptional situations. The process instance is terminated immediately without waiting for any outstanding activities. Process instances that are terminated are not compensated.

1. Retrieve the process instance to be terminated.

```
ProcessInstanceData processInstance =
       process.getProcessInstance("CustomerOrder");
```

2. Terminate the process instance.

```
PIID piid = processInstance.getID();
process.forceTerminate(piid);
```

The process instance is terminated immediately without waiting for any outstanding activities.

Terminating a process instance using the JMS interface

Sometimes it is necessary for someone with process administrator rights to terminate a process instance that is known to be in an unrecoverable state. For example, when an application is invoked and fails and does not return to a dormant state.

It is recommended that you terminate a process instance only in exceptional situations. The process instance is terminated immediately without waiting for any outstanding activities. Process instances that are terminated are not compensated.

1. Create a message, for example, an ObjectMessage message.

```
ObjectMessage message= queueSession.createObjectMessage();
```

2. Optional: Set the JMSReplyToQueue. If you do not want to receive a reply, this step is optional.

```
// Specify the destination object replies are to be sent to
message.SetJMSReplyTo(replyToQueue);
```

3. Set the Java Message Service (JMS) properties.

```
message.SetStringProperty("wf$verb", "forceTerminate");
message.SetStringProperty("wf$processInstanceName", "CustomerOrder");
```

4. Terminate the CustomerOrder process instance.

```
// Send message
queueSender.send(message);
```

The process instance is terminated immediately without waiting for any outstanding activities. If JMSReplyToQueue is set, this action returns an empty

reply message if the process instance was terminated successfully. The JMSCorrelationID value is set to the JMSMessageID value of the forceTerminate request. Neither properties nor payload are set on the reply message.

Managing worklists

A worklist is a query that is stored persistently in the database. This worklist represents a set of items that have the same characteristics. Although worklist definitions are stored persistently, items contained in the worklist are assembled dynamically when they are queried. All worklists are publicly accessible.

1. Create a worklist To create a worklist, save a query with a specific name:

```
process.newWorklist("CustomerOrdersStartingWithA",
             "PROCESS_INSTANCE.NAME, DISTINCT PROCESS_INSTANCE.PIID",
             "PROCESS_INSTANCE.NAME LIKE 'A%'",
             "PROCESS_INSTANCE.NAME",
             null,
             null);
```

   This query returns a sorted list of all the process-instance names that begin with the letter A and their associated process instance IDs (PIID).

2. Delete a worklist.

```
process.deleteWorklist("CustomerOrdersStartingWithA");
```

Authorization for EJB renderings

Security must be enabled in WebSphere Application Server. When an instance of the BusinessProcess session bean is created, WebSphere Application Server associates a session context with the instance. The session context contains the caller's principal. This information is used by both the container and the process engine to check the caller's authorization for each call.

The following reasons for a work-item assignment are used:
- For processes: reader, starter, administrator
- For activities: reader, editor, potential owner, owner

These assignment reasons are mapped to authorization authorities:
- Activity reader authority: can see properties of the associated activity instance, and its input and output messages
- Activity editor authority: has the authority of the activity reader, and has write access to messages and other data associated with the activity
- Potential activity owner authority: has the authority of the activity editor, and has the right to claim the activity
- Activity owner authority: has the authority of the potential activity owner, and has the right to complete the activity
- Process starter authority: can see properties of the associated process instance, its input and output messages, and write other data associated with the process
- Process reader authority: can see properties of the associated process instance, its input and output messages, and everything that the activity reader supports for all contained activities, including those in blocks, but not those of the independent subprocesses
- Process administrator authority: has the authority of the process reader and the process starter, and the right to intervene in a process that has started

Special authority is granted to a person with the role of business process administrator. A business process administrator is a special role; it is different from the process administrator of a process instance. A business process administrator has all privileges.

You cannot delete the user ID of the process starter from your user registry while the process instance exists. If you do, the navigation of this process cannot continue. You receive the following exception in the system log file:

```
no unique ID for: <user ID>
```

Required authorizations for process requests

Access to the remote BusinessProcess interface does not guarantee that the caller can perform all actions on a process; the caller must also be authorized to perform the request. The following minimum authorization authorities are needed for process requests:

Request	Required authorization
createMessage	reader
getActivityInstance	reader
getAllActivities	reader
getAllWorkItems	reader
getCustomAttribute	reader
getEventNames	reader
getFaultMessage	reader
getFaultTerminalNames	reader
getInputMessage	reader
getOutputMessage	reader
getProcessInstance	reader
getVariable	reader
getUISettings	reader
setCustomAttribute	starter
delete	process administrator
forceTerminate	process administrator

Required authorizations for activity requests

Access to the remote BusinessProcess interface does not guarantee that the caller can perform all actions on an activity; the caller must also be authorized to perform the request. The following minimum authorization authorities are needed for activity requests:

Request	Required authorization
createMessage	activity reader or process reader
getActivityInstance	activity reader or process reader
getCustomAttribute	activity reader or process reader
getFaultMessage	activity reader or process reader
getFaultTerminalNames	activity reader or process reader

Request	Required authorization
getInputMessage	activity reader or process reader
getOutputMessage	activity reader or process reader
getOutputTerminalNames	activity reader or process reader
getUserInput	activity reader or process reader
getUISettings	activity reader or process reader
setCustomAttribute	activity editor or process administrator
setOutputMessage	activity editor or process administrator
setFaultMessage	activity editor or process administrator
setUserInput	activity editor or process administrator
claim	potential activity owner or process administrator
sendEvent	potential activity owner or process administrator
cancelClaim	activity owner or process administrator
complete	activity owner or process administrator
forceRetry	process administrator
forceComplete	process administrator

Authorization for JMS renderings

Security must be enabled in WebSphere Application Server. When an instance of the BusinessProcess message-driven bean (MDB) is deployed, the role MDBUser role must map to a specific user ID. This user ID is used by both the business process container and the process engine to check the caller's authorization for each request.

The following authorization authorities are needed:

Request	Required authorization
forceTerminate	process administrator
sendEvent	potential activity owner or process administrator

Special authority is granted to a person with the role of business process administrator. A business process administrator is a special role; it is different from the process administrator of a process instance. A business process administrator has all privileges.

You cannot delete the user ID of the process starter from your user registry while the process instance exists. If you do, the navigation of this process cannot continue. You receive the following exception in the system log file:

```
no unique ID for: <user ID>
```

Structure of a process choreographer JMS message

A Java Message Service (JMS) message consists of:

- A message header for message identification and routing information.

- Properties for JMS-specific data, application-specific data, and provider-specific data (optional).
- The body (payload) of the message that holds the content (optional).

Process choreographer supports text, object, and bytes-message formats.

Message header

A process choreographer client application can set the following fields:
- JMSReplyTo

  The destination to send a reply to the request. If this field is not set, a reply is not returned.
- JMSMessageID

  Uniquely identifies a message. This value is set by process choreographer when the message sent returns. This identifier is used as the JMSCorrelationID in the reply message.
- JMSCorrelationID

  Links messages. Do not set this field. A process choreographer reply message contains the JMSMessageID of the request message.

Data properties

This data is passed as name-value pairs. Process choreographer adds the following properties:
- wf$verb

  The function to call. Possible values are: initiate, call, forceTerminate, sendEvent.

  If the wf$verb property is not set and the JMSReplyTo field is set, then call is issued. If neither the wf$verb property nor the JMSReplyTo field are set, then initiate is issued.
- wf$processTemplateName

  The name of the process template to instantiate. This template name must be set for the call and the initiate functions. If the template name is not set, Dispatch is assumed.
- wf$piid

  The object ID of the process instance. You can use this ID with the forceTerminate and the sendEvent functions to identify the process instance. The value is set as the result of the initiate function if the JMSReplyTo filed is set. If both the wf$piid and the wf$processInstanceName values are set, the value for wf$processInstanceName is used.
- wf$processInstanceName

  The name of the process instance. You can use this name with the forceTerminate and the sendEvent functions to identify the process instance to process. It can be used with the call and the initiate functions to specify the process instance name of the process instance to create.
- wf$eventName

  The name of the event to send. This value must be set for the sendEvent function.
- wf$processState

  The state of the process instance. This value is set in reply messages.
- wf$exceptionText

  Specifies the message text of the exception that ended the process.

Message payload

Use to specify input, output, and fault messages.

- Request messages
  - TextMessage must contain an object of the type String. This object contains the input message to pass. It requires that the defined input message is also of type String.
  - ObjectMessage must contain an object of the type ClientObjectWrapper. This object contains the input message to pass.
  - BytesMessage must contain a byte array that represents the streamed version of the input message to pass.
- Reply messages
  - If possible, the reply message is the same format as the request message. That is, if the request message has a TextMessage format, then the reply message also has a TextMessage format. If a TextMessage format cannot be used, for example, because the type of the process instance output message is not a String, then a message with ObjectMessage format is returned.
  - If an ObjectMessage message is returned, it contains an object of the type ClientObjectWrapper. This object contains the output message or the message of the fault terminal.
  - If a BytesMessage message is returned, it contains the streamed output message or the streamed message of the fault terminal.
  - If a process can be navigated until its output or fault terminal is reached, the wf$processInstanceName and the wf$processState properties are set.
  - If an exception occurs during the processing of a request or if an exception occurs during the execution of a process instance and the fault is not connected to a fault terminal of the process, the reply message does not contain a payload. The wf$processInstanceName, the wf$processState, and the wf$exceptionText properties are set. The wf$exceptionText property contains the message text of the exception that ended the process. Nested exceptions follow the message text. These exceptions are separated by new-line characters.

Queries on business-process objects

You can use the process query interface to retrieve business-process information that is stored persistently. You use SQL-like syntax to query the following objects:

- Process templates
- Process instances
- Activity instances
- Work items

The query function is provided by the remote interface of the BusinessProcess session bean. For process templates, the query function has the following syntax:

```
queryProcessTemplates (java.lang.String whereClause,
                       java.lang.String orderByClause,
                       java.lang.Integer threshold,
                       java.util.Timezone timezone);
```

For the other business-process objects, the query function has the following syntax:

```
QueryResultSet query (java.lang.String selectClause,
                      java.lang.String whereClause,
                      java.lang.String orderByClause,
                      java.lang.Integer threshold,
                      java.util.Timezone timezone);
```

The query is made up of:

- Select clause

- Where clause
- Order-by clause
- Threshold parameter
- Timezone parameter

For example, a list of work items accessible to the caller of the function is retrieved by:

```
QueryResultSet result = process.query("WORK_ITEM.WIID",
                                      null, null,
                                      null, null);
```

The query function returns objects according to the caller's authorization. The query result set contains only those objects that the caller is authorized to see.

For more information, see the Javadoc.

Predefined views for queries on business process objects

Process choreographer provides the following predefined views for queries on business process objects:

- PROCESS_TEMPLATE
- PROCESS_INSTANCE
- PROCESS_ATTRIBUTE
- ACTIVITY
- ACTIVITY_ATTRIBUTE
- EVENT
- WORK_ITEM

PROCESS_TEMPLATE view

Process choreographer provides the following predefined view for queries on process templates:

Column name	Type	Comments
PTID	ID	Process template ID.
NAME	String	Name of the process template.
APPLICATION_NAME	String	Name of the enterprise application to which the process template belongs.
VALID_FROM	Timestamp	The time from when the process template can be instantiated.
VERSION	String	User-defined version.
CREATED	Timestamp	The time the process template is created in the database.
STATE	Integer	Specifies whether the process template is available to create process instances. Possible values: STATE_STARTED STATE_STOPPED
DESCRIPTION	String	Description of the process template.
CATEGORY	String	The category to which the process template belongs.
CAN_RUN_SYNC	Boolean	Specifies if the process can run non-interrupted.
CAN_RUN_INTERRUP	Boolean	Specifies if the process can run interrupted.

PROCESS_ATTRIBUTE view

Process choreographer provides the following predefined view for queries on process attributes:

Column name	Type	Comments
PIID	ID	The ID of the process instance that has a customer attribute.
NAME	String	Name of the customer attribute.
VALUE	String	Value of the customer attribute.

PROCESS_INSTANCE view

Process choreographer provides the following predefined view for queries on process instances

Column name	Type	Comments
PTID	ID	Process template ID.
PIID	ID	Process instance ID.
NAME	String	Name of the process instance.
STATE	Integer	State of the process instance. Possible values: STATE_READY STATE_RUNNING STATE_FINISHED STATE_COMPENSATING STATE_FAILED STATE_TERMINATED STATE_COMPENSATED STATE_TERMINATING STATE_FAILING
CREATED	Timestamp	The time the process instance is created.
STARTED	Timestamp	The time the process instance is started.
COMPLETED	Timestamp	The time the process instance is completed.
PARENT_NAME	String	The name of the parent process instance.
TOP_LEVEL_NAME	String	The name of the top-level process instance. This name is the current process instance if there is no top level.
STARTER	String	Principal ID of the starter.
TEMPLATE_NAME	String	The name of the associated process template.
TEMPLATE_DESCR	String	Description of the associated process template.
TEMPLATE_CATEGORY	String	The category to which the associated process template belongs.

ACTIVITY view

Process choreographer provides the following predefined view for queries on activities:

Column name	Type	Comments
PIID	ID	Process instance ID.
AIID	ID	Activity instance ID.
PTID	ID	Process template ID.
ATID	ID	Activity template ID.
KIND	Integer	Kind of activity. Possible values: KIND_PROCESS_SUBFLOW KIND_PROCESS_BLOCK KIND_EMPTY KIND_SINK KIND_SOURCE KIND_ELEMENTAL KIND_FAULT KIND_PERSON KIND_EVENT
RUN_MODE	Integer	Possible values: RUN_MODE_SYNCHRONOUS RUN_MODE_INTERRUPTIBLE RUN_MODE_ATOMIC_SPHERE RUN_MODE_CHAINED
ACTIVATED	Timestamp	The time the activity is activated.
STARTED	Timestamp	The time the activity is started.
COMPLETED	Timestamp	The time the activity is completed.
STATE	String	State of the activity. Possible values: STATE_INACTIVE STATE_READY STATE_RUNNING STATE_SKIPPED STATE_FINISHED STATE_FAILED STATE_TERMINATED STATE_CLAIMED STATE_TERMINATING STATE_FAILING STATE_WAITING STATE_EXPIRED STATE_STOPPED
OWNER	String	Principal ID of the owner.
TEMPLATE_NAME	String	Name of the associated activity template.
TEMPLATE_DESCR	String	Description of the associated activity template.

ACTIVITY_ATTRIBUTE view

Process choreographer provides the following predefined view for queries on activity attributes:

Column name	Type	Comments
AIID	ID	The ID of the activity instance that has a customer attribute.
NAME	String	Name of the customer attribute.
VALUE	String	Value of the customer attribute.

EVENT view

Process choreographer provides the following predefined view for queries on events:

Column name	Type	Comments
EIID	ID	The ID of the awaited event.
AIID	ID	The ID of activity waiting for the event.
PIID	ID	The ID of the process instance that contains the event.
NAME	String	Name of the event.

WORK_ITEM view

Process choreographer provides the following predefined view for queries on work items:

Column name	Type	Comments
WIID	ID	Work item ID.
OWNER_ID	String	Principal ID of the owner.
EVERYBODY	Boolean	Flag indicating whether everybody owns this work item.
OBJECT_TYPE	Integer	Type of the associated object. Possible values: OBJECT_TYPE_ACTIVITY OBJECT_TYPE_PROCESS_INSTANCE OBJECT_TYPE_EVENT
OBJECT_ID	ID	ID of the associated object, for example, the associated process or activity.
ASSOC_OBJECT_TYPE	Integer	Type of object associated with, or containing, the associated object of the work item. Possible values: OBJECT_TYPE_ACTIVITY OBJECT_TYPE_PROCESS_INSTANCE OBJECT_TYPE_EVENT
ASSOC_OID	ID	ID of the object associated with, or containing, the associated object of the work item. For example, the process instance ID (PIID) of the process instance containing the activity instance for which this work item was created.

Column name	Type	Comments
REASON	Integer	The reason that the work item was assigned. Possible values: REASON_POTENTIAL_OWNER REASON_EDITOR REASON_READER REASON_OWNER REASON_POTENTIAL_STARTER REASON_STARTER REASON_ADMINISTRATOR

Select clause

The select clause describes the query result. It specifies a list of names that identifies the object properties (columns of the result) to return. Its syntax is the same as an SQL select clause; use commas to separate parts of the clause. Each part of the clause must specify a property from one of the predefined views. The columns returned in the QueryResultSet object appear in the same order as the properties specified in the select clause.

The select clause does not support SQL aggregation functions, such as AVG(), SUM(), MIN(), MAX(), or COUNT().

Example select clauses

- ″WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON″

  Gets the object types of the associated objects and the assignment reasons for the work items.
- ″DISTINCT WORK_ITEM.OBJECT_ID″

  Gets all object IDs of objects for which the caller has a work item without duplicates.
- ″ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON″

  Gets the names of the activities the caller has work items for and their assignment reason.
- ″ACTIVITY.STATE, PROCESS_INSTANCE.STARTER″

  Gets the states of the activities and the starters of their associated process instances.

If an error occurs during the processing of the select clause, a QueryUnknownTable or a QueryUnknownColumn exception is thrown with the name of the property that is not recognized as a table or column name.

Where clause

The where clause describes the filter criteria to apply to the query domain. Its syntax is the same as an SQL where-clause. If you do not want to filter a query, you must specify `null` for the where clause.

The where-clause syntax supports:

- Keywords: AND, OR, NOT
- Comparison operators: =, <=, <, <>, >,>=, LIKE
- Set operation: IN

The LIKE operation supports the wildcard characters defined for the queried database.

The following rules also apply:
- Specify object ID constants as ID('string-rep-of-oid').
- Specify timestamp constants as TS('yyyy-mm-ddThh:mm:ss'). To refer to the current date, specify CURRENT_DATE as the timestamp.

  You must specify at least a date or a time value in the timestamp:
  - If you specify a date only, the time value is set to zero.
  - If you specify a time only, the date is set to the current date.
  - If you specify a date, the year must consist of four digits; the month and day values are optional. Missing month and day values are set to 01. For example, TS('2003') is the same as TS('2003-01-01T00:00:00').
  - If you specify a time, these values are expressed in the 24-hour system. For example, if the current date is 1 January 2003, TS('T16:04') or TS('16:04') is the same as TS('2003-01-01T16:04:00').
- Specify binary constants as BIN('UTF-8 string')
- It is recommended that you use symbolic constants instead of integer enumerations. For example, instead of specifying an activity state expression "ACTIVITY.STATE=2", specify "ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY".

Examples of where clauses
- Comparing an object ID with an existing ID

  ```
  "WORK_ITEM.WIID = ID('_WI:800c00ed.df8d7e7c.feffff80.38')"
  ```

  This type of where clause is usually created dynamically with an existing object ID from a previous call. If this object ID is stored in a wiid1 variable, the clause can be constructed as:

  ```
  "WORK_ITEM.WIID = ID('" + wiid1.toString() + "')"
  ```
- Using timestamps

  ```
  "ACTIVITY.STARTED >= TS('2002-06-1T16.00.00')"
  ```
- Using symbolic constants

  ```
  "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER"
  ```
- Using boolean values true and false

  ```
  "PROCESS_TEMPLATE.CAN_RUN_SYNC = TRUE"
  ```

Order-by clause

Use the order-by clause to specify the sort criteria for the query result set. The order-by clause syntax is the same as an SQL order-by clause; use commas to separate each part of the clause. Each part of the clause must specify a property from one of the predefined views.

If you identify more than one property, the query result set is ordered by the values of the first property, then by the values of the second property, and so on.

If you do not want to sort the query result set, you must specify `null` for the order-by clause.

Examples of order-by clauses:
- "PROCESS_TEMPLATE.NAME"

  Sorts the query result alphabetically by the process-template name.
- "PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC"

Sorts the query result by the creation date and, for a specific date, sorts the results alphabetically by the process-instance name in reverse order.

- "ACTIVITY.OWNER, ACTIVITY_TEMPLATE.NAME, ACTIVITY.STATE"

  Sorts the query result by the activity owner, then the activity-template name, and then the state of the activity.

Threshold parameter

The threshold parameter in the query function restricts the number of objects returned in the query result set. This parameter can be useful, for example, in a graphical user interface where only a small number of items should be displayed. If you set the threshold parameter accordingly, it improves performance. The database query is faster and less data needs to transfer from the server to the client.

If the parameter is set to `null`, a threshold is not applied and all the qualifying objects are returned.

Example of a threshold parameter:

- new Integer(50)

  Specifies to return 50 qualifying objects.

Timezone parameter

Timezones can differ between the client that starts the query and the process engine that processes the query. Use the timezone parameter to specify the timezone of the timestamp constants used in the where clause, for example, to specify local times. The dates returned in the query result set have the same timezone as that specified in the query.

If the parameter is set to `null`, the timestamp constants are assumed to be Coordinated Universal Time (UTC) times.

Examples of timezone parameters:

- ```
  process.query("ACTIVITY.AIID",
              "ACTIVITY.STARTED > TS ('2002-01-01T17:40')",
              null,
              null,
              java.util.Timezone.getDefault() );
  ```

  Specifies to return object IDs for activities that started later than 17:40 local time on 1 January 2002.

- ```
  process.query("ACTIVITY.AIID",
              "ACTIVITY.STARTED > TS ('2002-01-01T17:40')",
              null,
              null,
              null;
  ```

  Specifies to return object IDs for activities that started later than 17:40 UTC on 1 January 2002. This specification is, for example, 6 hours earlier in Eastern Standard Time.

Query results

A query result set contains the results of a query. The elements of the set are objects that the caller is authorized to see. You can read elements in a relative fashion using the next() method or in an absolute fashion using the first() and last() methods. Because the implicit cursor of a query result set is initially

positioned before the first element, you must call either first() or next() methods before reading an element. You can use the size() method to determine the number of elements in the set.

An element of the query result set comprises the selected attributes of work items and their associated referenced objects, such as activity instances and process instances. The first attribute (column) of a QueryResultSet element specifies the value of the first attribute specified in the select clause of the query request. The second attribute (column) of a QueryResultSet element specifies the value of the second attribute specified in the select clause of the query request, and so on.

You can retrieve the values of the attributes by calling a method that is compatible with the attribute type and by specifying the appropriate column index. The numbering of the column indexes starts with 1.

Attribute type	Method
String	getString
ID	getOID
Timestamp	getTimestamp getString
Integer	getInteger getShort getLong getString getBoolean
Boolean	getBoolean getShort getInteger getLong getString

Example:

The following query is run:

```
QueryResultSet resultSet = process.query("ACTIVITY.STARTED,
                                         ACTIVITY.TEMPLATE_NAME AS NAME,
                                         WORK_ITEM.WIID, WORK_ITEM.REASON",
                                         null, null, null, null);
```

The returned query result set has four columns:
- Column 1 is a timestamp
- Column 2 is a string
- Column 3 is an object ID
- Column 4 is an integer

You can use the following methods to retrieve the attribute values:

```
while (resultSet.next())
{
 java.util.Calendar activityStarted = resultSet.getTimestamp(1);
 String templateName = resultSet.getString(2);
 WIID wiid = (WIID) resultSet.getOID(3);
 Integer reason = resultSet.getInteger(4);
}
```

You can use the display names of the result set, for example, as headings for a printed table. These names are the column names of the view or the name defined by the AS clause in the query. You can use the following methods to retrieve the display names in the example:

```
resultSet.getColumnDisplayName(1) returns "STARTED"
resultSet.getColumnDisplayName(2) returns "NAME"
resultSet.getColumnDisplayName(3) returns "WIID"
resultSet.getColumnDisplayName(4) returns "REASON"
```

Chapter 11. Troubleshooting process choreographer

Process choreographer uses the WebSphere Application Server JRas framework for traces, messages, and audit logs. The following topics contain troubleshooting information that applies to process choreographer only.

- Using process-related trace information
- Using process-related audit trail information
- Using process-related messages
- Troubleshooting the business process container

Using process-related trace information

Process choreographer uses the WebSphere Application Server JRas framework for traces.

The following process-specific information is relevant for process choreographer:

- The Java package for process choreographer is com.ibm.bpe. Use the com.ibm.bpe package to trace process-related events only.
- The trace information is stored in the trace file that corresponds to the Java package that you are using.

Using process-related audit trail information

When a process instance is started and audit trailing is enabled, process choreographer writes information about each significant event into an audit log, which is located in the corresponding database table. Process choreographer provides a plug-in for the audit trail database.

- Process choreographer - structure of the audit trail database views describes the structure of the audit trail database views.
- Process choreographer - audit event types contains a list and description of all available event type codes that can occur during the processing of processes.

Process choreographer - structure of the audit trail database views

5.1 + Process choreographer provides two database views to show audit log information. These views are:

- AUDIT_LOG

  Contains audit logs for both V5.0-style processes and processes based on the Business Process Execution Language (BPEL).
- AUDIT_LOG_B

  Contains audit logs for BPEL-based processes only.

To read the content of the audit trail, use SQL or any other administration tool that supports the reading of database tables and views.

Audit events are related to process entities. The audit event types depend on the entity to which the event refers. The audit event types include:

- Process instance events (PIE)

- Activity instance events (AIE)
- Events related to variables (VAR)
- Control link events (CLE)
- Process template events (PTE)
- 5.1 + Scope-related events (SIE). These events are applicable to BPEL processes only.

For a list of the possible audit event type codes refer to Process choreographer - audit event types.

The following table describes the structure of the AUDIT_LOG audit trail view. It lists the names of the columns, the event types, and gives a short description for the column. The BPEL column specifies whether the information is also shown in the AUDIT_LOG_B view.

Process choreographer - audit event types

The following tables list the types of audit events that can occur while processes are running with their corresponding codes. These events are generated only if the business relevance indicator is set for the business object type in the process model.

Table 3. Process instance events

Audit event	Event code
PROCESS_STARTED	21000
PROCESS_COMPLETED	21004
PROCESS_TERMINATED	21005
PROCESS_DELETED	21020
PROCESS_FAILED	42001
PROCESS_COMPENSATING	42003
PROCESS_COMPENSATED	42004
PROCESS_TERMINATING	42009
PROCESS_FAILING	42010
PROCESS_CORRELATION_SET_INITIALIZED	42027
PROCESS_COMPENSATION_INDOUBT	42030
PROCESS_WORKITEM_DELETED	42041
PROCESS_WORKITEM_CREATED	42042

Table 4. Activity events

Audit event	Event code
ACTIVITY_READY	21006
ACTIVITY_STARTED	21007
ACTIVITY_COMPLETED	21011
ACTIVITY_CLAIM_CANCELED	21021
ACTIVITY_CLAIMED	21022
ACTIVITY_TERMINATED	21027
ACTIVITY_FAILED	21080
ACTIVITY_EXPIRED	21081

Table 4. Activity events (continued)

Audit event	Event code
ACTIVITY_LOOPED	42002
ACTIVITY_TERMINATING	42008
ACTIVITY_FAILING	42011
ACTIVITY_OUTPUT_MESSAGE_SET	42012
ACTIVITY_FAULT_MESSAGE_SET	42013
ACTIVITY_STOPPED	42015
ACTIVITY_FORCE_RETRIED	42031
ACTIVITY_FORCE_COMPLETED	42032
ACTIVITY_UNDO_STARTED	42033
ACTIVITY_UNDO_COMPLETED	42035
ACTIVITY_UNDO_SKIPPED	42034
ACTIVITY_MESSAGE_RECEIVED	42036
ACTIVITY_LOOP_CONDITION_TRUE	42037
ACTIVITY_LOOP_CONDITION_FALSE	42038
ACTIVITY_WORKITEM_DELETED	42039
ACTIVITY_WORKITEM_CREATED	42040

Table 5. Events related to variables

Audit event	Event code
VARIABLE_UPDATED	21090

Table 6. Control link events

Audit event	Event code
LINK_EVALUATED_TO_TRUE	21034

Table 7. Process template events

Audit event	Event code
PROCESS_INSTALLED	42006
PROCESS_UNINSTALLED	42007

5.1 +

Table 8. Scope instance events

Audit event	Event code
SCOPE_STARTED	42020
SCOPE_FAILED	42021
SCOPE_FAILING	42023
SCOPE_TERMINATED	42024
SCOPE_COMPLETED	42026

Using process-related messages

Process choreographer uses the WebSphere Application Server JRas framework for handling messages. All messages that belong to process choreographer are prefixed with *BPE*. The format of process choreographer messages is *BPEComponentNumberTypeCode*. The type code can be:

I Information message

W Warning message

E Error message

You can find process choreographer messages in the `SystemOut.log` file, the `SystemError.log` file, and traces. If the message text provided in these files is not enough to help you solve your problem, you can use the WebSphere Application Server symptom database to find more information. To view process choreographer messages, check the `activity.log` file by using the WebSphere log analyzer.

1. 5.1+ Click **Start > Programs > IBM WebSphere > Application Server v5.1 > Log Analyzer**.
2. Optional: Click **File > Update database > WebSphere Application Server Symptom Database** to check for the newest version of the symptom database.
3. Optional: Load Activity log. Load the log from the:
   - *install_root*`\logs\activity.log` file on Windows systems
   - *install_root*`/logs/activity.log` file on UNIX systems and z/OS
4. Select the activity log file and click **Open**.

Chapter 12. Process choreographer: Resources for learning

Use the following links to find relevant supplemental information about process choreographer. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the process choreographer, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Planning, business scenarios, and IT architecture

- Process Choreographer Concepts and Architecture

  This paper uses scenarios to show how you can benefit from using process choreographer in today's business environment. It explains some basic business-process concepts that are used by process choreographer and describes how to develop, use, and administer business processes. An overview of the architecture is also given.
- Process Choreographer Staff Resolution Architecture

  This paper describes the architecture of the components involved in staff resolution. It explains the interaction between the process choreographer Web client, business process engine, work item manager, staff resolution plug-ins, and staff repositories, then focuses on the role of staff resolution.
- Web services and business process management

  This paper presents the relationship between Web services and the management of business processes in a tutorial-like manner.

Programming instructions and examples

- Process choreographer Samples and tutorial

  5.1+ Follow the instructions in the Samples Gallery to access the process choreographer Samples on your local machine. The process choreographer Samples includes three sample processes. Tutorials are also provided for each of the Samples.
- 5.1+ Business Process Execution Language for Web Services (BPEL) specification.

  Process choreographer includes support for BPEL-based processes. The specification describes the BPEL constructs in detail.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 USA

Trademarks and service marks

The following terms are trademarks of IBM Corporation in the United States, other countries, or both:

- AIX
- AS/400
- CICS
- Cloudscape
- DB2
- DFSMS
- Domino
- Everyplace
- iSeries
- IBM
- IMS
- Informix
- iSeries
- Language Environment
- Lotus
- MQSeries
- MVS
- OS/390
- RACF
- Redbooks
- RMF
- SecureWay
- SupportPac
- Tivoli
- ViaVoice
- VisualAge
- VTAM
- WebSphere
- z/OS
- zSeries

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

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

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

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