



Business Process Choreographer

Note

Before using this information, be sure to read the general information in "Notices" on page 353.

30March2007

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

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

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

© Copyright International Business Machines Corporation 2007. All rights reserved.

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

Contents

Chapter 1. Planning to use Business Process Choreographer	1
About Business Process Choreographer	2
Business Process Choreographer and Network Deployment.	2
Business Process Choreographer scenarios for clustering.	4
Chapter 2. Configuring Business Process Choreographer.	11
Using the bpeconfig.jacl script file to configure Business Process Choreographer	13
Configuring the business process container using the installation wizard	25
Creating the queue manager and queues for the business process container	28
Creating the database for the business process container	30
Business process container installation wizard settings	34
Business process container settings	43
Customizing the WebSphere MQ JMS resources in a cluster	44
Configuring the human task container, using the installation wizard	46
Human task container installation wizard settings	48
Human task container settings	51
Configuring the LDAP staff plug-in provider	54
Staff service settings	56
Staff plug-in provider collection	56
Staff plug-in provider settings	57
Staff plug-in configuration collection	58
Staff plug-in configuration settings	59
About the staff service.	60
Overview: Configuring Business Process Choreographer Explorer	81
About Business Process Choreographer Explorer	81
Configuring Business Process Choreographer Explorer	82
Configuring the Business Process Choreographer Observer infrastructure.	84
About Business Process Choreographer Observer.	85
Configuring the Business Process Choreographer event collector.	86
Configuring Business Process Choreographer Observer.	89
Activating Business Process Choreographer.	90
Verifying that Business Process Choreographer works	91
Understanding the startup behavior of the business process container	91
Configuring Business Process Choreographer to use an LDAP user registry	92
Federating a stand-alone node that has Business Process Choreographer configured	95
Promoting a server that has Business Process Choreographer configured to a cluster	96
Chapter 3. Removing the Business Process Choreographer configuration	97
Using a script to remove the Business Process Choreographer configuration.	97
Using the administrative console to remove the Business Process Choreographer configuration	99
Using the administrative console to remove the Business Process Choreographer event collector	103
Using the administrative console to remove Business Process Choreographer Observer.	105
Chapter 4. Administering	107
Using Business Process Choreographer Explorer.	107
Business Process Choreographer Explorer user interface	107
Business Process Choreographer Explorer navigation pane	108
Starting Business Process Choreographer Explorer	110
Customizing Business Process Choreographer Explorer	111
Administering Business Process Choreographer	120
Using the administrative console to administer Business Process Choreographer	120
Using scripts to administer Business Process Choreographer.	128
Administering business processes and human tasks	139
About business processes	139
About human tasks	145

Administering process templates and process instances	148
Administering task templates and task instances	157
Reporting on business processes and activities	164
Chapter 5. Developing	165
Developing client applications for business processes and tasks	165
Developing EJB client applications for business processes and human tasks	165
Developing Web service API client applications	243
Developing Web applications for business processes and human tasks, using JSF components	266
Chapter 6. Deploying	287
Installing business process and human task applications	287
Deployment of models	287
Deploying business process applications interactively	288
When you can install a process application on a cluster in which no servers are running	289
Uninstalling business process and human task applications, using the administrative console	291
Uninstalling business process and human task applications, using administrative commands	291
Chapter 7. Monitoring	293
Monitoring business processes and human tasks	293
Situation-independent event data	293
Business process events	294
Human task events	305
Chapter 8. Tuning business processes	313
Tuning long-running processes	314
Specifying initial database settings	314
Planning messaging engine settings	317
Tuning the application server	318
Fine-tuning the messaging provider	319
Fine-tuning the database	319
Tuning microflows	321
Tuning business processes that contain human tasks	321
Reduce concurrent access to human tasks	322
Reduce query response time	322
Avoid scanning whole tables	322
Chapter 9. Troubleshooting Business Process Choreographer	325
Troubleshooting the Business Process Choreographer configuration	325
Business Process Choreographer log files	325
Enabling tracing for Business Process Choreographer	326
The task container application fails to start	326
Troubleshooting the Business Process Choreographer database and data source	327
Troubleshooting business process and human tasks	329
Troubleshooting the installation of business process and human task applications	329
Troubleshooting the execution of business processes	330
Working with process-related or task-related messages	335
Troubleshooting Business Process Choreographer Explorer	336
Troubleshooting the administration of business processes and human tasks	337
Troubleshooting the staff service, staff plug-ins, and staff resolution	337
Using process-related and task-related audit trail information	342
Notices	353
Programming interface information	355
Trademarks and service marks	355

Chapter 1. Planning to use Business Process Choreographer

For each application server or cluster that runs business processes or human tasks, you will have to configure the business process container and the human task container before installing any enterprise applications that contain business processes or human tasks.

1. If you intend to use Business Process Choreographer on a cluster plan your cluster.
2. Decide which database system to use:

Note: The Business Process Choreographer Observer requires a database that is either Cloudscape or DB2.

- Cloudscape

Note:

- Because Cloudscape™ Network Server has no XA support, Business Process Choreographer can only use the embedded Cloudscape version that cannot be accessed remotely. This restriction is why Cloudscape cannot be used as database system for Business Process Choreographer in a clustered environment.
- Cloudscape serializes database access. Activities are therefore always performed sequentially, even in flows that are modeled to support the parallel execution of activities.

- DB2® for z/OS®
3. Check the requirements for the DB2 for z/OS Universal JDBC Driver provider and data source.
 4. Decide which server you want to host the database. If the database server is remote, you need a suitable database client or a type-4 JDBC driver that has XA-support.
 5. Decide which Java™ Message Service (JMS) provider you will use:
 - WebSphere® default messaging
 - WebSphere MQ
 6. Plan the settings that are described in “Business process container installation wizard settings” on page 34.
 7. Plan the settings that are described in “Human task container installation wizard settings” on page 48.
 8. Decide if you will configure the business process container manually (recommended) or if you will use the installation wizard to configure the business process container.
 - If you are going to configure the business process container manually, plan the settings as described in “Business process container settings” on page 43.
 - If you are going to use the installation wizard, plan the settings described in “Business process container installation wizard settings” on page 34.

After installing WebSphere Process Server, you are ready to perform Configuring Business Process Choreographer.

About Business Process Choreographer

Describes the facilities provided by the business process container and the human task container.

Business Process Choreographer is an enterprise workflow engine that supports both business processes and human tasks in a WebSphere Application Server environment. These constructs can be used to integrate J2EE resources, services, and activities that require human interaction. Business Process Choreographer manages the life cycle of business processes and human tasks, navigates through the associated model, and invokes the appropriate services.

Business Process Choreographer provides the following facilities:

- Support for business processes and human tasks. Business processes offer the standard way to model your business process using the Web Services Business Process Execution Language (WS-BPEL, abbreviated to BPEL). With human tasks, you can use the Task Execution Language (TEL) to model the interactions that involve humans, such as human-to-human, human-to-computer, computer-to-human. Both business processes and human tasks are exposed as services in a Service Oriented Architecture (SOA) or Service Component Architecture (SCA); they also support both simple data objects and business objects.
- Application programming interfaces for developing customized applications for interacting with business processes and human tasks.
- Business Process Choreographer Explorer. This Web application offers functions for managing and administering business process and human tasks. For more information refer to “About Business Process Choreographer Observer” on page 85.
- Business Process Choreographer Observer. This Web applications allows you to observe the states of running business processes and human tasks. For more information refer to “About Business Process Choreographer Explorer” on page 81.

Related concepts

“Business Process Choreographer and Network Deployment”

Describes special considerations when using Business Process Choreographer in a Network Deployment environment.

“Business Process Choreographer scenarios for clustering” on page 4

Describes different configuration options and considerations for Business Process Choreographer scenarios that use clusters.

Business Process Choreographer and Network Deployment

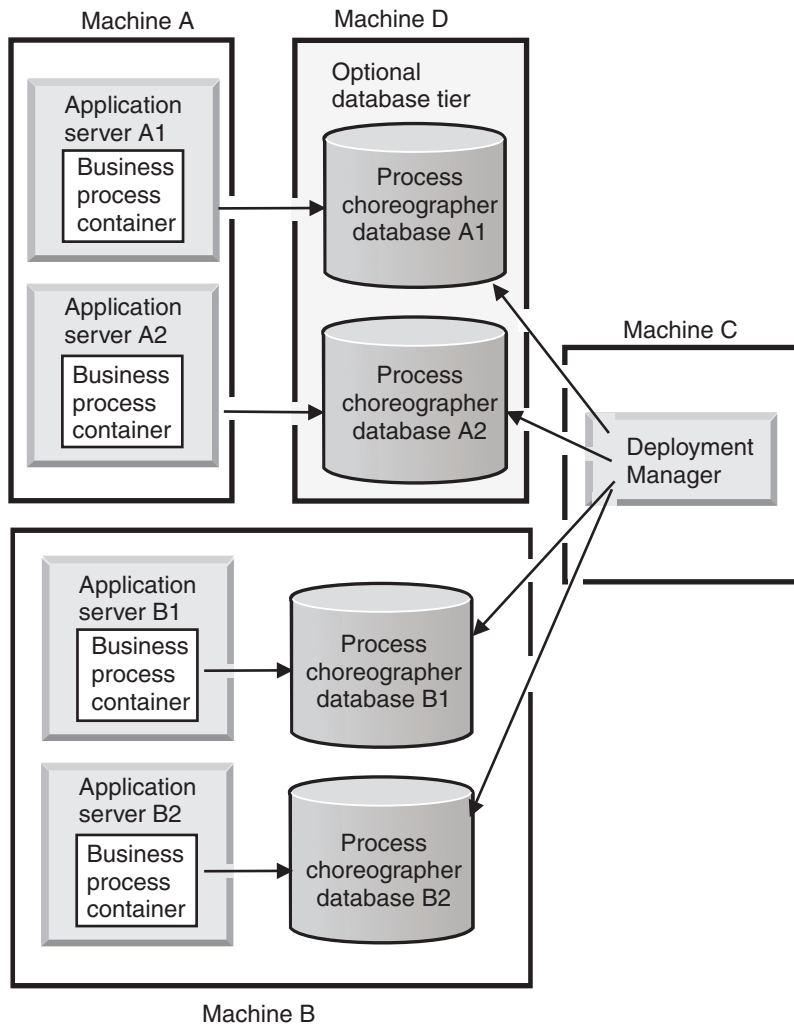
Describes special considerations when using Business Process Choreographer in a Network Deployment environment.

If you use Network Deployment (ND), the following points must be considered.

Deployment manager must have access to the Business Process Choreographer databases

The deployment manager must have access to all the Business Process Choreographer databases that are used by business process containers and human task containers in the cell. You must install an appropriate database client or JDBC driver on the computer that hosts the deployment manager and set the WebSphere

environment variable for the JDBC provider's classpath on the deployment manager's node scope. The following figure shows this configuration.



Before you install Business Process Choreographer on a cluster

Make sure that you have prepared your cluster as described in the *Administering WebSphere Process Server* PDF.

Customization required after installing and configuring Business 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 Business 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 Business Process Choreographer, see *Business Process Choreographer scenarios for clustering*

Before installing an application that contains business processes, or human tasks, or both

Make sure that the servers where you want to install the application are running before installing the application. At least one server must be running for Java Naming and Directory Interface (JNDI) names to be resolved.

Related concepts

“About Business Process Choreographer” on page 2

Describes the facilities provided by the business process container and the human task container.

Business Process Choreographer scenarios for clustering

Describes different configuration options and considerations for Business Process Choreographer scenarios that use clusters.

The main advantages of using WebSphere Process Server clusters for Business Process Choreographer instances are:

- High service availability due to failover
- Increased workload capacity
- Improved resource utilization
- Workload sharing
- Easier administration

Configuration options

You can configure Business 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 Process Server cell

One or more. All nodes are administered from a single deployment manager.

Number of nodes in each WebSphere Process Server cluster

One or more. Horizontal WebSphere clustering can increase service availability and the total workload capacity.

Number of application servers in each node

One or more. Vertical WebSphere Process Server clustering can increase resource utilization.

Database host

- Remote, on a dedicated server
- Local to one of the application servers in the cluster

It is recommended to host the database on a dedicated server, preferably one with a hot standby.

Application messaging queues

- Local queues
- Remote queues

Connection (WebSphere Platform Messaging)

When WebSphere Platform Messaging (WPM) is used, you can configure the message engines in the same cluster or in a remote cluster. For Business Process Choreographer, you should use the same approach used for the other WebSphere Process Server components. For more details about possible scenarios, see “Preparing a server or cluster to support service applications” in the *Administering WebSphere Process Server* PDF. The recommended configuration is to have the messaging engines run in a

different cluster than the cluster in which Business Process Choreographer is installed. This is similar to the central queue manager configuration that can be used with WebSphere MQ.

Connection (WebSphere MQ queue managers)

Restriction: Support for WebSphere MQ messaging in this product is limited to Business Process Choreographer (BPC). Except in BPC, WebSphere MQ messaging is not supported in this product.

- One central (remote) queue manager hosting the queues for the application servers within one cluster. This configuration is generally recommended.
- One local queue manager per application server. This provides no failover and no intraprocess workload sharing.
- Two local queue managers per node, and WebSphere MQ clustering used to balance workload across several application servers.

Workload distribution between different Business Process Choreographer instances requires that the queue managers that are used by the business process container of each application server are members of the same WebSphere MQ cluster. This configuration provides no failover and is not generally recommended.

WebSphere MQ is not recommended as JMS provider if a clustered scenario is used.

Database system

You can use any of the supported databases except Cloudscape.

Hot standby servers

You have the following options for hot standby servers:

- 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 Business Process Choreographer services, consider the following items:

- By creating cloned application servers in a WebSphere cluster, the services provided by the application servers become highly available.
- The Business 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 Business Process Choreographer can use the available system resources.

Workload sharing

If you are using WebSphere MQ messaging and you want different instances of Business Process Choreographer to share the same workloads, they must use one of the following queue manager configurations:

Central queue manager

A central queue manager hosts the queues that are needed by Business Process Choreographer. All Business 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 local queues and is used for getting messages, the other queue manager hosts no queues and is used only for putting messages. All the queue managers of all the Business 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.

Business Process Choreographer database

Hosting the database on a dedicated server, preferably one with a hot standby is recommended. The database can be on a server 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 Business 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 Business Process Choreographer instances in the WebSphere cell. 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 that is used by Business Process Choreographer instances in the same WebSphere cell must be accessible using a unique name. For DB2, 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®.

Business Process Choreographer Observer database

Business Process Choreographer Observer requires a database. It can use the same database as the Business Process Choreographer business process container, but in

a production system, it is preferable for performance reasons, to have a separate database.

WebSphere Platform Messaging JMS provider

Business Process Choreographer can use WebSphere Platform Messaging (WPM), which supports clustering, workload management, and failover.

Two topologies are supported:

- The messaging resources are hosted by a different cluster than the applications. This is the recommended topology since it provides failover capabilities together with low administration overheads. This topology is similar to the central queue manager approach in the WebSphere MQ case.
- The messaging resources and the applications are hosted by the same cluster. This topology is ideal for high performance, though it requires more administration effort, especially when changes are applied.

For more information about considerations that apply when WPM is used, refer to "Creating a clustered environment" in the *Administering WebSphere Process Server* PDF.

WebSphere MQ

Restriction: Support for WebSphere MQ messaging in this product is limited to Business Process Choreographer (BPC). Except in BPC, WebSphere MQ messaging is not supported in this product.

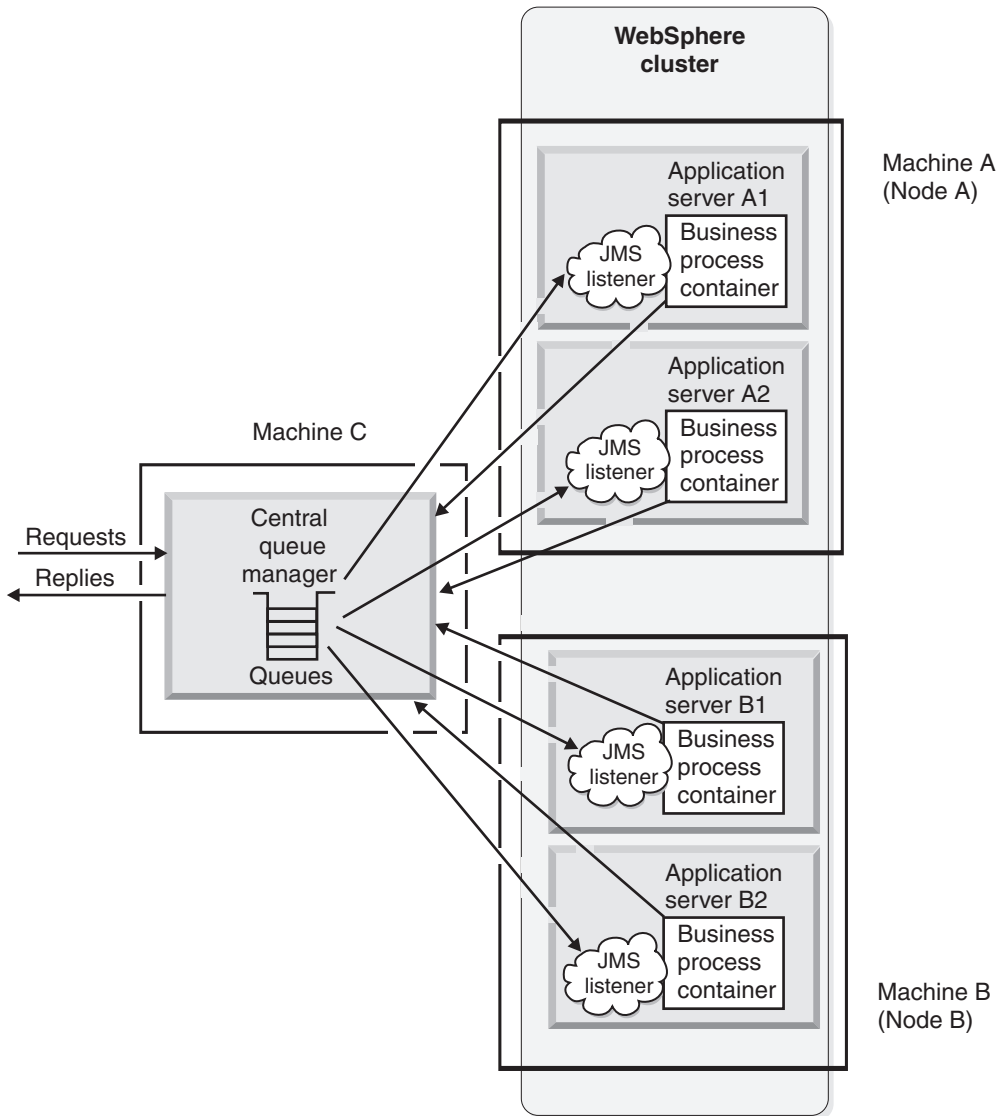
Business Process Choreographer can use WebSphere MQ queues for receiving requests and sending replies. WebSphere MQ is not recommended as the JMS provider if a clustered scenario is used. If you use WebSphere MQ, you must still configure the default messaging for the Service Component Architecture (SCA), which Business Process Choreographer uses for inbound and outbound service invocation. Each application server that hosts Business 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

Central queue manager

By using a central queue manager for all queues, administration becomes easier. One queue manager is used by all cloned containers for human tasks and business processes. 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 server. Every application server shown with a business process container, can also have a human task container.



Local queue manager without WebSphere MQ clustering

This example presents the standard, stand-alone Business Process Choreographer configuration. Each business process container has one local queue manager. This approach does not offer intraprocess workload sharing nor failover service availability.

WebSphere MQ clustering

This complex technique is not recommended. It supports intraprocess workload sharing for Business Process Choreographer services in a WebSphere cluster. The business processes in the cluster must all run on UNIX[®] only, or Windows[®] workstations only; a combination of UNIX and Windows servers 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

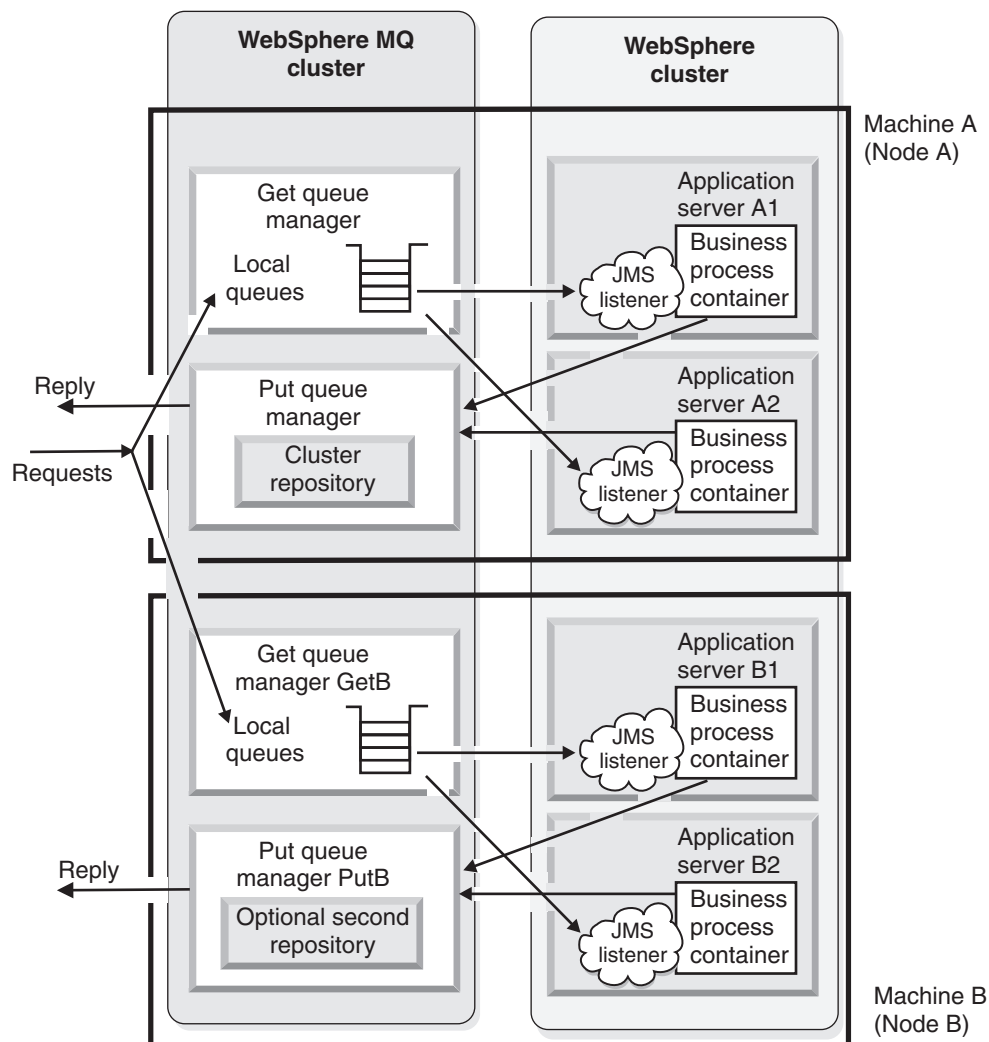
factories so that all the put queue managers use the binding protocol (interprocess communications) and all the get queue managers use the default, client (TCP/IP) protocol.

On Windows and UNIX systems, using the local bindings transport type is approximately 5% faster than using the client transport type, but has the effect that you must stop the entire application server to stop the local WebSphere MQ queue manager.

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 that are 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 Business Process Choreographer. If you have already configured a standalone server, perform “Promoting a server that has Business Process Choreographer configured to a cluster” on page 96, otherwise, the following sequence is recommended:

1. Create a cluster using the defaultProcessServer template as the server template for the cluster members.
2. Add members to the cluster.
3. Enable the cluster for service applications.
4. If you want to use the Business Process Choreographer Observer to monitor business processes and human tasks, make sure that the Common Event Infrastructure (CEI) is configured on the cluster.
5. Configure Business Process Choreographer on the cluster.
6. If you are using WebSphere MQ, and 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 Business Process Choreographer Install wizard configuration.

Related concepts

“About Business Process Choreographer” on page 2

Describes the facilities provided by the business process container and the human task container.

Chapter 2. Configuring Business Process Choreographer

This describes how to configure the Business Process Choreographer containers for business processes and for human tasks. It also describes how to configure Business Process Choreographer Explorer and Business Process Choreographer Observer.

If you have a Network Deployment (ND) environment, make sure that the Service Component Architecture (SCA) is configured. Click **Servers** → **Application servers** → *server_name* then in the **Business Integration** section, click **Service Component Architecture**. If necessary, make changes and click **Apply**. If you want to install Business Process Choreographer Observer in an ND environment, you must have already configured the Common Event Infrastructure (CEI).

For information on installing Business Process Choreographer through the WebSphere Process Server for z/OS installation and configuration scripts, including information on how to create a sample Business Process Choreographer configuration, see the WebSphere Process Server for z/OS installation and configuration PDF.

Business Process Choreographer supports enterprise applications that include business processes and human tasks. It provides a container for business processes and a container for human tasks. These containers must be installed and configured before being used. The human task container requires the business process container and the staff service. The Business Process Choreographer Explorer provides a Web client interface for human interaction and administrating business processes and human tasks. Business Process Choreographer Observer allows you to create reports on processes and tasks that have been completed. You can also use Business Process Choreographer Observer to view the status of running processes and tasks.

1. If you specified values for the Business Process Choreographer sample configuration parameters in the response file, a sample configuration that includes the business process container, the human task container and the Business Process Choreographer Explorer already exists.

You can check to see if these components are configured by looking in the administrative console for enterprise applications with names that start with BPEContainer, BPCEplorer, and TaskContainer.

Depending on the response file used, the sample configuration can be for a Cloudscape or a DB2 for z/OS database and the WebSphere default messaging provider.

Note: The sample Business Process Choreographer configuration with a Cloudscape database is not suitable for a production system. Because you can only have one Business Process Choreographer configuration, you must remove the sample configuration that utilizes Cloudscape, as described in Chapter 3, “Removing the Business Process Choreographer configuration,” on page 97 before you can continue configuring Business Process Choreographer to use WebSphere MQ or a different database.

2. Configure the resources.

You can configure the resources in one of the following ways:

- a. Manually (**recommended**), by setting properties on the administrative console or through an administrative scripting process.

If you want to use an administrative script to configure Business Process Choreographer, see “Using the bpeconfig.jacl script file to configure Business Process Choreographer” on page 13.

- b. Automatically, by using the installation wizard available on the administrative console.

If you want to use the installation wizard available on the administrative console, perform both of the following:

- “Configuring the business process container using the installation wizard” on page 25
- “Configuring the human task container, using the installation wizard” on page 46

Note: The installation wizard configures WebSphere resources only. If you choose to configure the business process container by using the installation wizard, you will still need to run the corresponding manual step(s) to create a database (Cloudscape or DB2) and to create the WebSphere MQ queues (if you are using WebSphere MQ as your Java Message Service (JMS) provider).

3. If you are using an LDAP staff plug-in, perform: “Configuring the LDAP staff plug-in provider” on page 54. The system and user registry staff plug-in providers can be used without configuring them.
4. If you have a Network Deployment (ND) environment, and you either used the human task container install wizard, or an error occurred when running the bpeconfig.jacl script, then you must setup the scheduler calendars application. Perform one of the following:
 - If you have already installed the scheduler calendars application on a server, install it on additional servers by performing the following steps:
 - a. Select **Applications** → **Enterprise Applications**.
 - b. Select **SchedulerCalendars**.
 - c. Under the Additional Properties section, select **Map modules to servers**.
 - d. Select the check box for the **Module Calendars**.
 - e. Select all the servers and the clusters on which you have configured a business process container, be sure to also select any servers or clusters where you want the SchedulerCalendars application to remain.
 - f. Select **Apply**.
 - g. Select **OK**.
 - h. Save and Synchronize changes with Nodes.
 - If this is the first time that you are installing the scheduler calendars application on a server, perform the following steps:
 - a. Select **Applications** → **Install New Application**.
 - b. In the file selector window, browse to the installableApps subdirectory of the *install_root* directory.
 - c. Select **ScheduleCalendars.ear**.
 - d. Select **Next**.
 - e. Accept the default values and select **Next** again.
 - f. Continue to accept the default values until you get to the ‘Map modules to servers’ step, then select any servers and clusters on which you want to load the ScheduleCalendars application, then select **Next**.

Using a DB2 for z/OS database

The bpeconfig.jacl script cannot create a DB2 for z/OS database. You must create it manually.

For information on creating the database and storage groups, see the WebSphere Process Server for z/OS installation and configuration guide.

Running the script in a stand-alone server environment

In a stand-alone server environment:

- Include the `-conntype NONE` option only if the application server is not running.
- If the server is running and global security is enabled, include the `-username` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

Running the script in an ND environment

In a Network Deployment environment:

- Run the bpeconfig.jacl script on the deployment manager node.
- Include the `-conntype NONE` option only if the deployment manager is not running.
- If global security is enabled, include the `-username` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

Configuring the business process container, Business Process Choreographer Explorer, and Business Process Choreographer Observer non-interactively

If you provide the necessary parameters on the command line, you will not be prompted for them. To configure Business Process Choreographer, the following command: if your current directory is `install_root/ProcessChoreographer`, enter `bin/wsadmin.sh -f ProcessChoreographer/config/bpeconfig.jacl parameters`

where *parameters* are as follows:

```
-conntype NONE
-user userName
-password userPassword
-profileName profileName
{-node nodeName -server serverName}
{-adminBFMUsers userList | -adminBFMGroups groupList}
{-monitorBFMUsers userList | -monitorBFMGroups groupList}
-jmsBFMRUNAsUser userID
-jmsBFMRUNAsPwd password
{-adminHTMUsers userList | -adminHTMGroups groupList}
{-monitorHTMUsers userList | -monitorHTMGroups groupList}
-jmsHTMRUNAsUser userID
-jmsHTMRUNAsPwd password
-contextRootBFM contextRootBFM
-contextRootHTM contextRootHTM
-mailServerName mailServerName
-mailUser mailUserID
-mailPwd mailPassword
-hostName explorerVirtualHostname
-explorerHost explorerURL
-remoteNodeName nodeName
-remoteServerName serverName
-remoteClusterName clusterName
-contextRootExplorer explorerContextRoot
```

```

-createDB { yes | no }
-dbType databaseType
-dbVersion version
-dbHome databaseInstallPath
-dbJava JDBCDriverPath
-dbName databaseName
-dbUser databaseUser
-dbPwd databasePassword
-dbAdmin databaseAdministratorUserID
-driverType JDBCDriverType
-dbTablespaceDir databaseTablespacePath
-dbServerName databaseServerName
-dbServerPort databaseServerPort
-dbStorageGroup DB2zOSSStorageGroup
-dbSubSystem DB2zOSSSubSystem
-dbSQLID DB2zOSSSchemaQualifier
-dbInstance InformixInstance
-mqType JMSProviderType
-createQM { yes | no }
-qmNameGet getQueueManagerName
-mqClusterName appServerClusterName
-qmNamePut putQueueManagerName
-mqHome MQInstallationDirectory
-mqUser JMSProviderUserID
-mqPwd JMSProviderPassword
-mqSchemaName mqSchemaName
-mqCreateTables { true | false }
-mqDataSource datasourceName
-shell shell
-createEventCollector { yes | no }
-createObserver { yes | no }

```

Note: Some of the above parameters are optional, depending on the values provided for other parameters. The dependencies between parameters and the conditions that determine whether a parameter is optional or required is described for each parameter in the descriptions below. Any required parameters that are not specified on the command line are prompted for interactively.

Parameters

You can use the following parameters when invoking the script using wsadmin:

conntype *NONE*

This specifies that no administration connection is available. Only include this option if the application server (for stand-alone) or deployment manager (for ND) is not running.

user *userName*

If global security is enabled, you must provide a user ID for authentication.

password *userPassword*

If global security is enabled, you must provide the password for the user ID *userName*.

profileName *profileName*

Where *profileName* is the name of a user-defined profile. On z/OS the profile name is default.

node *nodeName*

Where *nodeName* is the name of the node where Business Process Choreographer is to be configured. If you have only one node and exactly one server, this parameter is optional.

server *serverName*

Where *serverName* is the name of the server where Business Process Choreographer is to be configured. If you have only one node and exactly one server, this parameter is optional.

adminBFMUsers *userList*

Where *userList* is the list of names of users, from the user registry, to which to map the BPESystemAdministrator Java 2 Enterprise Edition (J2EE) role. The separator character is the vertical line (|). This property is needed to install the business process container. This parameter has no default value. Either one or both of the adminBFMUsers or adminBFMGroups options must be set.

adminBFMGroups *groupList*

Where *groupList* is the list of names of groups, from the user registry, to which to map the BPESystemAdministrator J2EE role. The separator character is the vertical line (|). This property is needed to install the business process container. This parameter has no default value. Either one or both of the adminBFMUsers or adminBFMGroups options must be set.

monitorBFMUsers *userList*

Where *userList* is the list of names of users, from the user registry, to which to map the BPESystemMonitor J2EE role. The separator character is the vertical line (|). This property is needed to install the business process container. This parameter has no default value. Either or both monitorBFMUsers or monitorBFMGroups must be set.

monitorBFMGroups *groupList*

Where *groupList* is the list of names of groups, from the user registry, to which to map the BPESystemMonitor J2EE role. The separator character is the vertical line (|). This property is needed to install the business process container. This parameter has no default value. Either or both monitorBFMUsers or monitorBFMGroups must be set.

jmsBFMRunAsUser *userID*

Where *userID* is the run-as user ID from the user registry for the business process container JMS API. This property is needed to install the business process container. This parameter has no default value. It must be set.

jmsBFMRunAsPwd *password*

Where *password* is the password for the business process container JMS API. This property is needed to install the business process container. This parameter has no default value. It must be set.

adminHTMUsers *userList*

Where *userList* is the list of names of users, from the user registry, to which to map the TaskSystemAdministrator Java 2 Enterprise Edition (J2EE) role. The separator character is the vertical line (|). This property is needed to install the task container. This parameter has no default value. Either one or both of the adminHTMUsers or adminHTMGroups options must be set.

adminHTMGroups *groupList*

Where *groupList* is the list of names of groups, from the user registry, to which to map the TaskSystemAdministrator J2EE role. The separator character is the vertical line (|). This property is needed to install the task container. This parameter has no default value. Either one or both of the adminHTMUsers or adminHTMGroups options must be set.

monitorHTMUsers *userList*

Where *userList* is the list of names of users, from the user registry, to which to map the TaskSystemMonitor J2EE role. The separator character is the vertical

line (1). This property is needed to install the task container. This parameter has no default value. Either or both `monitorHTMUsers` or `monitorHTMGroups` must be set.

monitorHTMGroups *groupList*

Where *groupList* is the list of names of groups, from the user registry, to which to map the `TaskSystemMonitor` J2EE role. The separator character is the vertical line (1). This property is needed to install the task container. This parameter has no default value. Either or both `monitorHTMUsers` or `monitorHTMGroups` must be set.

jmsHTMRunAsUser *userID*

Where *userID* is the run-as user ID from the user registry for the human task container JMS API. This property is needed to install the human task container. This parameter has no default value. It must be set.

jmsHTMRunAsPwd *password*

Where *password* is the password for the human task container JMS API. This property is needed to install the human task container. This parameter has no default value. It must be set.

contextRootBFM *contextRootBFM*

Where *contextRootBFM* is the context root for the Web Service Endpoint URL. For a Business Flow Manager (BFM), on a server, the default context root is `/BFMIF_{nodeName}_{serverName}`. On a cluster, the default is `/BFMIF_{clusterName}`. It must be set.

contextRootHTM *contextRootHTM*

Where *contextRootHTM* is the context root for the Web Service Endpoint URL. For a Human Task Manager (HTM), on a server, the default context root is `/HTMIF_{nodeName}_{serverName}`. On a cluster, the default is `/HTMIF_{clusterName}`. It must be set.

mailServerName *mailServerName*

Where *mailServerName* is the host name of the mail server to be used by the Human Task Manager to send notification mails. It is needed when configuring the mail session. If this parameter is not set, the mail session configuration will be skipped. The default value is the fully qualified host name of the local host.

mailUser *mailUserID*

Where *mailUserID* is the user ID to access the mail server. It is needed to create the mail session for the Human Task Manager to send notification mails. The default value is empty, which is only appropriate if no authentication is required.

mailPwd *mailPassword*

Where *mailPassword* is the password to access the mail server. It is needed to create the mail session for the Human Task Manager to send notification mails.

hostName *explorerVirtualHostname*

Where *explorerVirtualHostname* is the virtual host where the Business Process Choreographer Explorer will run. The default value is `default_host`.

explorerHost *explorerURL*

Where *explorerURL* is the URL of the Business Process Choreographer Explorer. If this parameter is not specified for a non-cluster environments, a default value is computed, for example, `http://localhost:9080`. The value of this parameter is used for the `EscalationMail.ClientDetailURL` custom property of the Human Task Manager.

precompileJSPs { yes | no }

Determines whether Java Server Pages (JSPs) will be precompiled, or not.

remoteNodeName *nodeName*

Use this parameter and `remoteServerName` if you do not want to connect to the local Business Process Choreographer Explorer. Do not specify this parameter if you want to connect to the Business Process Choreographer server identified by the node and server parameters or the cluster parameter.

remoteServerName *serverName*

Use this parameter and `remoteNodeName` if you do not want to connect to the local Business Process Choreographer Explorer. Do not specify this parameter if you want to connect to the Business Process Choreographer server identified by the node and server parameters or the cluster parameter.

remoteClusterName *clusterName*

Use this parameter, if you do not want to connect to the local Business Process Choreographer Explorer and you do not specify `remoteNodeName` and `remoteServerName`. Do not specify this parameter if you want to connect to the Business Process Choreographer server identified by the node and server parameters or the cluster parameter.

contextRootExplorer *contextRootExplorer*

Where *contextRootExplorer* is the context root for the Business Process Choreographer Explorer. The default value is `/bpc`, which results in the default URL of `http://host:port/contextRootExplorer`. The context root must be unique within the WebSphere cell.

createDB { yes | no }

Possible values are `yes` or `no`. If set to `yes`, the script will create the database. For z/OS databases, this script cannot create the database, it can only create the table spaces and tables. For other database types, the default value is `yes`.

For information on how to create the database and storage groups for DB2 for z/OS, see *Creating databases and storage groups* in the section titled *Planning your configuration*.

dbType *databaseType*

Where *databaseType* is the database type. This is needed for installing the business process container, for creating the database or database tables, and for creating the data source. There is no default value. Possible values are:

- Cloudscape
- zOS-DB2

dbVersion *DB2zOSversion*

Where *DB2zOSversion* is either the value 7 or 8. This parameter is only required when the database type is DB2 for z/OS. It has no default value.

dbHome *databaseInstallPath*

Where *databaseInstallPath* is the installation directory of the database system. This value will vary depending on version of DB2 being used and where DB2 is installed.

dbJava *JDBCdriverPath*

Where *JDBCdriverPath* is the directory where the JDBC driver is located. This parameter is only required for the following combinations of database and driver types:

- DB2 or DB2 for z/OS, with a type 4 driver. The default value is `${dbHome}/java`.

dbName *databaseName*

Where *databaseName* is the name of the Business Process Choreographer database. It is used to create the database or the database tables, and for creating the data source. The default value is BPEDB..

dbUser *databaseUser*

Where *databaseUser* is the user ID to access the database. It is used to create the database tables and the data source. The default value depends on the database and platform. For DB2 for z/OS the default value is db2inst1.

dbPwd *databasePassword*

Where *databasePassword* is the password for the user ID *databaseUser*.

dbAdmin *databaseAdministratorUserID*

Where *databaseAdministratorUserID* is the user ID of the database administrator. It is only required when creating the database and database tables for the following database types:

- For DB2 for z/OS, the default is db2inst1.

driverType *JDBCdriverType*

Where *JDBCdriverType* is the type of JDBC driver. It is used to create the data source.

- For DB2, possible values are Universal or CLI. It is also used for creating the database tables.

dbTablespaceDir *databaseTablespacePath*

Where *databaseTablespacePath* is the directory where the database table spaces are created. It is used to create the database and database tables. This parameter is only required for the following database types:

- For DB2, the default value is empty, which means that no table spaces are created.

dbServerName *databaseServerName*

Where *databaseServerName* is the host name server that hosts the database for Business Process Choreographer. database. It is used to create the data source. For Sybase it is also used to create the database.

- For DB2, the default value is empty.
- For all other database types, the default value is the fully qualified host name of the local host.

dbServerPort *databaseServerPort*

Where *databaseServerPort* is the TCP/IP port for the database server for Business Process Choreographer. This parameter is required if *dbServerName* is specified. For DB2, the default value is 50000.

dbStorageGroup *DB2zOSSStorageGroup*

Where *DB2zOSSStorageGroup* is the storage group used to create the Business Process Choreographer database table . This parameter is only required for DB2 on z/OS. There is no default value, and must not be empty. See *Creating the database and storage groups* for more information/

dbSubSystem *DB2zOSSubSystem*

Where *DB2zOSSubSystem* is the DB2 sub system used to create the Business Process Choreographer database table and the data source. This parameter is only required for DB2 on z/OS. The default value is BPEDB.

dbSQLID *DB2zOSSchemaQualifier*

Where *DB2zOSSchemaQualifier* is the schema qualifier used to create the

database tables. This parameter is only required for DB2 on z/OS. There is no default value. The value can be empty. Only specify a value when using the Universal JDBC driver type.

mqType *JMSProviderType*

Where *JMSProviderType* is the type of Java Message Service (JMS) provider to use for Business Process Choreographer. It is used to create the queue manager and the queues, the listener ports or ActivationSpecs, and the queue connection factories.

Where *JMSProviderType* is one of the following values:

WPM For default messaging (WebSphere Platform Messaging). This option is always available.

MQSeries

For WebSphere MQ. This option requires that the product WebSphere MQ is installed.

The `bpeconfig.sh` utility will run `createQueues.sh` when you specify `MQSeries` as your `mqType`.

As a result of running `createQueues.sh` the following file is created: `/tmp/tmp_crt_ques.mqs`. This file contains the WebSphere MQ definitions that need to be provided to the WebSphere MQ administrator. He or she can add the contents of this file to their current jobs for configuring WebSphere MQ on z/OS.

createQM { *yes* | *no* }

Controls whether the script creates a local WebSphere MQ queue manager. This option only has an effect if the parameter `mqType` has the value `MQSeries`. The default value for this parameter is `yes`. Use the value `no` if you do not want the script to create the WebSphere MQ queue manager, for example, if you want to create the queue manager on a different server to the one where you are running the script.

qmNameGet *getQueueManagerName*

Where *getQueueManagerName* is the name of the queue manager for GET requests. It is used to create the queue manager and the queues, and to create the listener ports and the queue connection factories. It must not contain the `-` character. The default value for *getQueueManagerName* is `BPC_nodeName_serverName`. This option only has an effect if the parameter `mqType` has the value `MQSeries`.

mqClusterName *appServerClusterName*

Where *appServerClusterName* is the name of the WebSphere Application Server cluster where the default JMS provider's message engines are to be created. This has nothing to do with a WebSphere MQ cluster. This option is only used when configuring Business Process Choreographer in a cluster and the `mqType` option is set to `WPM`.

qmNamePut *putQueueManagerName*

Where *putQueueManagerName* is the queue manager name for PUT requests. It is used only when the `mqClusterName` parameter has been set. It is used to create the queue manager and the queues, and to create the listener ports and the queue connection factories. It must not contain the `-` character, and it must not be the same as the queue manager name specified for the `qmNameGet` parameter. The default value for *putQueueManagerName* is `BPCC_nodeName_serverName`.

mqHome *MQInstallationDirectory*

Where *MQInstallationDirectory* is the installation directory of WebSphere MQ. This is used to create the queue manager and the queues (Windows systems only) and for creating the listener ports and the queue connection factories. If the WebSphere variable `MQ_INSTALL_ROOT` is set, its value is used, and is not modified. This option only has an effect if the parameter `mqType` has the value `MQSeries`.

If `MQ_INSTALL_ROOT` is not set, the default value used for *MQInstallationDirectory* depends on the platform:

AIX: /usr/mqm

Solaris and HP-UX:
/opt/mqm

mqUser *JMSProviderUserID*

Where *JMSProviderUserID* is the user ID to access the JMS provider.

- If `mqType` has the value `WPM`, this parameter is used to create the `ActivationSpecs` and the connection factories; the default value is the currently logged on user.
- If `mqType` has the value `MQSeries`, this parameter is used on non-Windows platforms to create the queue manager and the queues. The default value for *JMSProviderUserID* is :

mqm

mqPwd *JMSProviderPassword*

Where *JMSProviderPassword* is the password for the user ID provided for `mqUser`. This parameter has no default value.

mqSchemaName *mqSchemaName*

Where *mqSchemaName* is the name of the database schema for the default JMS provider's messaging engine. The default value is `BPEME`. This option is only used when configuring Business Process Choreographer in a cluster and the `mqType` option is set to `WPM`.

mqCreateTables { *true* | *false* }

This Boolean parameter controls whether the default JMS provider automatically creates its tables in the message engine database upon the first connection. The default value is `true`. This option is only used when configuring Business Process Choreographer in a cluster and the `mqType` option is set to `WPM`.

mqDataSource *datasourceName*

Where *datasourceName* is the JNDI name of the data source to be used by the default JMS provider's message engine. This must be a cluster-level data source in the WebSphere cluster identified by *mqClusterName*. The underlying database for the default JMS provider must be created manually. This option is only used when configuring Business Process Choreographer in a cluster and the `mqType` option is set to `WPM`.

shell *shell*

This parameter determines the shell that is used to run external commands. The default value is `/bin/sh`.

createEventCollector { *yes* | *no* }

When run in batch mode, the default is `yes`, which causes the Business Process Choreographer event collector application to be configured, which is required by Business Process Choreographer Observer. If you do not want it installed, set the value of this parameter to `no`.

createObserver { *yes* | *no* }

When run in batch mode, the default is yes, which causes the Business Process Choreographer Observer application to be configured. If you do not want it installed, set the value of this parameter to no.

Running the configuration script interactively

This example, illustrates running the `bpeconfig.jacl` script to install and configure a business process container that uses an existing DB2 database, a human task container, and a Business Process Choreographer Explorer.

Restriction: When run interactively, this script cannot configure Business Process Choreographer Observer, nor the necessary event collector application. If you want to use Business Process Choreographer Observer, you must perform “Configuring the Business Process Choreographer Observer infrastructure” on page 84.

1. On the server, or for ND, on the deployment manager, start the script by entering the command:

```
install_root/bin/wsadmin.sh
-f install_root/ProcessChoreographer/sample/bpeconfig.jacl
```

2. Interactively enter responses to the questions that are displayed:
 - a. In an ND environment, you will be offered a cluster to configure in. If it is not the correct cluster, enter **No** to be offered the next cluster. If it is the correct cluster, enter **Yes**.
 - b. For the question Install the business process container?, enter **Yes**.
 - c. For the question User(s) to add to role BPSystemAdministrator, enter the user IDs for the users who will perform the role of business process administrator.
 - d. For the question Group(s) to add to role BPSystemAdministrator, enter the groups from the domain user registry that are mapped onto the role of business process administrator.
 - e. For the question User(s) to add to role BPSystemMonitor, enter the user IDs for the users who will perform the role of business process monitor.
 - f. For the question Group(s) to add to role BPSystemMonitor, enter the groups from the domain user registry that are mapped onto the role of business process monitor.
 - g. If you get the question Use WebSphere default messaging or WebSphere MQ, enter one of the two displayed options.
 - h. For the question Run-as UserId for role JMSAPIUser, enter the run-as user ID that will be used for the JMSAPIUser role.
 - i. Enter the password for the run-as user ID.
 - j. For the question Use a DB2, an Informix, an Oracle, or an SQL Server database [DB2/Informix/Oracle/MSSQL]?, for this example, enter DB2. Selecting a different database results in other database-specific questions.
 - k. For the question Use WebSphere default messaging or WebSphere MQ [WPM/MQSeries]?, select the JMS provider that you want to use.
 - l. If you selected WebSphere Platform Messaging (WPM), also enter the following:
 - 1) For the question Virtual Host for the SCA Web Service [default_host]: , press **Enter** to accept the default value default_host.

- 2) For the question Context root for the SCA Web Service [/BFMIF_PNODE_server1]:, press **Enter** to accept the default value /BFMIF_PNODE_server1.
- m. For the question Create the DataSource for the Process Choreographer database?, enter **Yes**.
- n. Enter the database name.
- o. For the question Universal or CLI?, enter the type of the JDBC driver.
- p. For the question DB2 User ID, enter the user ID used to create the database tables and schema.
- q. For the question Database server name (may be empty, set to use the type 4 driver), enter the name of the server that hosts the database.
- r. For the question Database server port, enter the database server port, for example, 50000.
- s. For the question Create the Process Choreographer database?, if your user ID has sufficient authority to create the database you can enter **Yes**, otherwise, if the database already exists, or if your user ID does not have sufficient authority to create the database, enter **No**.
- t. For the question DB2 tablespace directory (may be empty) enter the directory for the table space, or leave it empty.
- u. For the question Create the ActivationSpecs for the business flow manager?, enter **Yes** or **No**.
- v. If you get the question User ID for access to default messaging, enter the user ID to use to access the default JMS provider.
- w. If you get the question Name of the message engine cluster, enter the name of the message engine cluster.
- x. If you get the question Name of the message engine database schema, enter the name of the message engine database schema.
- y. If you get the question Automatically create the message engine database tables [true/false]?, enter true to automatically create the message engine database tables, otherwise enter false.
- z. If you get the question Message engine datasource JNDI name, enter the JNDI name of the message engine data source.
- aa. For the question Install the task container?, enter **Yes**.
- ab. For the question User(s) to add to role TaskSystemAdministrator, enter the user IDs for the users who will perform the role of task administrator.
- ac. For the question Group(s) to add to role TaskSystemAdministrator, enter the groups from the domain user registry that are mapped onto the role of task administrator.
- ad. For the question User(s) to add to role TaskSystemMonitor, enter the user IDs for the users who will perform the role of task monitor.
- ae. For the question Run-as UserID for role EscalationUser, enter the run-as user ID for the role of escalation user, for example db2admin.
- af. For the question Context root for the SCA Web Service [/HTMIF_PNODE_server1]:, enter the context root for the Service Component Architecture (SCA) Web server, or press **Enter** to accept the default value.
- ag. For the question Create the mail notification session for the human task manager?, enter **No** if you do not want to create the mail notification session for the Human Task Manager. Otherwise, enter **Yes**, and specify the mail transport host and user ID.

- ah. For the question Create the ActivationSpecs for the human task manager?, enter **Yes** to create J2EE ActivationSpecs for the Human Task Manager Message Driven Bean (MDB), otherwise enter **No**.
 - ai. If you get the question Configure in cluster 'MECluster' [Yes/no]?, enter **Yes** to configure in the specified cluster, otherwise, enter **No**.
 - aj. If you get the question Add JDBC provider permissions to server.policy [Yes/no]?, enter **Yes** to automatically add the permissions for the JDBC provider to the server.policy file, otherwise, enter **No**.
 - ak. For the question Install the Business Process Choreographer Explorer?, enter **Yes** to install Business Process Choreographer Explorer, then for the Virtual host for the Business Process Choreographer Explorer, enter the name of the virtual host for Business Process Choreographer Explorer, for example, **default_host**, then for the question Precompile JSPs?, enter **Yes** if you want Java Server Pages (JSPs) to be precompiled, otherwise enter **No**. For a remote Business Process Choreographer Explorer, for the question Node of Process Choreographer to connect to [PNODE]: enter the name of the Business Process Choreographer node to connect to, and for the question Server of Process Choreographer to connect to [server1]: enter the name of the Business Process Choreographer server to connect to or press **Enter** to accept the default.
 - al. If you get the question Context root for the Business Process Choreographer Explorer [/bpc]: , enter the context root for Business Process Choreographer Explorer or press **Enter** to use the default value /bpc.
 - am. For the question Create aliases for *your_server* in host *your_host*?, enter **Yes** to create aliases for your server in the your virtual host, otherwise enter **No**.
 - an. Various information is displayed, for example providing the URL of the Business Process Choreographer Explorer and reminders where to find the script files that you can use to configure Business Process Choreographer Observer.
 - ao. For the question Enable global security using the Local OS user registry?, enter **Yes** to enable global security using the local operating system user registry, otherwise, enter **No**.
 - ap. For the question Server user ID, enter the server user ID.
 - aq. For the question Enforce Java 2 security?, enter **Yes** to enforce Java 2 security, otherwise, enter **No**.
 - ar. For the question Set 'com.ibm.SOAP.loginuserid' in soap.client.props?, enter **Yes** to set the login user ID in the SOAP client properties, otherwise, enter **No**.
 - as. For the question Delete the temporary directory?, enter **Yes** to delete the temporary directory specified, otherwise, enter **No**.
3. In case of problems, check the log files.

Log files

If you have problems creating the configuration using the bpeconfig.jacl script file, check the following log files:

- bpeconfig.log
- wsadmin.traceout

Both files can be found in the logs directory for your profile:

- In the directory *install_root/profiles/profileName/logs/*.

Configuring the business process container using the installation wizard

This describes how to create the necessary resources and then run the business process container installation wizard.

You must configure the necessary resources and install the business process container application before you can run applications that contain business processes or human tasks. The recommended method for configuring the necessary resources is by running `bpeconfig.jacl`.

Note: The installation wizard configures WebSphere resources only. If you choose to configure the business process container by using the installation wizard, you will still need to run the corresponding manual step(s) to create a database (Cloudscape or DB2 for z/OS) and to create the WebSphere MQ queues (if you are using WebSphere MQ as your Java Message Service (JMS) provider).

1. If you are preparing a clustered Business Process Choreographer setup:
 - a. Create the cluster: Perform "Creating a clustered environment" in the PDF for Administering.
 - b. If you are using the default JMS messaging provider for your cluster:
 - 1) Make sure that the cluster supports service applications as described in "Preparing a server or cluster to support Service Component Architecture applications" in the PDF for Administering.
 - 2) Create the database for the message engine's data store. You can either use the same database that is used for the Service Component Architecture (SCA) message engines or a separate database. It is recommended to use one messaging database for all buses that are created by WebSphere Process Server, that is, for the SCA system bus, the SCA application bus, the Common Event Infrastructure bus, and the Business Process Choreographer bus. The database should be accessible to all members of the cluster that hosts the message engine to ensure failover availability of the message engine.
 - If the JMS user is authorized to create tables, the default message engine creates the necessary tables in the database the first time it is accessed access.
 - If the JMS user is not authorized to create tables, create the tables before the default messaging provider tries to access the database. You can use the `sibDDLGenerator` utility that is in the `bin` subdirectory of your *install_root* directory to generate a DDL file that can be used to create the tables.
2. Create the database for Business Process Choreographer: Perform "Creating the database for the business process container" on page 30.
3. Make sure that the server is started and that you are logged on to the administrative console with a user ID with sufficient administration rights.
4. In the administrative console, select the server or cluster where you want to install the business process container. Click one of the following:
 - **Servers** → **Application Servers** → *serverName*
 - **Servers** → **Clusters** → *clusterName*

Where *serverName* or *clusterName* is the name of the application server or cluster where you want to install the business process container. In a cluster, you can select any application server, and the business process container is installed simultaneously on all application servers in the cluster.

5. Go to the Business Process Container settings. On the **Configuration** tab, under **Container Settings**, expand **Business process container settings**, and click **Business process container**.
6. Verify that the business process container is not installed. There should be a message indicating that the business process container is not currently installed. If the business process container is already installed, perform Chapter 3, “Removing the Business Process Choreographer configuration,” on page 97 before starting the installation wizard.
7. Start the installation wizard. In the **Additional Properties** section, click the link **Business process container installation wizard**.
8. Select the database configuration (wizard step 1):
 - a. In the **JDBC Providers** drop-down list, select the entry with the database system, system version and Java Database Connectivity (JDBC) driver that you are using. Where possible, the installation 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 in “Business process container installation wizard settings” on page 34.
 - b. For the **Implementation class name** use the default class name that is provided for the JDBC driver implementation.
 - c. For **Classpath** enter the location of the Java archive (JAR) or the compressed file that contains the JDBC driver. To use the path variable that is displayed in the text field, it must be defined in **Environment** → **Manage WebSphere Variables**.
 - d. The **Data source user name** must be a user ID that has the authority to connect to the database and to modify the data. If the user ID has the authority to create tables and indexes in the database, then the database schema will be updated automatically, when necessary, after applying a service or fix pack. This is not required for a Cloudscape database.
 - e. Enter the **Data source password** for the data source user name. This is not required for a Cloudscape database.
 - f. The **Custom properties** field contains default values for the database that you selected.
 - If you are using a Cloudscape database that is not in the default directory, change the value for the custom property **databaseName** to specify the fully qualified location of the database.
 - You might need to change or add some other properties. For more information, see the Business process container installation wizard settings page and the product documentation for your database system.
 - g. Click **Next** to go to the next step in the installation wizard.
9. Select the JMS provider and security (wizard step 2):
 - a. In the drop-down list for **JMS provider**, select the messaging service for the business process container to use.
 - For default messaging, select Default messaging provider.
 - For WebSphere MQ, select WebSphere MQ.
 - b. Use the default value for **Queue Manager** (**BPC_nodeName_serverName**). If you are using the default messaging provider, this field is ignored.

- c. If you are using the WebSphere MQ JMS provider and the WebSphere environment variable `{MQ_INSTALL_ROOT}` is not defined, make sure that the **Classpath** points to the WebSphere MQ Java lib directory. By default, `MQ_INSTALL_ROOT` is predefined with the value `{WAS_INSTALL_ROOT}/lib/WMQ`.
 - d. For the **JMS user ID**, enter a user ID that has administration rights for the messaging service. Use root
 - e. For the **JMS password**, enter the password for the JMS user ID.
 - f. For the **Webservices Endpoint**, enter the Webservice endpoint for the Webservice API.
 - g. For the **JMS API User ID**, enter a user ID from the user registry. This user ID will be used to process asynchronous API calls.
 - h. For the **JMS API Password**, enter the password for the JMS API User ID.
 - i. For the **Administrator security role mapping**, enter the name of the group, defined in the user registry, that will map onto the role of Business Process Administrator.
 - j. For the **System monitor security role mapping**, enter the name of the group in the user registry to map onto the role of Business Process System Monitor.
 - k. Click **Next** to go to the next step in the installation wizard.
10. Select the JMS Resources and Business Process Choreographer Explorer (wizard step 3): Either select **Create new JMS resources using default values**, 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**.
 11. **Optional:** To install **Business Process Choreographer Explorer**, select the check box; otherwise, clear the check box. You can optionally specify the context root. If you want to have more than one Business Process Choreographer Explorer installed on the same server, at most one of them can use the default context root `/bpc`.
 12. **Optional:** To enable the audit log, select **Enable audit logging for all processes running in this container**.
 13. **Optional:** To use the Common Event Infrastructure, select **Enable Common Event Infrastructure logging for all processes running in this container**.
 14. **Optional:** To install **Business Process Choreographer Observer** select the check box; otherwise, clear the check box. If you cannot select the check box, make sure that the check box for **Enable Common Event Infrastructure logging for all processes running in this container** is selected. For **JMS User ID**, enter a user ID from the user registry that can be used to connect to the CEI bus. For **JMS password**, enter the associated password.
 15. Click **Next** to view the summary (wizard step 4).
 16. Check that the information on the summary page is correct. 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 helps 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**. The progress is shown on the **Installing** page.
17. If the installation did not succeed, check for any error messages that can help you correct the problem, then try again.
 18. If the installation succeeded, click **Save Master Configuration**, then click **Save**.
 19. If you configured Business Process Choreographer in a cluster and you are using the WebSphere MQ JMS provider: Perform “Customizing the WebSphere MQ JMS resources in a cluster” on page 44.
 20. Restart the application server.
 21. Verify that the business process container has started successfully: In the administrative console, select **Applications** → **Enterprise Applications** verify that the status of the application named `BPEContainer_scope` is started. If you installed the business process container on an application server, `scope` is `nodeName_serverName`. If you installed the business process container on a cluster `scope` is the cluster name.

If you also installed the Business Process Choreographer Explorer you should also see an application running that is named `BPCExplorer_scope` if it uses the default context root, `/bpc` or `BPCExplorer_scope_contextroot` if it does not use the default context root.

If you installed the Business Process Choreographer Observer, you should also see two applications running that are name `BPCObserver_scope` and `BPCCollector_scope`

The business process container is configured.

Continue configuring at step Configuring the human task container, using the installation wizard .

Creating the queue manager and queues for the business process container

This topic describes how to create the WebSphere MQ queue manager and queues on z/OS.

WebSphere MQ must already be installed.

Scripting is not available for creating queue managers on z/OS. Consult your MQ administrator for information on creating queue manager (QMGR) and listeners on z/OS.

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

1. Create the queue definitions. Type the following:

```
cd install_root/ProcessChoreographer createQueues.sh z/OS
```

This command creates the file `/tmp/tmp_crt_ques.mqs`.

2. Tailor `/tmp/tmp_crt_ques.mqs`

Edit the contents of `/tmp/tmp_crt_ques.mqs` to meet your site-specific standards as appropriate.

The file `/tmp/tmp_crt_ques.mqs` contains your site-specific standards as it pertains to definitions for WebSphere MQ queues and queue manager.

The MQ administrator can use `tmp_crt_ques.mqs` to configure the WebSphere MQ resources using the WebSphere MQ z/OS Commands in JCL scripts.

Continue configuring at step Create the database for Business Process Choreographer.

Creating clustered queue managers and queues for the business process container

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

Note: Running `createQueues.sh` produces sample WebSphere MQ definitions only. Consult with your WebSphere MQ administrator on using the sample definitions.

1. 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.

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

See Using the `bpeconfig.jacl` script file to configure Business Process Choreographer for information on parameters required to create clustered queue managers and queues for the business process container.

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

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 server, 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 TCP/IP') +
  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 TCP/IP') +
  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.
2. To verify the status of the channels on a server, enter the MQ command:

```
display chstatus(*)
```

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

Creating the database for the business process container

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

In a clustered Business 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. On the server that hosts the database, create the database according to the description for your database system.
 - “Creating a Cloudscape database for Business Process Choreographer” on page 31.
 - “Creating a DB2 for z/OS database for Business Process Choreographer” on page 32.
2. On each server that runs Business Process Choreographer without a local database, you must make the remote database accessible:
 - a. Install a suitable database client or Java Database Connectivity (JDBC) driver on the server that hosts the application server.
 - b. If you are not using a type-4 JDBC driver, make the new database known to the database client as follows:

For Cloudscape

No action is required, because Business Process Choreographer supports only the embedded version of Cloudscape, which does not

support remote access. The 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.

- c. Test the connection to the database.
 - 1) Click **Resources** → **JDBC Providers**
 - 2) If necessary, select a different scope and click **Apply**. For clustered Business Process Choreographer configurations, the data source can be defined at the cluster level or server level. For non-clustered configurations, the data source is defined at the server level.
 - 3) Click *provider_name* → **Data sources**
 - 4) Locate the appropriate data source, `BPEDataSourcedatabase`. For example, `BPEDataSourceCloudscape`.
 - 5) Select the check box for the data source, and click **Test connection**.
 - 6) You should see a message indicating that the test connection was successful.

The Business Process Choreographer database exists and is accessible from the servers that host the application server and the deployment manager.

Continue configuring at step 3 on page 25.

Creating a Cloudscape database for Business Process Choreographer

Use this task to create a Cloudscape database for Business Process Choreographer.

The Cloudscape database system is implemented in the Java language. It comes with the WebSphere Process Server as several Java Archive (JAR) files.

The Cloudscape license that comes with WebSphere Process Server is only for development and test, not for production purposes. Cloudscape cannot be used as database system for Business Process Choreographer in a Network Deployment environment. The Cloudscape version that comes with this product includes the Cloudscape Network Server that supports client/server JDBC access over the Distributed Relational Database Architecture™ (DRDA®) protocol. Because the version of Cloudscape Network Server that is provided with this version of WebSphere Process Server has no XA support, Business Process Choreographer can only use the Embedded Cloudscape version that cannot be accessed remotely.

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

1. Prepare to run the database creation script file by performing one of the following:
 - To prepare to create the database in the default location, manually create a databases subdirectory in the appropriate profile directory. Create `install_root/profiles/Profile_name/databases`. Change to the new directory.
 - To prepare to create a database location other than the default location, change to the directory where you want the new database created. If you run the business process container installation wizard, you must remember to specify the fully qualified database location as the value of the custom property `databaseName`.

2. Copy the database creation script to the current directory. Copy the file `install_root/ProcessChoreographer/Cloudscape/createDatabase.sql`
3. Check whether you have Java configured on your server. Enter the command:


```
java -version
```

If you get an error message, then in step 5, when you run the database creation script, you must prefix the Java command with the full path to the Java executable, add the path `install_root/java/bin/`
4. Read the instructions in the header of the database creation script, `createDatabase.sql`, in an editor. The sample files are delivered in ASCII format. Depending on the capabilities of the tool you use to view, edit and run this file, you may need to convert the file to a readable format, EBCDIC for example. For example, you can edit the directly in ASCII using `viascii` (`viascii createDatabase.sql`. and then Use `iconv` to convert the file to EBCDIC so you can use `vi`:


```
iconv -t IBM-1047 -f ISO8859-1 createDatabase.sql >
createDatabase_EBCDIC.sql
```
5. Run the database creation script file. Type the following:


```
java -Djava.ext.dirs=install_root/cloudscape/lib
-Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
install_root/ProcessChoreographer/Cloudscape/createDatabase.sql
```
6. If you also want Business Process Choreographer Observer to use this database, perform the following:
 - a. Copy the following SQL scripts to your database server:


```
clearSchema_Observer.sql
createDatabase_Observer.sql
createSchema_Observer.sql
dropSchema_Observer.sql
```

 - The SQL files are located in `install_root/dbscripts/ProcessChoreographer/Cloudscape/`.
 - b. In a text editor, read the instructions in the header of the script file `createSchema_Observer.sql`. Avoid using the Notepad editor, because it does not display the file in a readable format.
 - c. Create the schema. From the directory where you created the database, run the script file `createSchema_Observer.sql` as described in the header of the script. The sample files are delivered in ASCII format. Depending on the capabilities of the tool you use to view, edit and run this file, you may need to convert the file to a readable format, EBCDIC for example. For example, you can edit the directly in ASCII using `viascii` (`viascii createSchema_Observer.sql`. and then Use `iconv` to convert the file to EBCDIC.
 - d. In case of errors, you can run the script file `dropSchema_Observer.sql` to drop the schema.

The database for Business Process Choreographer exists.

Continue configuring at step 2.

Creating a DB2 for z/OS database for Business Process Choreographer

Use this task to create a DB2 for z/OS database for Business Process Choreographer.

See DB2 Universal JDBC Driver Support for information on support for DB2 Universal JDBC Driver in WebSphere Process Server for z/OS.

When using DB2 for z/OS, the following updates may be required:

- DB2 configuration parameters (zParms) need to be increased to support Business Process Choreographer LOBs.
 - `_LOBVALA`
 - `_LOBVALS`
- Required DB2 Conversion services:
 - `CONVERSION 367,1208,ER;`
 - `CONVERSION 1208,367,ER;`

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

1. You must have already installed WebSphere Process Server on a z/OS server.
2. On the z/OS server 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 server running WebSphere Process Server. Normally, for security reasons, this user ID is not the one that 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.
 - h. Decide if you want to create the tables and views in the schema of the connected user ID or if you want to customize the schema qualifier (`_SQLID`). If a single user ID accesses multiple databases with tables of the same name, you must use different schema qualifiers to avoid name collisions.
3. On the server that hosts the WebSphere Process Server:
 - a. Take note of the following information:

An important difference exists between DB2 for z/OS and DB2 for Linux, UNIX, and Windows. DB2 for Linux, UNIX, and Windows does not have the concept of a subsystem, but DB2 for z/OS does. To avoid confusion between database name and subsystem name, it is important to understand that because DB2 for z/OS runs in a subsystem, the catalog node and catalog database commands must identify the appropriate subsystem. On DB2 for Linux, UNIX, and Windows, the subsystem name is not a known concept, so the database name that the catalog command makes a link to is really the name of the DB2 for z/OS subsystem.
 - b. On the server that hosts your application server, change to the directory where the Business Process Choreographer configuration scripts for your database system are located:
 - On Windows systems, depending on your DB2 version, enter one of the following commands:

```
cd install_root\dbscripts\ProcessChoreographer\DB2z0SV7
cd install_root\dbscripts\ProcessChoreographer\DB2z0SV8
```
 - On UNIX and Linux[®] systems, depending on your DB2 version, enter one of the following commands:

```
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV7
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
```

- c. Edit the createTablespace.sql script. Replace @STOGRP@ with the storage group name and replace @DBNAME@ with the database name (not the subsystem name).
- d. Run your customized version of the createTablespace.sql script, as described in the header of the script. If you want to drop the table space, use the dropTablespace.sql script.
- e. Edit the createSchema.sql script.
 - 1) Replace @STOGRP@ with the storage group name.
 - 2) Replace @DBNAME@ with the database name (not the subsystem name).
 - 3) Replace @_SQLID@ with the schema qualifier or remove @_SQLID@ (including the following dot) from the script. A custom schema qualifier can only be used with the DB2 Universal JDBC driver and requires that the configuration customSQLID property is set to the appropriate value.
- f. Run your customized version of the createSchema.sql 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 the dropSchema.sql script to drop the schema, but replace @_SQLID@ before running the script.

The database for Business Process Choreographer exists.

Note: The SQL definitions are provided, you must add them to your DB2 environment manually.

Business process container installation wizard settings

Use the installation wizard to install and configure the business process container.

Access the business process container installation wizard by clicking **Servers** → **Application servers** → *server_name* → **Business Process Container Settings** → **Business process container** → **Business process container installation wizard**. This page describes the installation wizard fields, in the order that they display in the wizard.

Step 1 database configuration:

- JDBC provider
- Implementation class name
- Class path (for JDBC provider)
- Data source user name
- Data source password
- Custom Properties

Step 2 JMS provider and security:

- JMS provider
- Queue manager
- Class path (for the JMS provider)
- JMS user ID (for the JMS provider)
- JMS password (for the JMS provider)
- Context root for the Web Service endpoint
- JMS API user ID

- JMS API password
- Administrator security role mapping
- System monitor security role mapping

Step 3 Business Process Choreographer Explorer and logging:

- Install Business Process Choreographer Explorer
- Context root
- Enable CEI logging
- Enable audit logging
- Install Business Process Choreographer Observer
- JMS user ID (for the observer)
- JMS password (for the observer)

Attention: After the container is configured, you can only change the logging options, retry limit, and retention queue message limit. If you want to change any of the other values, you must remove the existing Business Process Choreographer configuration and then create a new one.

JDBC provider

You must create a new data source that is only used by Business Process Choreographer. When you select your JDBC provider, appropriate defaults are inserted in the Implementation class name field.

Type	Value
Mandatory	Yes
Data type	Drop-down list
Choices for z/OS	Create a new XA data source for z/OS: <ul style="list-style-type: none"> • Cloudscape 5.1 (Cloudscape JDBC Provider (XA)) • DB2 Universal JDBC Driver on z/OS (type 2) <p>Note: If this driver is not selectable from the drop down list, configure this data source as described in “Creating a DB2 for z/OS database for Business Process Choreographer” on page 32.</p>

Implementation class name

The Java class name of the Java Database Connectivity (JDBC) driver implementation. Appropriate defaults are inserted in this field after you select your JDBC provider, there is no need to change this value.

Type	Value
Mandatory	Yes
Data type	String
Default for Cloudscape 5.1 (Cloudscape JDBC Provider (XA))	com.ibm.db2j.jdbc.DB2jXADataSource
Default for DB2 Universal JDBC Driver on z/OS (type 2)	com.ibm.db2.jcc.DB2ConnectionPoolDataSource

For more information about properties and settings for the database, refer to Vendor-specific data sources minimum required settings.

Class path (for JDBC provider)

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

Type	Value
Mandatory	For Cloudscape No, the JDBC driver is already on the WebSphere classpath. For DB2 Universal JDBC Driver on z/OS Yes
Data type	String
Default for Cloudscape 5.1	<code>\${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar</code>
Default for DB2 Universal JDBC Driver on z/OS 7 (type 2)	The value for <code>\${CLOUDSCAPE_JDBC_DRIVER_PATH}</code> is predefined and does not need to be set. <code>\${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar</code> <code>\${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar</code> <code>\${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar</code> The value for <code>\${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar</code> 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 are: On z/OS: <code>/home/db2inst1/sqllib/java</code>

Data source user name

A user ID that has the authority to connect to the database and to modify the data. If the user ID has the authority to create tables and indexes in the database, then the database schema will be updated automatically, when necessary, after applying a service or fix pack.

Type	Value
Mandatory	For Cloudscape No For DB2 Universal JDBC Driver on z/OS 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.

Type	Value
Mandatory	For Cloudscape No For DB2 Universal JDBC Driver on z/OS Yes
Data type	String

Type	Value
Default	None

Custom Properties

Extra parameters that are required by the database system.

CAUTION:

It is not recommended that you change any of the optional properties before you have configured and verified that your business process container is working. Making such changes belongs to advanced tuning and troubleshooting, and can cause your system to stop working.

Type	Value
Mandatory	Yes
Data type	String
Data format	Multiple lines of <i>Property=Value</i>
Minimum required properties	Refer to Vendor-specific data sources minimum required settings.
Properties that are not listed in this table	Properties that are optional or that are ignored are not listed in this table. For information about such properties, refer to the documentation for your JDBC provider.
Required properties	All of the required properties for each JDBC provider are described below.
Required properties for Cloudscape	<p>databaseName = \${USER_INSTALL_ROOT}/databases/BPEDB</p> <p>Required string. Defines which database to access. The value must be a fully qualified path.</p> <p>Remember: After running the wizard, make sure that you create the database in the location specified for databaseName.</p>

Type	Value
Properties for DB2 z/OS 7 (DB2 Universal JDBC Driver Provider)	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.
	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 Business Process Choreographer Optional string. Description of the data source. Not used by the data source object. Used for information purposes only.
	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).
	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

JMS provider

Specifies which messaging service the business process container uses.

Type	Value
Mandatory	Yes
Data type	Drop-down list
Choices	Default messaging provider WebSphere MQ

Queue manager

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

Type	Value
Mandatory	If you selected WebSphere MQ JMS Provider ; otherwise, this field is disabled.
Data type	String
Value	Your queue manager name, for example, <code>BPC_nodeName_serverName</code> .

Class path (JMS provider)

The path to the MQ Java lib directory.

Type	Value
Mandatory	If the WebSphere environment variable <code>{MQ_INSTALL_ROOT}</code> is not defined 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 MQ installation: For z/OS <code>/opt/mqm/java/lib</code>

JMS user ID

Used to authenticate the connection to the Java Message Service (JMS) provider. This user ID must have administration rights for the messaging service.

Type	Value
Mandatory	Yes
Data type	String
Restrictions	If you are using WebSphere default messaging, the JMS user ID must be less than or equal to 12 characters.
Default	The user ID that you used to log into the administrative console.
For z/OS	Use root. The user ID must be a member of the group mqm.

JMS password

The password for the Java Message Service (JMS) user ID.

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

WebService Endpoint context root

The root context used for the Web service.

Type	Value
Mandatory	Yes
Data type	String
Default when configured on a server	/BFMIF_\${nodeName}_\${serverName}
Default when configure on a cluster	/BFMIF_{\$clusterName}

JMS API user ID

The user ID that the business process container message-driven bean (MDB) uses when processing asynchronous API calls.

Type	Value
Mandatory	Yes
Data type	String
Description	If WebSphere security is enabled, even if you do not use the JMS API 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.

Setting up Roles using RACF security:

These RACF permissions apply when the following security field is specified:

- `com.ibm.security.SAF.delegation= true`
`RDEFINE EJBROLE JMSAPIUser UACC(NONE)`
`APPLDATA(' userid')`

JMS API password

The password for the JMS API User ID.

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

Administrator security role mapping

The group from the domain user registry that is mapped onto the role of business process administrator.

Type	Value
Mandatory	Yes
Data type	String
Default	None
Restrictions	The group specified must already exist in the domain user registry. The user registry can be the local operating system, Lightweight Directory Access Protocol (LDAP), or custom registry.

Setting up Roles using RACF security:
These RACF permissions apply when the following security field is specified:

- `com.ibm.security.SAF.authorization=true`

```
RDEFINE EJBROLE BPESystemAdministrator UACC(NONE)
PERMIT BPESystemAdministrator CLASS(EJBROLE)
ID(userid) ACCESS(READ)
```

System monitor security role mapping

The group from the domain user registry that is mapped onto the role of business process monitor.

Type	Value
Mandatory	Yes
Data type	String
Default	None
Restrictions	The group specified must already exist in the domain user registry. The user registry can be the local operating system, Lightweight Directory Access Protocol (LDAP), or custom registry.

Setting up Roles using RACF security:
These RACF permissions apply when the following security fields are specified:

- `com.ibm.security.SAF.authorization=true`

```
RDEFINE EJBROLE BPESystemMonitor UACC(NONE)
PERMIT BPESystemMonitor CLASS(EJBROLE)
ID(userid) ACCESS(READ)
```

Business Process Choreographer Explorer

If this check box is selected, the Business Process Choreographer Explorer is also installed.

Type	Value
Data type	Check box
Default	selected

Context root

This context root becomes part of the URL for the Business Process Choreographer Explorer.

Type	Value
Data type	String
Default	/bpc
Restrictions	If you configure multiple instances of the Business Process Choreographer Explorer, each instance must have a root context that is unique in the WebSphere cell.

Enable audit logging

Audit logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Enable Common Event Infrastructure logging

Common Event Infrastructure (CEI) logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Business Process Choreographer Observer

If this check box is selected, the Business Process Choreographer Observer and event collector are also installed.

Type	Value
Data type	Check box
Mandatory	No
Default	Not selected (the Business Process Choreographer Observer will not be installed)
Dependencies	CEI logging must be enabled before you can select this option.

JMS user ID (for the observer)

The user ID the Business Process Choreographer Observer uses to authenticate against the CEI bus.

Type	Value
Mandatory	If you install the Business Process Choreographer Observer
Data type	String

JMS password (for the observer)

The password for the JMS API User ID.

Type	Value
Mandatory	If you install the Business Process Choreographer Observer
Data type	String

Business process container settings

Use this panel 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* → **Container Settings** → **Business Process Container**.

Note: Changes made on this panel do not affect the server until after it is restarted.

Enable Common Event Infrastructure logging

Common Event Infrastructure (CEI) logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Enable audit logging

Audit logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

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.

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

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 quiesce mode.

Type	Value
Data type	Integer
Default	20

Retention Queue

The JNDI name of the queue that contains messages that cannot be processed currently, and that require a retry later.

Type	Value
Data type	Read-only string
Default	jms/BPERetQueue

Hold Queue

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

Type	Value
Data type	Read-only string
Default	jms/BPEHldQueue

Customizing the WebSphere MQ JMS resources in a cluster

Use this task to customize the connection factory resources for business process containers that are in a cluster and use the WebSphere MQ JMS provider.

Do not perform this task if you are using default messaging. If you are using the WebSphere MQ JMS provider, perform the following steps for each application server in the cluster:

1. Open the connection factory page: Click **Resources** → **JMS Providers** → **WebSphere MQ** → **Scope: Server** → **Apply** → **WebSphere MQ connection factories**.
2. Select the business process container connection factory **BPECF** and set the property values for the type of queue manager configuration that you are using:
 - For a central queue manager:

Property	Description
Host	The host name of the server that is hosting the central queue manager.
Port	The port number that the central queue manager is using.
Transport Type	Client
Client ID	The message channel agent (MCA) user ID to use. This is normally the owner or creator of the queue manager, typically this is the root user..
CCSID	Use the value 819 .

- For a cluster of queue managers:

Property	Description
Transport Type	Bindings or Client
Queue Manager	The name of the server get queue manager.

When using WebSphere MQ, the local bindings transport type is slightly faster than using the client transport type, but has the effect that you must stop the entire application server to stop the local WebSphere MQ queue manager. If you specify `Client`, you must also provide the host name and port number for the get queue manager.

3. Select the business process container connection factory **BPECFC** and set the property values for the type of queue manager configuration you are using:
 - For a central queue manager:

Property	Description
Host	The host name of the server that is hosting the central queue manager.
Port	The port number that the central queue manager is using.
Transport Type	<code>Client</code>
Client ID	The message channel agent (MCA) user ID to use. This is normally the owner or creator of the queue manager, typically this is the root user.
CCSID	Use the value 819

- For a cluster of queue managers:

Property	Description
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	<code>Client</code>
Client ID	The message channel agent (MCA) user ID to use. This is normally the owner or creator of the queue manager, typically this is the root user.
CCSID	819

4. Select the human task manager connection factory **HTMCF** and set the property values for the type of queue manager configuration that you are using:
 - For a central queue manager:

Property	Description
Host	The host name of the server that is hosting the central queue manager.
Port	The port number that the central queue manager is using.
Transport Type	<code>Client</code>
Client ID	The message channel agent (MCA) user ID to use. This is normally the owner or creator of the queue manager, typically this is the root user..
CCSID	Use the value 819 .

- For a cluster of queue managers:

Property	Description
Transport Type	<code>Bindings</code> or <code>Client</code>
Queue Manager	The name of the server get queue manager.

When using WebSphere MQ, the local bindings transport type is slightly faster than using the client transport type, but has the effect that you must stop the entire application server to stop the local WebSphere MQ queue manager. If you specify `Client`, you must also provide the host name and port number for the get queue manager.

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

Continue configuring at step 20 on page 28.

Configuring the human task container, using the installation wizard

Use this task to configure the human task container.

Before configuring the human task container, see information on configuring the business process container manually in Chapter 2, “Configuring Business Process Choreographer,” on page 11.

If you have run the `bpeconfig.jacl` script, the human task container is already configured. The following steps describe how to configure the human task container using the installation wizard.

1. In the administrative console, select the server or cluster where you want to install the business process container. Click one of the following:

- **Servers** → **Application Servers** → *serverName*
- **Servers** → **Clusters** → *clusterName*

Where *serverName* or *clusterName* is the name of the application server or cluster where you want to install the human task container.

2. In the **Container Settings** section, click **Human task container settings** → **Human task container** → **Human task container installation wizard** (in the **Additional Properties** section). Where possible, the installation wizard offers appropriate default values in the parameter fields, you can view the recommended values on the “Human task container installation wizard settings” on page 48.
3. Verify that the human task container is not configured. There should be a message indicating that the Human Task Manager is not currently installed. If the human task container is already configured, remove the configuration before you start the installation wizard. For details about how to remove the configuration, see [Removing the Business Process Choreographer configuration](#).
4. Select the JMS provider and security settings (step 1):
 - a. In the drop-down list for **JMS provider**, select the messaging service that is used by the business process container.
 - For default messaging, select `Default messaging provider`.
 - For WebSphere MQ, select `WebSphere MQ`.
 - b. Use the default value for **Queue Manager** (`BPC_nodeName_serverName`). If you are using the default messaging provider, this field is ignored.
 - c. If you are using external messaging (WebSphere MQ JMS provider) and you have not defined the WebSphere environment variable `{MQ_INSTALL_ROOT}`, make sure that **Classpath** points to the WebSphere MQ Java lib directory.

- d. For the **JMS user ID**, enter a user ID that has administration rights for the messaging service. This user ID will be used to connect to the JMS queue manager. Use root
 - e. For the **JMS password**, enter the password for the JMS user ID.
 - f. For the **Webservices Endpoint**, enter the Webservice endpoint for the Webservice API.
 - g. For **Escalation user ID**, enter the user ID that will be used by the human task container to perform scheduled actions, for example triggering escalations to verify the expected task state, timed task deletion, and task expiration. Use root
 - h. For **Escalation password**, enter the password for the escalation user ID.
 - i. For the **Administrator security role mapping**, enter the name of the group, defined in the user registry, that will map onto the role of Business Process Administrator. On Windows systems, for example, you can specify the group Administrators.
 - j. For the **System monitor security role mapping**, enter the name of the group in the user registry to map onto the role of Business Process System Monitor. On Windows systems, for example, you can specify the group Administrators.
 - k. Click **Next** to go to step 2 in the installation wizard.
5. **Optional:** Select **Mail session** to create the default mail session resource with cell scope.
 - If you are configuring the human task container on a server, the default mail session is named `mail/HTMNotification_nodeName_serverName`.
 - If you are configuring the human task container on a cluster, the default mail session is named `mail/HTMNotification_clusterName`.

Attention: If this is not set, no escalation mails are sent.
 6. **Optional:** To use the Common Event Infrastructure, select **Enable Common Event Infrastructure logging** .
 7. **Optional:** To enable the audit log, select **Enable audit logging for all human tasks** .
 8. Click **Next** to view the Summary (step 3).
 9. Check that the information on the summary page is correct. The summary includes reminders of which external resources are necessary. If you have not already created them, you can continue configuring the human task container, but you must create the resources before you activate the human task container. Printing the summary page helps you to create the correct resources.
 - a. To make corrections, click **Previous**.
 - b. To install the human task container and define its resources, click **Finish**. The progress is shown on the Installing page.
 - c. Verify that no error messages are displayed.
 10. If you selected the Mail session option in step 5, you must set the mail transport host:
 - a. Click **Resources** → **Mail Providers**.
 - b. Select the cell scope: **Built-in Mail Provider**.
 - c. Under **Mail sessions**, click **HTMMailSession_scope**, where *scope* consists of the cluster name or the node and server names, then set the **Mail transport host**.

- d. If the mail transport host is secured, also set **Mail transport user ID** and **Mail transport password**.
 - e. Click **OK**.
11. Click **Save Master Configuration**, then click **Save**.
 12. Restart the application server.
 13. If the container did not install successfully, check for any error messages that can help you correct the problem, then repeat this task.
Check the administrative console or the `SystemOut.log` file for the application server. On a cluster, check the log for all application servers in the cluster.

The human task container is configured.

Continue configuring at step 3 on page 12.

Human task container installation wizard settings

Use the installation wizard to install and configure the human task container.

Access the human task container installation wizard by clicking **Servers** → **Application servers** → *server_name*. Then in the **Container Settings** section, click **Human task container settings** → **Human task container** → **Human task container installation wizard**. This page describes the installation wizard fields, in the order in which they are displayed in the wizard.

Step 1 JMS provider and security:

- JMS provider
- Queue manager
- Class path
- JMS user ID (for the JMS provider)
- JMS password (for the JMS provider)
- Webservice Endpoint context root
- Escalation user ID
- Escalation password
- Administrator security role mapping
- System monitor security role mapping

Step 2 Mail session and logging:

- Mail session
- Enable CEI logging
- Enable audit log

Attention: After applying these fields, you can only enable and disable the logging options.

JMS provider

Specifies which messaging service the human task container uses.

Type	Value
Mandatory	Yes
Data type	Drop-down list
Choices	WebSphere MQ Default messaging provider

Type	Value
Default	Default messaging provider

Queue manager

The name of the queue manager that is used by the human task container.

Type	Value
Mandatory	If you selected WebSphere MQ JMS Provider ; otherwise, this field is disabled.
Data type	String
Value	Your queue manager name, for example, <i>BPC_nodeName_serverName</i> .

Class path

The path to the MQ Java lib directory.

Type	Value
Mandatory	If the WebSphere environment variable <code>MQ_INSTALL_ROOT</code> is not defined 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 MQ installation: For z/OS <code>/opt/mqm/java/lib</code>

JMS user ID

Used to authenticate the connection to the Java Message Service (JMS) provider. This user ID must have administration rights for the messaging service. It will be used to connect to the JMS queue manager.

Type	Value
Mandatory	Yes
Data type	String
Restrictions	If you are using WebSphere default messaging, the JMS user ID must be less than or equal to 12 characters.
Default	The user ID that you used to log into the administrative console.
For z/OS	Use root. The user ID must be a member of the group <code>mqm</code> .

JMS password

The password for the Java Message Service (JMS) user ID.

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

Webservice Endpoint context root

The root context used for the Web service.

Type	Value
Mandatory	Yes
Data type	String
Default when configured on a server	<i>/HTMIF_nodeName}_serverName</i>
Default when configured on a cluster	<i>/HTMIF_clusterName</i>

The context root entered here is embedded in the URL for the Web service endpoint: `http://host:port/contextRoot/sca/com/ibm/task/api/sca/HTMWS`.

Escalation user ID

A user ID used by the human task container to perform scheduled actions.

Type	Value
Mandatory	Yes, even if no escalation mails will be sent.
Data type	String
Description	This is the run-as user ID for the Human Task Manager message driven bean (MDB) to perform scheduled escalation, deletion and expiration actions.

Escalation password

The password for the escalation user ID.

Type	Value
Mandatory	Yes
Data type	String

Administrator security role mapping

The group from the user registry that is mapped onto the role of task administrator.

Type	Value
Mandatory	Yes
Data type	String
Default	None
Restrictions	The user registry can be the local operating system, Lightweight Directory Access Protocol (LDAP), or custom registry. The group that is specified must already exist in the user registry being used.

System monitor security role mapping

The group from the user registry that is mapped onto the role of task monitor.

Type	Value
Mandatory	Yes
Data type	String
Default	None

Type	Value
Restrictions	The user registry can be the local operating system, Lightweight Directory Access Protocol (LDAP), or custom registry. The group that is specified must already exist in the user registry being used.

Mail session

If you select the mail session check box, a mail session will be created with cell scope. This is necessary for sending escalation mails.

Type	Value
Data type	Check box
Default	Not selected

- On a server, the default mail session name is `mail/HTMNotification_node_name_server_name`.
- On a cluster, the default mail session name is `mail/HTMNotification_cluster_name`.

Enable Common Event Infrastructure logging

Common Event Infrastructure (CEI) logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Enable audit logging

Audit logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Human task container settings

Use this panel to manage human task containers.

A human task container provides services to run human task within an application server. To view this administrative console page, click **Servers** → **Application Servers** → *server_name* → **Human task container settings** → **Human task container**.

Note: Changes made on this panel do not affect the server until after it is restarted.

E-mail session JNDI name

The Java Naming and Directory Interface (JNDI) name of the mail session resource that will be used by the human task container to send escalation mails.

Type	Value
Data type	Read-only string
Default when configured on a server	<code>mail/HTMNotification_nodeName_serverName</code>
Default when configured on a cluster	<code>mail/HTMNotification_clusterName</code>

Enable Common Event Infrastructure logging

Common Event Infrastructure (CEI) logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Enable audit logging

Audit logging can be enabled or disabled.

Type	Value
Data type	Check box
Default	Not selected

Sender e-mail address

The address that will be shown as the sender for escalation e-mails.

Type	Value
Data type	String
Default	taskmanager.emailservice @htm.companydomain

Escalation URL prefix

This prefix is used to provide a link in escalation e-mails for more details about the escalation.

Type	Value
Data type	String
Default	None

Task URL prefix

The prefix for the URL that is included in e-mails for more details about a task.

Type	Value
Data type	String
Default	None

Administrator URL prefix

The prefix for the URL that is included in the escalation e-mail received by recipients who perform the task administrator role.

Type	Value
Data type	String
Default	None

Process Explorer URL prefix

The prefix for the URL for the Business Process Choreographer Explorer.

Type	Value
Data type	String
Default	None

Staff query refresh schedule

The schedule for refreshing staff queries.

Type	Value
Data type	String
Format	This field uses the crontab format <Minute> <Hour> <Day_of_the_Month> <Month_of_the_Year> <Day_of_the_Week> <Command> according to com.ibm.websphere.scheduler.UserCalendar
Example	To refresh the staff queries once per hour on Thursdays, use the schedule 0 * * * 4 ?
Default	0 0 1 * * ? (midnight on the first of each month)

Timeout for staff query result

The duration in seconds that the results of a staff query are considered to be valid. After this duration, the staff query results will expire.

Type	Value
Data type	Integer
Default	3600
Units	Seconds

Enable group work items

Select this check box to enable group work items. Any applications that were written for a version before Version 6.0.2 that use one or both of the interfaces `com.ibm.bpe.api.StaffResultSet` or `com.ibm.task.api.StaffResultSet` should be modified in order to cope with group work items before group work items are enabled.

Type	Value
Data type	Check box
Default	Not selected (group work items are not enabled to ensure compatibility with applications written before Version 6.0.2)

Human task container custom properties

Use this panel to manage custom properties for the human task container.

Use custom properties to set additional configuration parameters for the human task container, for example, for sending e-mails when escalation events occur. To view this administrative console page, click **Servers** → **Application Servers** → *server_name* → **Human task container settings** → **Human Task Container** → **Custom Properties**.

EscalationEmail.ClientDetailURL:

The URL for the escalation details view in Business Process Choreographer Explorer or in a custom Web client application. You can use the `htm:task.ClientDetailURL` variable to query the value of this property. This property is only for compatibility with version 6.0.1 tasks. If your tasks were created using Version 6.0.2 or later, this property has no effect.

Type	Value
Data type	String
Default	http://clientapphost[:port]/bpc/DetailsView.jsp?id=

EscalationEmail.Subject:

The e-mails that are sent when an escalation occurs have the subject that you specify here. You can use task or process variables in the subject string. This property is only for compatibility with version 6.0.1 tasks. If your tasks were created using Version 6.0.2 or later, this property has no effect.

Type	Value
Data type	String
Default	The task '%htm:task.displayName%' has been escalated

EscalationEmail.Template:

The URL for the template file that is used for the body of escalation e-mails. The file can be an HTML file or a text file and it can contain task and process variables that are resolved when the e-mail is created. This property is only for compatibility with version 6.0.1 tasks. If your tasks were created using Version 6.0.2 or later, this property has no effect.

Type	Value
Data type	String
Default	file:\${WAS_INSTALL_ROOT}/ProcessChoreographer/sample/emailNotification.html

Configuring the LDAP staff plug-in provider

Use this task to configure the LDAP staff plug-in provider that Business Process Choreographer uses to determine who can start a process or claim an activity or a task.

Each type of supported user directory service requires a corresponding staff plug-in. The following staff plug-ins are supported:

Table 1. Supported staff plug-in providers

User directory service	Plug-in provider
Lightweight Directory Access Protocol (LDAP)	LDAP Staff Plug-in Provider
Local operating system user registry	System Staff Plug-in Provider
WebSphere Application Server user registry	User Registry Staff Plug-in Provider

All of these plug-ins are already installed. You can use the user registry and system plug-ins without any configuration.

The LDAP staff plug-in is configured for an LDAP server with anonymous access; the LDAP server is local to the installed application server. You can change the configuration of the LDAP plug-in.

1. In the administrative console, click **Resources** → **Staff Plugin Provider**.
2. If the scope is not set to Node, select **Node** and click **Apply**.
3. 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** → **Browse**, and select the sample Extensible Stylesheet Language (XSL) transformation file to use. The standard XSL transformation for LDAP is located 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.
 - d. Click **Next**.
 - e. Enter an administrative name for the staff plug-in provider.
 - f. Enter a description.
 - g. Enter the Java Naming and Directory Interface (JNDI) name for business processes to use in referencing this plug-in, for example, `bpe/staff/ldapserver1`.
 - h. Click **Apply**.
 - i. Click **Custom Properties**.
 - j. 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**.
 - k. 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, <code>mycomputer/My LDAP Alias</code> . You must define this alias in the administrative console by clicking Security → Global security → 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, <code>"o=mycompany, c=us"</code>
CasesentivenessForObjectclasses	Optional	Determines whether the names of LDAP object classes are case-sensitive.
ContextFactory	Required	Sets the Java Naming and Directory Interface (JNDI) context factory, for example, <code>com.sun.jndi.ldap.LdapCtxFactory</code>
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, <code>ldap://localhost:389</code>

LDAP plug-in property	Required or optional	Comments
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.

4. To activate the plug-in, stop and start the server.
5. If you have problems with any of these steps, refer to the *Troubleshooting WebSphere Process Server* PDF.

Processes can now use the staff support services to resolve staff queries, and to determine which activities can be performed by certain people.

Continue configuring at step .

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*. Then in the section **Business Integration**, click **Staff Service**.

Enable service at server startup

Specifies whether the server attempts to start the staff service.

Type	Value
Default	Selected
Range	<p>Selected</p> <p>When the application server starts, it attempts to start the staff service automatically.</p> <p>Cleared</p> <p>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.</p>

Staff plug-in provider collection

A staff plug-in is responsible for retrieving user information. Use this panel to manage staff plug-in providers.

To view this administrative console panel, click **Resources** → **Staff plug-in provider**. Existing plug-in providers are displayed.

Normally, staff plug-in providers are only defined with Node scope. The scope of the staff plug-in providers that are displayed in the table is indicated by an arrow

to a selected radio button for Cell, Node, or Server scope. To view staff plug-in providers with a different scope, select the radio button for the desired scope and click **Apply**.

To view or change the properties for an existing provider, click on the name of the provider. To configure a new provider, click **New**.

Name

The name by which the staff plug-in provider is known for administrative purposes.

Type	Value
Data type	String

Description

A description of the staff plug-in provider.

Type	Value
Data type	String

Staff plug-in provider settings

Use this panel to modify the settings for a staff plug-in provider.

Staff plug-ins are used to get information from a directory of users. Each staff plug-in provider is registered with the runtime 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, first click **Resources** → **Staff plug-in provider**. You will see a table that lists all staff plug-in providers that have the scope indicated.

Normally, staff plug-in providers are only defined with Node scope. The scope of the staff plug-in providers that are displayed in the table is indicated by an arrow to a selected radio button for Cell, Node, or Server scope. To view staff plug-in providers with a different scope, select the radio button for the desired scope and click **Apply**.

To view or modify the configuration for a staff plug-in and any custom properties that it has, click on the name of the plug-in, *staffpluginprovider_name*.

Scope

The scope for this staff plug-in provider.

Type	Value
Data type	Read-only string
Valid values	The name of the cell, node, or server. For example: cells:viennaNode02Cell:nodes:viennaNode02
Description	The scope determines the level at which the resource definition is visible. Use Node scope for staff plug-in configurations because there are settings that are specific to a node.

Name

The name by which the staff plug-in provider is known for administrative purposes.

Type	Value
Data type	String

Description

A description of the staff plug-in provider.

Type	Value
Data type	String

JAR File

The file name, including the absolute path, of the JAR file containing the plug-in.

Type	Value
Data type	Read-only string

Staff plug-in 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. Click **New** to create a new configuration, or click on the name of an existing configuration to view or change its properties.

To view this administrative console page, click **Resources** → **Staff plug-in provider** → *staffpluginprovider_name* → **Staff Plug-in Configuration**.

Name

The name of the staff plug-in configuration used for administrative purposes. Click on the name to view or change its configuration settings.

Type	Value
Data type	String

Description

A description of the staff plug-in configuration.

Type	Value
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.

Type	Value
Data type	String

XSL Transform File

The file name, including the absolute path, of the Extensible Style Language (XSL) transformation file.

Type	Value
Data type	String

Staff plug-in configuration settings

Use this page to view or modify the settings for a staff plug-in configuration.

To view this administrative console page, click **Resources** → **Staff plug-in provider** → *staffpluginprovider_name* → **Staff Plug-in Configuration** → *staffpluginconfiguration_name*.

Scope

The scope for this staff plug-in provider. The scope determines the level at which the resource definition is visible.

Type	Value
Data type	Read-only string
Valid values	The name of the cell, node, or server. For example: cells:viennaNode02Cell:nodes:viennaNode02

Name

The name of the staff plug-in configuration used for administrative purposes.

Type	Value
Data type	String

Description

A description of the staff plug-in configuration.

Type	Value
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.

Type	Value
Data type	String

XSL Transform File

The file name, including the absolute path, of the Extensible Style Language (XSL) transformation file. Default XSL transform files are provided for the sample plug-ins. If you have customized the transform file, specify the path to your file. The path name can include WebSphere environment variables.

Type	Value
Data type	String

About the staff service

With Business Process Choreographer you can separate the logic of your business processes and human tasks from the staff resolution. Staff queries are resolved using a plug-in that is specific to the directory service. The basic aspects of using the staff service are described below:

- “Staff query and staff service concept”
- “Implementing a staff query” on page 61
- “Staff query verb set” on page 61
- “Repository-specific staff queries” on page 63
- “Staff verb XSL transformation files” on page 64
- “Using task and process context variables in staff verbs” on page 65
- “E-mail verb set” on page 66

For detailed information on the staff resolution plug-ins, refer to the *Process Choreographer: Staff Resolution Architecture*, the *Process Choreographer: Programming Model for Staff Resolution*, and the *Process Choreographer: Staff Resolution Parameter Reference* White papers in WebSphere Business Process Choreographer

Staff query and staff service concept

Use WebSphere Integration Developer to define staff queries for the staff support service. Staff queries are based on staff query templates, staff verbs, and are associated with the roles foreseen for human tasks and business processes, such as ProcessStarter and PotentialOwners.

A staff verb is identified by a unique name and includes a set of query parameters. The parameterized staff verb is transformed at application deployment time to determine a repository-specific staff query. This is used during execution of a business process or human task to retrieve the identities of users from a user repository.

Every business process or human task is associated with a specific staff plug-in configuration by its JNDI name. The configuration is extracted at deployment time from the process or task definition, and is used to map every staff verb found to a repository specific staff query. The mapping is governed by an XSL transformation file which takes a staff verb as input and produces the corresponding repository-specific query as output.

By default, three staff plug-in providers are included, representing different user repository options:

- The LDAP staff plug-in provider is used to generate staff queries which can be executed against an LDAP server.
- The user registry staff plug-in provider is used to generate staff queries which can be executed against the WebSphere Application Server user registry.
- The system staff plug-in provider is not associated with a user repository. Instead, it returns user identities that are derived directly from the staff verb parameters. This plug-in provider is intended for testing and prototyping purposes.

Each of the above staff plug-in providers is associated with at least one configuration. In particular, a configuration specifies an XSL transformation file that performs the mapping between staff verbs and staff queries that are specific to the repository. The following transformation files are provided by default:

- The LDAPTransformation.xml file maps of staff verbs to LDAP-specific staff queries which can be executed via an JNDI interface.
- The UserRegistryTransformation.xml file maps staff verbs to staff queries that are specific to the WebSphere user registry.
- The SystemTransformation.xml file maps staff verbs to the actual user IDs specified in the verbs. It does not require a real user repository.
- The EverybodyTransformation.xml file maps all staff verbs to the default result "everybody". It does not require a real user repository.

Implementing a staff query

The following example summarizes the steps involved in implementing a staff query:

1. Using WebSphere Integration Developer, a modeler associates a newly created task with the staff plug-in configuration bpe/staff/sampleldapconfiguration.
2. Using WebSphere Integration Developer, the modeler associates the roles for the task with corresponding staff verbs, for example, PotentialOwners is associated with the staff verb "Group Members" including the parameters :
 - "GroupName" set to the value "cn=group1,dc=mycomp,dc=com"
 - "IncludeSubgroups" set to the value "true"
3. In the context of the task, WebSphere Integration Developer stores the verb definition as an XML snippet:


```
<verb>
  <name>Group Members</name>
  <parameter id="GroupName">cn=group1,dc=mycomp,dc=com</parameter>
  <parameter id="IncludeSubgroups">true</parameter>
</verb>
```
4. When the task is deployed in WebSphere Application Server, the staff support service establishes that the LDAP staff plug-in provider bpe/staff/sampleldapconfiguration is to be used. The associated LDAPTransformation.xml file is used to transform the staff verb into an LDAP-specific query, which is stored internally.

Staff query verb set

The staff support service accepts queries in an abstract form that is independent of the user repository infrastructure. Both the process editor and the task editor have a set of predefined staff verbs that can be used when you model processes and tasks. These verbs are contained in the VerbSet.xml file. This file is installed with WebSphere Integration Developer.

The individual staff resolution plug-ins and the XSLT mapping files do not support all of the verbs. The *Manager of Employee* verb, for example, is not available if you use the user registry or the system plug-in. You can modify the 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 can be used 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. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Group Use this verb to retrieve the name of the group that matches the groupName parameter. This is used with group work items. This verb is supported by all plug-ins.

Everybody

Use this verb to assign a work item to every user authenticated by the WebSphere Process 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 up to three groups. 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. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

Group Members without Named Users

Use this verb to define a query to retrieve the members of a group except for explicitly named users of that group. One or more members can be specified for exclusion as a comma separated list. 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.

Group Members without Filtered Users

Use this verb to define a query to retrieve the members of a group except for a set of users defined by an LDAP search filter. 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.

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. You might need to customize the default mapping XSLT file to match the LDAP schema of your organization.

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; For inline tasks, only the business process administrator and business process system administrator have access. For standalone tasks, only the human task administrator and human task system administrator have access. Depending on the API used, the authorized J2EE administrator will be different. For the business process API this will be the

BPESystemAdministrator user, for the human task API this will be the TaskSystemAdministrator user. 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 staff repository 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 purposes. 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 and task templates, this verb is useful in combination with context queries, for example:

```
User [username='%wf:process.starter%']
```

This verb is useful for testing purposes. This verb is supported by the system, user registry, and LDAP plug-ins.

Users by user ID without Named Users

Use this verb to define a staff query for users whose user ID is known and exclude explicitly named user IDs. Even though it is not recommended that you hard code user IDs in process and task templates, this verb is useful in combination with context queries, for example:

```
User [userID='%htm:task.potentialStarters%', NamedUsers='%wf:activity(...).owner%']
```

Repository-specific staff queries

The XSL transformation file that is associated with a staff plug-in configuration is used to generate staff queries that are specific to a particular repository. Each query can be executed by the respective staff plug-in to obtain a list of user IDs. The predefined queries which are available to a staff plug-in correspond to the calls which can be executed by the plug-in and are therefore fixed.

Based on predefined queries, more complex queries can be formed using the following mechanisms:

- A union of query results implies that the user IDs returned by the individual queries will be added to the current result list of user identities. For example: The LDAP staff plug-in allows, among others, for predefined queries of the following types:

The list of user IDs for the group members of a specified group:

```
<ldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
```

```
...
```

```
</ldap:usersOfGroup>
```

The user ID of a specified user:

```
<ldap:user dn="uid=user1,dc=mycomp" .../>
```

A complex query can be constructed for the list of user IDs for the members of the specified group, plus the identity of the specified user:

```
<ldap:staffQueries>
  <ldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
    ...
  </ldap:usersOfGroup>
  <ldap:user dn="uid=user1,dc=mycomp" .../>
</ldap:staffQueries>
```

- A difference of query results implies that user IDs returned by a <remove> query will be removed from the current result list. For example, removing "user1" from the list of IDs retrieved for the specified group members:

```
<ldap:staffQueries>
  <ldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
    ...
  </ldap:usersOfGroup>
  <ldap:remove value="user1"/>
</ldap:staffQueries>
```

- Referencing query results implies that the results obtained from one query are used to influence the behavior of in a subsequent query. For example, in the following snippet, two queries are performed. First, the value of the "manager" attribute in the LDAP entry for the user "uid=user1,..." is retrieved and saved in an intermediate variable "supervisor", which is then used to look up the manager's LDAP entry and retrieve the associated user identity.

```
<ldap:staffQueries>
  <ldap:intermediateResult name="supervisor">
    <ldap:user dn="uid=user1,dc=mycomp" attribute="manager" ... />
  </ldap:intermediateResult>
  <ldap:user dn="%supervisor%" .../>
</ldap:staffQueries>
```

Staff queries constructed according to above three combination rules can be executed by the staff plug-ins. For a detailed description of all predefined staff queries for each of the supported staff plug-ins and more examples of combining them consult the *Process Choreographer: Staff Resolution Parameter Reference* White papers in WebSphere Business Process Choreographer.

Staff verb XSL transformation files

The XSL transformation file specified for a staff plug-in configuration defines the mapping between staff verbs and repository-specific staff queries. Every staff plug-in configuration is expected to have its own XSL transformation file.

The default transformation files are:

- LDAPTransformation.xml for the LDAP staff provider plug-in
- UserRegistryTransformation.xml for the user registry staff provider plug-in
- SystemTransformation.xml and EverybodyTransformation.xml for the system staff provider plug-in

These transformation files map the predefined set of staff verbs to corresponding simple and composite repository-specific queries. These files are located in the *install_root/ProcessChoreographer/Staff* directory.

The transformation files assume certain semantics for staff verbs and their execution using generated repository-specific staff queries. If other semantics are required, the mapping in the transformation file must be changed accordingly.

For example, the LDAP staff plug-in comes with a predefined staff verb:


```

<staff:verb>
  <staff:name>Manager of Employee</staff:name>
  <staff:parameter id="EmployeeName">
    uid=anEmployeeName,cn=users,dc=ibm,dc=com
  </staff:parameter>
</staff:verb>

```

This is mapped by the LDAPTransformation.xml file to an LDAP query:

```

<sldap:staffQueries>
  <sldap:intermediateResult name="supervisor">
    <sldap:user dn="anEmployeeName" attribute="manager"
      objectclass="inetOrgPerson"/>
  </sldap:intermediateResult>
  <sldap:user dn="%supervisor%" attribute="uid" objectclass="inetOrgPerson"/>
</sldap:staffQueries>

```

Which explicitly assumes that the LDAP DN of the supervisor is stored in the employee's attribute "manager". If that the verb is to have different semantics, for example, if the supervisor should come from the LDAP attribute "teacher". Then the LDAP specific query must be changed accordingly:

```

<sldap:staffQueries>
  <sldap:intermediateResult name="supervisor">
    <sldap:user dn="anEmployeeName" attribute="teacher"
      objectclass="inetOrgPerson"/>
  </sldap:intermediateResult>
  <sldap:user dn="%supervisor%" attribute="uid" objectclass="inetOrgPerson"/>
</sldap:staffQueries>

```

The means to achieve this is to adapt the LDAPTransformation.xml file accordingly:

```

<xsl:template name="ManagerOfEmployee">
  <sldap:staffQueries>...
  <sldap:intermediateResult>
    <xsl:attribute name="name">supervisor</xsl:attribute>
    <sldap:user>
      <xsl:attribute name="dn">
        <xsl:value-of select="staff:parameter[@id='EmployeeName']"/>
      </xsl:attribute>
      <xsl:attribute name="attribute">teacher</xsl:attribute>
      ...
    </sldap:user>
  </sldap:intermediateResult>
  <sldap:user>
    <xsl:attribute name="dn">%supervisor%</xsl:attribute>
    ...
  </sldap:user>
</sldap:staffQueries>
</xsl:template>

```

You can get an deeper understanding of the mapping behavior by viewing the default transformation files. The semantics of the default transformations are described in "Staff query verb set" on page 61.

Using task and process context variables in staff verbs

In certain staff verbs, you can use business process and human task context variables as parameter values. This enables the staff support service to resolve staff verbs at run time, based on information supplied by the contexts. For example, the staff verb:

```

<verb>
<name>Users by user ID</staff:name>
  <parameter id="UserID">%htm:input.\name%</staff:parameter>
</verb>

```

specifies as a parameter, the task context variable `htm:input.\name`, which denotes the "name" part of the input message received by the task when it is initiated. The staff support service dynamically replaces the context variable with the actual task context value.

For a description of the verbs and the parameters in which you can use context variables, see "Predefined staff verbs."

E-mail verb set

The e-mail verb set in WebSphere Integration Developer is for e-mail notifications for task escalations. These e-mail verbs are transformed during modeling and deployment into a set of queries that can be run on a staff repository. E-mail verbs are defined for the most common staff verbs supported by the LDAP plug-in. The following e-mail verbs are available:

- Email Address for Department Members
- Email Address for Group Members
- Email Address for Group Members without Names Users
- Email Address for Group Members without Filtered Users
- Email Address for Group Search
- Email Address for Role Members
- Email Address for Users
- Email Address for Users by User ID

For the other LDAP staff verbs, the user identifiers retrieved by the staff verbs are used as input to the Email Address for Users by User ID verb.

Before the e-mail verbs can be run as queries on a specific staff repository, they must be translated into executable queries using the LDAP XSL transformation. The result of a transformation (mapping) can be run by the LDAP staff resolution plug-in. At run time, the query returns a set of e-mail addresses, for example, `user1@mycomp.com`, `user2@mycomp.com`, and so on.

Predefined staff verbs

Predefined staff verbs are provided for creating queries against a staff repository.

You can use staff verbs in WebSphere Integration Developer to model staff assignments in a business process or human task. You can only use the staff verbs that are available for your staff repository type. The staff verbs are transformed during modeling and deployment into a set of queries that can be run on a staff repository.

Predefined staff verbs for LDAP:

Describes the predefined staff verbs and parameters for use with the LDAP staff plug-in for Business Process Choreographer.

You can use staff verbs in WebSphere Integration Developer to model staff assignments in a business process or human task. These staff verbs are transformed during modeling and deployment into a set of queries that can be run on an LDAP staff repository. The parameters for the following predefined staff verbs are listed here:

- Department Members
- Group

- Everybody
- Group Members
- Group Members without Named Users
- Group Members without Filtered Users
- Group Search
- Manager of Employee
- Manager of Employee by user ID
- Native Query
- Nobody
- Person Search
- Role Members
- Users
- Users by user ID
- Users by user ID without Named Users

Department Members

Use this verb to define a query to retrieve the members of a department.

Parameter	Use	Type	Description
DepartmentName	Mandatory	string	Department name of the users to retrieve.
IncludeNestedDepartments	Mandatory	boolean	Specifies whether nested departments are considered in the query.
Domain	Optional	string	The domain to which the department belongs. Use this parameter to limit the query to a subset of the directory.
AlternativeDepartmentName1	Optional	string	An additional department to which the users can belong.
AlternativeDepartmentName2	Optional	string	An additional department to which the users can belong.

Group

Use this verb to define a query to authorize the members of the group.

Parameter	Use	Type	Description
GroupId	Mandatory	string	The name of the group of users to authorize.

Everybody

Use this verb to assign a work item to every user authenticated by WebSphere Process Server. This verb has no parameters.

Group Members

Use this verb to define a query to retrieve the members of a group.

Parameter	Use	Type	Description
GroupName	Mandatory	string	Group name of the users to retrieve.
IncludeSubgroups	Mandatory	boolean	Specifies whether nested subgroups are considered in the query.

Parameter	Use	Type	Description
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	An additional group to which the users can belong.
AlternativeGroupName2	Optional	string	An additional group to which the users can belong.

Group Members without Named Users

Use this verb to define a query to retrieve all of the members of a group except for the explicitly named users.

Parameter	Use	Type	Description
GroupName	Mandatory	string	Group name of the users to retrieve. Supports custom properties that are evaluated at run time.
IncludeSubgroups	Mandatory	boolean	Specifies whether nested subgroups are considered in the query.
NamedUsers	Mandatory	string	The user IDs of the users to exclude from the retrieved group members list. Supports context variables and custom properties, such as %htm:task.originator%

Group Members without Filtered Users

Use this verb to define a query to retrieve the all of the members of a group except for a set of users that is defined by an LDAP search filter.

Parameter	Use	Type	Description
GroupName	Mandatory	string	Group name of the users to retrieve.
IncludeSubgroups	Mandatory	boolean	Specifies whether nested subgroups are considered in the query.
FilterAttribute	Mandatory	string	Name of the attribute to use in the LDAP filter.
FilterValue	Mandatory	string	Filter value to use in the LDAP filter.

Group Search

Use this verb to search for a group based on an attribute match and to retrieve the members of the group. You must set one attribute. If you set more than one attribute, only the first attribute is evaluated.

Parameter	Use	Type	Description
GroupID	Optional	string	The group ID of the users to retrieve.
Type	Optional	string	The group type of the users to retrieve.
IndustryType	Optional	string	The industry type of the group to which the users belong.

Parameter	Use	Type	Description
BusinessType	Optional	string	The business type of the group to which the users belong.
GeographicLocation	Optional	string	An indication of where the users are located.
Affiliates	Optional	string	The affiliates of the users.
DisplayName	Optional	string	The display name of the group.
Secretary	Optional	string	The secretary of the users.
Assistant	Optional	string	The assistant of the users.
Manager	Optional	string	The manager of the users.
BusinessCategory	Optional	string	The business category of the group to which the users belong.
ParentCompany	Optional	string	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	Description
EmployeeName	Mandatory	string	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	Description
EmployeeUserID	Mandatory	string	The user ID of the employee whose manager is retrieved. Supports context variables and custom properties, such as %wf:process.starter%
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	Description
QueryTemplate	Mandatory	string	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.

Parameter	Use	Type	Description
Query	Mandatory	string	Specifies the query. You can use context variables and custom properties, such as %wf:process.starter%. The type of query depends on the query template. <ul style="list-style-type: none"> search template: search filter user template: user dn usersOfGroup: group dn
AdditionalParameter1	Optional	string	Specifies the query. You can use context variables, such as %wf:process.starter%. The type of parameter depends on the query template. <ul style="list-style-type: none"> 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	Use this verb to specify an additional parameter.
AdditionalParameter3	Optional	string	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.
AdditionalParameter4	Optional	string	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.
AdditionalParameter5	Optional	string	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.

Nobody

For inline tasks, only the business process administrators have access. For standalone tasks, only the human task administrators have access. In addition, when using the Business Flow Manager API the BPESystemAdministrator role members have access, for the Human Task Manager API the TaskSystemAdministrator role members have access. This verb has no parameters.

Person Search

Use this verb to search for people based on an attribute match. You must set one attribute. If you set more than one attribute, only the first attribute is evaluated.

Parameter	Use	Type	Description
UserID	Optional	string	The user ID of the users to retrieve.
Profile	Optional	string	The profile of the users to retrieve.
LastName	Optional	string	The last name of the users to retrieve.
FirstName	Optional	string	The first name of the users to retrieve.
MiddleName	Optional	string	The middle name of the users to retrieve.
Email	Optional	string	The e-mail address of the users.
Company	Optional	string	The company to which the users belong.

Parameter	Use	Type	Description
DisplayName	Optional	string	The display name of the users.
Secretary	Optional	string	The secretary of the users.
Assistant	Optional	string	The assistant of the users.
Manager	Optional	string	The manager of the users.
Department	Optional	string	The department to which the users belong.
Phone	Optional	string	The telephone number of the users.
Fax	Optional	string	The fax number of the users.
Gender	Optional	string	Whether the user is male or female.
Timezone	Optional	string	The time zone in which the users are located.
PreferredLanguage	Optional	string	The preferred language of the user.

Role Members

Use this verb to retrieve the users associated with a business process role.

Parameter	Use	Type	Description
RoleName	Mandatory	string	Role name of the users to retrieve.
IncludeNestedRoles	Mandatory	boolean	Specifies whether nested roles are considered in the query.
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	An additional role name for the user.
AlternativeRoleName2	Optional	string	An additional 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	Description
Name	Mandatory	string	The name of the user to retrieve.
AlternativeName1	Optional	string	An additional user name. Use this parameter to retrieve more than one user.
AlternativeName2	Optional	string	An additional 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. Use short names to specify values, for example, wpsadmin. This verb does not imply access to a staff repository.

Parameter	Use	Type	Description
UserID	Mandatory	string	The user ID of the user to retrieve. Supports context variables and custom properties, such as %htm:task.potentialStarters%

Parameter	Use	Type	Description
AlternativeID1	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
AlternativeID2	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.

Users by user ID without Named Users

Use this verb to define a staff query for users whose user ID is known, while excluding explicitly named user IDs. Use short names to specify values, for example, wpsadmin. This verb does not imply access to a staff repository.

Parameter	Use	Type	Description
UserID	Mandatory	string	The user ID of the user to retrieve. Supports context variables and custom properties, such as %htm:task.potentialStarters%
AlternativeID1	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
AlternativeID2	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
NamedUsers	Mandatory	string	The user IDs of the users to exclude from the user ID list. Supports context variables and custom properties, such as %wf:activity(...).owner%

Predefined staff verbs for the system user repository:

Describes the predefined staff verbs and parameters for use with the system user repository staff plug-in for Business Process Choreographer.

You can use staff verbs in WebSphere Integration Developer to model staff assignments in a business process or human task. 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:

- Group
- Everybody
- Nobody
- Users
- Users by user ID
- Users by user ID without Named Users

Group

Use this verb to define a query to authorize the members of the group.

Parameter	Use	Type	Description
GroupId	Mandatory	string	The name of the group of users to authorize.

Everybody

Use this verb to assign a work item to every user authenticated by WebSphere Process Server. This verb has no parameters.

Nobody

For inline tasks, only the business process administrators have access. For standalone tasks, only the human task administrators have access. In addition, when using the Business Flow Manager API the BPESystemAdministrator role members have access, for the Human Task Manager API the TaskSystemAdministrator role members have access. This verb has no parameters.

Users

Use this verb to define a staff query for a user who is known by name.

Parameter	Use	Type	Description
Name	Mandatory	string	The name of the user to retrieve.
AlternativeName1	Optional	string	An additional user name. Use this parameter to retrieve more than one user.
AlternativeName2	Optional	string	An additional 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. Use short names to specify values, for example, wpsadmin. This verb does not imply access to a staff repository.

Parameter	Use	Type	Description
UserID	Mandatory	string	The user ID of the user to retrieve. Supports context variables and custom properties, such as %htm:task.potentialStarters%
AlternativeID1	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
AlternativeID2	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.

Users by user ID without Named Users

Use this verb to define a staff query for users whose user ID is known, while excluding explicitly named user IDs. Use short names to specify values, for example, wpsadmin. This verb does not imply access to a staff repository.

Parameter	Use	Type	Description
UserID	Mandatory	string	The user ID of the user to retrieve. Supports context variables and custom properties, such as %htm:task.potentialStarters%
AlternativeID1	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.

Parameter	Use	Type	Description
AlternativeID2	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
NamedUsers	Mandatory	string	The user IDs of the users to exclude from the user ID list. Supports context variables and custom properties, such as %wf:activity(...).owner%

Predefined staff verbs for the user registry:

Describes the predefined staff verbs and parameters for use with the user registry staff plug-in for Business Process Choreographer.

You can use staff verbs in WebSphere Integration Developer to model staff assignments in a business process or human task. 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:

- Group
- Everybody
- Group Members
- Group Members without Named Users
- Group Search
- Native Query
- Nobody
- Person Search
- Users
- Users by user ID
- Users by user ID without Named Users

Group

Use this verb to define a query to retrieve name of a group for use with group work items.

Parameter	Use	Type	Description
GroupName	Mandatory	string	The name of the group to retrieve.

Everybody

Use this verb to assign a work item to every user authenticated by WebSphere Process Server. This verb has no parameters.

Group Members

Use this verb to define a query to retrieve the members of a group.

Parameter	Use	Type	Description
GroupName	Mandatory	string	Group name of the users to retrieve.
AlternativeGroupName1	Optional	string	An additional group to which the users can belong.
AlternativeGroupName2	Optional	string	An additional group to which the users can belong.

Group Members without Named Users

Use this verb to define a query to retrieve all of the members of a group except for the explicitly named users.

Parameter	Use	Type	Description
GroupName	Mandatory	string	Group name of the users to retrieve. Supports custom properties that are evaluated at run time.
NamedUsers	Mandatory	string	The user IDs of the users to exclude from the retrieved group members list. Supports context variables and custom properties, such as %htm:task.originator%

Group Search

Use this verb to search for a group based on an attribute match and to retrieve the members of the group. You must set one attribute. If you set more than one attribute, only the first attribute is evaluated.

Parameter	Use	Type	Description
GroupID	Optional	string	The group ID of the users to retrieve.

Native Query

Use this verb to define a native query based on directory-specific parameters.

Parameter	Use	Type	Description
QueryTemplate	Mandatory	string	The query template to use for the query. The default mapping files support the templates search, user, and usersOfGroup.
Query	Mandatory	string	Specifies the query. You can use context variables and custom properties, such as %wf:process.starter%. The type of query depends on the query template. <ul style="list-style-type: none">• search template: search pattern• user template: user name• usersOfGroup: group name
AdditionalParameter1	Optional	string	Specifies the query. You can use context variables, such as %wf:process.starter%. Supported values are group and user.
AdditionalParameter2	Optional	string	Use this verb to specify an additional parameter.
AdditionalParameter3	Optional	string	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.
AdditionalParameter4	Optional	string	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.

Parameter	Use	Type	Description
AdditionalParameter5	Optional	string	Use this verb to specify an additional parameter. If you use the default mapping XSLT files, this parameter is not supported.

Nobody

For inline tasks, only the business process administrators have access. For standalone tasks, only the human task administrators have access. In addition, when using the Business Flow Manager API the BPESystemAdministrator role members have access, for the Human Task Manager API the TaskSystemAdministrator role members have access. This verb has no parameters.

Person Search

Use this verb to search for people based on an attribute match. You must set one attribute. If you set more than one attribute, only the first attribute is evaluated.

Parameter	Use	Type	Description
UserID	Optional	string	The user ID of the users to retrieve.

Users

Use this verb to define a staff query for a user who is known by name.

Parameter	Use	Type	Description
Name	Mandatory	string	The name of the user to retrieve.
AlternativeName1	Optional	string	An additional user name. Use this parameter to retrieve more than one user.
AlternativeName2	Optional	string	An additional 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. Use short names to specify values, for example, wpsadmin. This verb does not imply access to a staff repository.

Parameter	Use	Type	Description
UserID	Mandatory	string	The user ID of the user to retrieve. Supports context variables and custom properties, such as %htm:task.potentialStarters%
AlternativeID1	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
AlternativeID2	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.

Users by user ID without Named Users

Use this verb to define a staff query for users whose user ID is known, while excluding explicitly named user IDs. Use short names to specify values, for example, `wpsadmin`. This verb does not imply access to a staff repository.

Parameter	Use	Type	Description
UserID	Mandatory	string	The user ID of the user to retrieve. Supports context variables and custom properties, such as <code>%htm:task.potentialStarters%</code>
AlternativeID1	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
AlternativeID2	Optional	string	An additional user ID. Use this parameter to retrieve more than one user.
NamedUsers	Mandatory	string	The user IDs of the users to exclude from the user ID list. Supports context variables and custom properties, such as <code>%wf:activity(...).owner%</code>

Implementing new custom verbs

This describes how to add new staff verbs the staff support service infrastructure so that they can be used in WebSphere Integration Developer when modeling business processes and human tasks.

You must add the new staff verb specification to the `VerbSet.xml` file which is part of your WebSphere Integration Developer installation. For example, for a new verb `Mentor of Employee`:

```
<vs:DefineVerb name='Mentor of Employee'>
  <vs:Description>Assigns the mentor of an employee.
  Supported by sample XSLT files for:
  - LDAP
  </vs:Description>
  <vs:Mandatory>
    <vs:Parameter>
      <vs:Name>EmployeeName</vs:Name>
      <vs:Type>xsd:string</vs:Type>
    </vs:Parameter>
  </vs:Mandatory>
  <vs:Optional>
    <vs:Parameter>
      <vs:Name>Domain</vs:Name>
      <vs:Type>xsd:string</vs:Type>
    </vs:Parameter>
  </vs:Optional>
</vs:DefineVerb>
```

You must add the new verb to the dispatcher section of the LDAP transformation file. For example:

```
<xsl:choose>
  ...
  <xsl:when test="$verb='Mentor of Employee'">
    <xsl:call-template name="MentorOfEmployee"/>
  ...
</xsl:choose>
```

You must also add to the LDAP transformation file a template that implements the mapping. For example::

```

<!-- Begin template MentorOfEmployee -->
<xsl:template name="MentorOfEmployee">
  <ldap:staffQueries>
    <xsl:attribute name="threshold">
      <xsl:value-of select="$Threshold"/>
    </xsl:attribute>

    <ldap:intermediateResult>
      <xsl:attribute name="name">mentorvariable</xsl:attribute>
      <ldap:user>
        <xsl:attribute name="dn">
          <xsl:value-of select="staff:parameter[@id='EmployeeName']"/>
        </xsl:attribute>
        <xsl:attribute name="attribute">mentor</xsl:attribute>
        <xsl:attribute name="objectclass">inetOrgPerson</xsl:attribute>
      </ldap:user>
    </ldap:intermediateResult>

    <ldap:user>
      <xsl:attribute name="dn">%mentorvariable%</xsl:attribute>
      <xsl:attribute name="attribute">uid</xsl:attribute>
      <xsl:attribute name="objectclass">inetOrgPerson</xsl:attribute>
    </ldap:user>
  </ldap:staffQueries>
</xsl:template>
<!-- End template MentorOfEmployee -->

```

Verify that the mapping generates a valid LDAP specific query.

Adapting the LDAP transformation file

Describes how to adapt the LDAP transformation XSL file to suit your LDAP schema.

The default LDAPTransformation.xsl file maps predefined staff verbs to LDAP queries, which make use of elements of the default LDAP schema assumed by WebSphere. This schema assumes the following:

- The LDAP object class for group entries is groupOfName.
- The group entry attribute containing the member DNs for the group is member.
- The LDAP object class for person entries is inetOrgPerson.
- The attribute containing the login ID in a person entry is uid.
- The person entry attribute containing the e-mail address of a person is mail.
- The person entry attribute containing the distinguished name of the manager of a person is manager.

If your LDAP schema features different object class and attribute names, you must change these settings in the LDAP transformation files that you use. For the default LDAPTransformation.xsl file, changing the setting can be done in the variable declaration part of the file:

```

<xsl:variable name="DefaultGroupClass">groupOfNames</xsl:variable>
<xsl:variable name="DefaultGroupClassMemberAttribute">member</xsl:variable>

<xsl:variable name="DefaultPersonClass">inetOrgPerson</xsl:variable>
<xsl:variable name="DefaultUserIDAttribute">uid</xsl:variable>
<xsl:variable name="DefaultMailAttribute">mail</xsl:variable>
<xsl:variable name="DefaultManagerAttribute">manager</xsl:variable>

```

You can apply changes within the XSL templates that transform the individual staff verbs, as illustrated in the following examples.

Example: DepartmentMembers

Changing the object class for person entries to ePerson and the login ID attribute to cn:

```
<slldap:StaffQueries>
  <xsl:attribute name="threshold">
    <xsl:value-of select="$Threshold">
  </xsl:attribute>

  <slldap:search>
  ...
  <slldap:attribute>
    <xsl:attribute name="name">cn</xsl:attribute>
    <xsl:attribute name="objectclass">ePerson</xsl:attribute>
    <xsl:attribute name="usage">simple</xsl:attribute>
  </slldap:attribute>

  </slldap:search>
</slldap:StaffQueries>
```

Example: GroupMembers

Changing the object class for group entries to groupOfUniqueNames, the group entry attribute containing the member DN list to uniqueMember, and the person entry attribute containing the login in to cn:

```
<slldap:usersOfGroup>
  ...
  <slldap:attribute>
    <xsl:attribute name="name">uniqueMember</xsl:attribute>
    <xsl:attribute name="objectclass">groupOfUniqueNames</xsl:attribute>
    <xsl:attribute name="usage">recursive</xsl:attribute>
  </slldap:attribute>

  ...
  <slldap:attribute>
    <xsl:attribute name="name">cn</xsl:attribute>
    <xsl:attribute name="objectclass">inetOrgPerson</xsl:attribute>
    <xsl:attribute name="usage">simple</xsl:attribute>
  </slldap:attribute>

</slldap:usersOfGroup>
```

Example: GroupMembersWithoutFilteredUsers

Changing the LDAP filter operator to >=.

```
<slldap:StaffQueries>
  <slldap:usersOfGroup>
  ...
</slldap:usersOfGroup>

<slldap:intermediateResult>
  <xsl:attribute name="name">filteredusers</xsl:attribute>
  <slldap:search>
    <xsl:attribute name="filter">
      <xsl:value-of select="staff:parameter[@id='FilterAttribute']"/>
      >=
      <xsl:value-of select="staff:parameter[@id='FilterValue']"/>
    </xsl:attribute>
    ...
  </slldap:search>
  ...
```

```

    </sldap:intermediateResult>
    ...
</sldap:StaffQueries>

```

Example: GroupSearch

Changing the search attribute to MyType, the object class to mypersonclass, and the attribute containing the login ID to myuid.

```

<sldap:StaffQueries>
...
<sldap:search>
  <xsl:attribute name="filter">
    (&
      ...
      <xsl:if test="staff:parameter[@id='MyType']!="">
        (<xsl:value-of select="$GS_Type"/>=
          <xsl:value-of select=staff:parameter[@id='Type']"/>)
      </xsl:if>
    )
    ...
  </xsl:attribute>

  <sldap:attribute>
    <xsl:attribute name="name">myuid</xsl:attribute>
    <xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
    <xsl:attribute name="usage">simple</xsl:attribute>
  </sldap:attribute>
  ...
</sldap:search>
</sldap:StaffQueries>

```

Example: Manager of Employee

Changing the attribute containing the manager DN to managerentry and the source of the manager login ID attribute to name.

```

<sldap:StaffQueries>

  <sldap:intermediateResult>
    ...
    <sldap:user>
      ...
      <xsl:attribute name="name">managerentry</xsl:attribute>
      ...
    </sldap:user>
  </sldap:intermediateResult>

  <sldap:user>
    ...
    <xsl:attribute name="name">name</xsl:attribute>
    ...
  </sldap:user>
</sldap:StaffQueries>

```

Example: PersonSearch

Changing the search attribute to MyAttribute, the object class to mypersonclass, and the source of the return attribute to myuid.

```

<sldap:StaffQueries>
...
<sldap:search>
  <xsl:attribute name="filter">
    (&

```

```

...
<xsl:if test="staff:parameter[@id='MyAttribute']!="">
  (<xsl:value-of select="$PS_UserID"/>=
   <xsl:value-of select=staff:parameter[@id='UserID']"/>)
)
</xsl:if>
...
</xsl:attribute>

<ldap:attribute>
  <xsl:attribute name="name">myuid</xsl:attribute>
  <xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
  <xsl:attribute name="usage">simple</xsl:attribute>
</ldap:attribute>
...
</ldap:search>
</ldap:StaffQueries>

```

Example: Users

Changing the source of the return attribute to myuid and the object class to mypersonclass.

```

<ldap:user>
...
  <xsl:attribute name="attribute">myuid</xsl:attribute>
  <xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
</ldap:user>

```

Overview: Configuring Business Process Choreographer Explorer

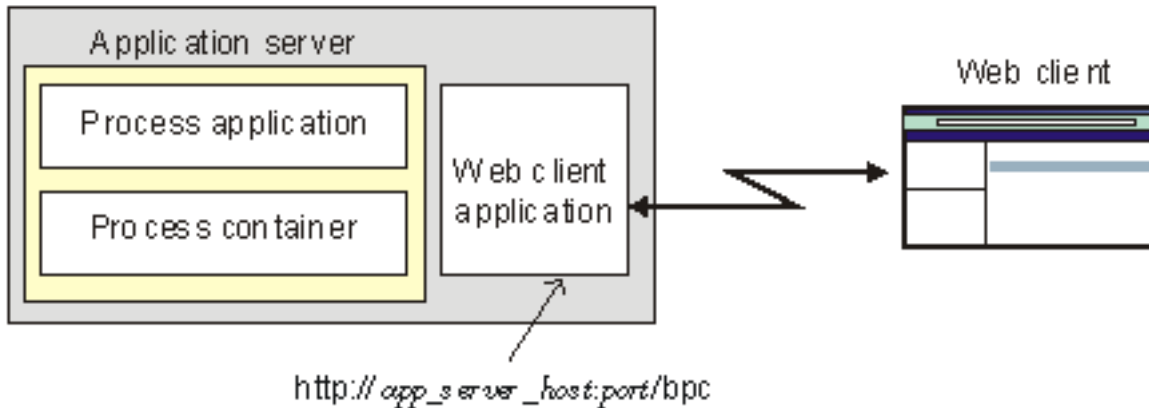
Business Process Choreographer Explorer provides a user interface for administering processes and handling tasks. It is a Java 2 Enterprise Edition (J2EE) Web application, based on the JavaServer Faces (JSF) technology and the Business Process Choreographer Explorer components.

- “About Business Process Choreographer Explorer”
- “Configuring Business Process Choreographer Explorer” on page 82

About Business Process Choreographer Explorer

Business Process Choreographer Explorer is a Web application that implements a generic Web user interface for interacting with business processes and human tasks.

You can install Business Process Choreographer Explorer on the application server, or on the cluster, where you installed both a business process container and a human task container. If you want to work with business process applications or human task applications on several application servers or clusters, it is sufficient to configure Business Process Choreographer Explorer on one server or cluster; you do not need to configure it on all application servers or clusters that have Business Process Choreographer configured on them.



A single Business Process Choreographer Explorer can not connect to multiple Business Process Choreographer configurations. You can configure multiple instances of the Business Process Choreographer Explorer on the same server.

When you start Business Process Choreographer Explorer, the objects that you see in the user interface and the actions that you can take depend on the user group that you belong to and the authorization granted to that group. For example, if you are an administrator, you are responsible for the smooth operation of deployed business processes and tasks. You can view information about process and task templates, process instances, task instances, and their associated objects. You can also act on these objects; for example, you can start new process instances, create and start tasks, repair and restart failed activities, manage work items, and delete completed process instances and task instances. However, if you are a user, you can view and act on only those tasks that have been assigned to you.

Configuring Business Process Choreographer Explorer

Use this task to configure Business Process Choreographer Explorer using a script.

You have configured the business process container and human task container.

One of the following applies:

- You have not yet installed Business Process Choreographer Explorer.
- You want to add it to an existing Business Process Choreographer configuration.
- You want to add another instance of Business Process Choreographer Explorer to a server where other instances are already running.

1. Change to the Business Process Choreographer directory and invoke the `clientconfig.jacl` script.

```
cd install_root/ProcessChoreographer/config
../../bin/wsadmin.sh -f clientconfig.jacl
( [-user userName][-password password] | [-conntype NONE] )
  [-profileName profileName]
  [-hostName explorerVirtualHostname]
  [-explorerHost explorerURL]
  [-precompileJSPs { yes | no }]
  [-remoteNodeName nodeName]
  [-remoteServerName serverName]
  [-remoteClusterName clusterName]
  [-contextRootExplorer explorerContextRoot]
```

On z/OS, run:

```

cd install_root/ProcessChoreographer/config
../bin/wsadmin.sh -f clientconfig.jacl
( [-user userName][-password password] | [-conntype NONE] )
  [-profileName profileName]
  [-hostName explorerVirtualHostname]
  [-explorerHost explorerURL]
  [-precompileJSPs { yes | no }]
  [-remoteNodeName nodeName]
  [-remoteServerName serverName]
  [-remoteClusterName clusterName]
  [-contextRootExplorer explorerContextRoot]

```

In a standalone server environment:

- Include the `-conntype NONE` option only if the application server is not running.
- If the server is running and global security is enabled, include the `-user` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

In a Network Deployment environment:

- Run the script on the deployment manager node.
- Include the `-conntype NONE` option only if the deployment manager is not running.
- If global security is enabled, include the `-user` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

You can also provide the following parameters:

node *nodeName*

Where *nodeName* is the name of the node where Business Process Choreographer is to be configured. If you have only one node and exactly one server, this parameter is optional.

server *serverName*

Where *serverName* is the name of the server where Business Process Choreographer is to be configured. If you have only one node and exactly one server, this parameter is optional.

cluster *clusterName*

Where *clusterName* is the name of the cluster where Business Process Choreographer is to be configured. Do not specify this option in a standalone server environment, nor if you specify the node and server. This option cannot be used non-interactively.

hostName *explorerVirtualHostname*

Where *explorerVirtualHostname* is the virtual host where the Business Process Choreographer Explorer will run. The default value is `default_host`.

explorerHost *explorerURL*

Where *explorerURL* is the URL of the Business Process Choreographer Explorer. If this parameter is not specified for a non-cluster environments, a default value is computed, for example, `http://localhost:9080`. The value of this parameter is used for the `EscalationMail.ClientDetailURL` custom property of the Human Task Manager.

precompileJSPs { *yes* | *no* }

Determines whether Java Server Pages (JSPs) will be precompiled, or not.

remoteNodeName *nodeName*

Use this parameter and `remoteServerName` if you do not want to connect

to the local Business Process Choreographer Explorer. Do not specify this parameter if you want to connect to the Business Process Choreographer server identified by the node and server parameters or the cluster parameter.

remoteServerName *serverName*

Use this parameter and remoteNodeName if you do not want to connect to the local Business Process Choreographer Explorer. Do not specify this parameter if you want to connect to the Business Process Choreographer server identified by the node and server parameters or the cluster parameter.

remoteClusterName *clusterName*

Use this parameter, if you do not want to connect to the local Business Process Choreographer Explorer and you do not specify remoteNodeName and remoteServerName. Do not specify this parameter if you want to connect to the Business Process Choreographer server identified by the node and server parameters or the cluster parameter.

contextRootExplorer *contextRootExplorer*

Where *contextRootExplorer* is the context root for the Business Process Choreographer Explorer. The default value is /bpc, which results in the default URL of `http://host:port/contextRootExplorer`. The context root must be unique within the WebSphere cell.

2. The `clientconfig.jacl` script prompts you for any required information that was not provided as parameters.
3. **Optional:** If you have problems with the configuration, check the log file written by the `clientconfig.jacl` script. This log is located in the `profiles/profileName/logs/clientconfig.log` file. This directory also contains a `wsadmin.traceout` file that might contain more information about the problem.

Business Process Choreographer Explorer is configured and ready to use.

Start Business Process Choreographer Explorer.

Configuring the Business Process Choreographer Observer infrastructure

Describes how to configure the Business Process Choreographer Observer and event collector, which allow you to observe the states of running business processes and human tasks.

You have configured Business Process Choreographer using the administrative console or the `bpeconfig` script, and have not configured the Business Process Choreographer Observer. The `WAS_HOME` environment variable must be set to the installation directory of your application server. You must be logged on using a user ID that has administrative rights, for example, `root`.

This topic does not describe how to configure the Business Process Choreographer Observer in an ND environment nor advanced configurations. For more information about configuring and using the Business Process Choreographer Observer, refer to support document 7008553.

1. Configure the event collector application and database. Perform “Configuring the Business Process Choreographer event collector” on page 86.

2. Configure the observer application. Perform “Configuring Business Process Choreographer Observer” on page 89.

The Business Process Choreographer Observer is installed and configured.

Verify that Business Process Choreographer works.

About Business Process Choreographer Observer

You can use Business Process Choreographer Observer to create reports on processes and tasks that have been completed. You can also use it to view the status of running processes and tasks.

For detailed information about configuring and using Business Process Choreographer Observer, including in an ND environment, refer to support document 7008553.

You can configure Business Process Choreographer Observer in several ways:

- If you have an existing Business Process Choreographer configuration you can use the configuration scripts, as described in “Configuring the Business Process Choreographer Observer infrastructure” on page 84.
- If you use the Profile Creation wizard, and select the option to create a sample Business Process Choreographer configuration, Business Process Choreographer Observer and event collector are configured, Common Event Infrastructure (CEI) logging if enabled for the Business Process Choreographer state observer, and the necessary database schema is created within the Business Process Choreographer Cloudscape database, names BPEDB.
- If you use the administrative console to run the Business Process Choreographer install wizard, there is an option to configure Business Process Choreographer Observer.

Business Process Choreographer Observer is based on two J2EE enterprise applications:

- Business Process Choreographer event collector
- Business Process Choreographer Observer

They use the same database. The event collector reads event information from the CEI bus and stores it in the Business Process Choreographer Observer database. Periodically the raw event data is transformed into a format suitable for queries from the Business Process Choreographer Observer. If Business Process Choreographer uses Cloudscape, DB2, or Oracle, Business Process Choreographer Observer can use the same database instance, otherwise you must create a new database.

The Business Process Choreographer Observer database is a set of databases tables that store the event data. The tables can be created in an existing database or in a new database dedicated to the Business Process Choreographer Observer. Before you install Business Process Choreographer Observer, the database must be available and a XA data source must be configured in your application server to point to the database. You must create the data source manually. To create the tables for the observer you can either use the observer configuration scripts or you can create the schema manually using the scripts located *install_root\dbscripts\ProcessChoreographer\database_type*

If your database system is DB2 or Oracle you have to run the `createTablespace_observer.sql` script before using the configuration tools.

Configuring the Business Process Choreographer event collector

This describes how to use a script to configure the event collector and database tables that are necessary for the Business Process Choreographer Observer.

- A database is available.
- The CLASSPATH environment variable contains the JDBC driver for your database.
- You have to create an XA data source manually using the application server's administrative console. The data source must point to the database.

Note: For z/OS, when using a type 2 non-XA Provider, do not create an XA data source. The XA data source is for Type 4 XA Provider only.

1. Change to the Business Process Choreographer subdirectory where the configuration scripts are located.

Type the following command:

```
cd install_root/ProcessChoreographer/config
```

2. Start the script to set up the event collector.

Type the following command:

```
setupEventCollector.sh [-conntype SOAP | RMI | JMS | NONE]  
  ( [-node nodeName] -server serverName ) | ( -cluster clusterName )  
  [ -remove [-silent]]
```

Where:

conntype SOAP | RMI | JMS | NONE

The connection mode that wsadmin tool uses.

node *nodeName*

The name of the node. This parameter is optional. The default value is the local node.

server *serverName*

The name of the server. If you do not specify the option -conntype none, this parameter is optional.

cluster *clusterName*

clusterName If you do not specify the option -conntype none, this parameter is optional.

remove

Specify this option to remove the event collector. If you do not specify this option, the default is that the event collector will be configured.

silent This option can only be used with the remove option. It causes the script to not output any prompts. This parameter is optional.

Note:

In a stand-alone server environment:

- Include the -conntype NONE option only if the application server is not running.
- If the server is running and global security is enabled, include the -username and -password options.
- If you are not configuring the default profile, add the -profileName option.

You see the Commands Menu:

Commands Menu

- 1) Prepare a database for the Event Collector
 - 2) Install the Event Collector application
 - 3) Remove the Event Collector application and related objects
 - 4) Change configuration settings of an installed Event Collector
- 0) Exit Commands Menu

Note: If you started the script in local mode, that is, with the `-conntype none` parameter and without specifying a server or cluster, only menu items 0 and 1 are displayed.

3. Select option 1 to prepare a database for the event collector.

a. When you see the commands menu:

```
Select the type of your DBMS :
'd' ... DB2
'c' ... Cloudscape
'7' ... DB2 V7 on z/OS
'8' ... DB2 V8 on z/OS
'o' ... Oracle
'x' ... Exit
Your selection : [c]
```

You see the Commands Menu:

4. Select your database type.

- For DB2, enter d.
- For Cloudscape, enter c.
- For DB2 V7 on z/OS, enter 7.
- For DB2 V8 on z/OS, enter 8.
- For Oracle, enter o.

5. Enter the database settings.

- For DB2, enter the following:
 - a. Either the database name, or an alias, for example BPEDB.
 - b. The user ID and password to connect to the database, for example, db2admin.
 - c. The password for user ID.
 - d. The database schema to be used for the database objects. If you specify a schema that does not exist, it is created. If you enter a space character or leave the field empty, the schema of the user ID specified in a. is used.
- For Cloudscape, enter the following:
 - a. The fully qualified path to the database, for example, d:\w\p\profiles\Srv01\databases\BPEDB.
 - b. The database schema to be used for the database objects. If you specify a schema that does not exist, it is created. If you enter a space character or leave the field empty, the default schema is used (normally APP) .
 - c. If you are prompted to stop the server, do so, then press c to continue.
- For Oracle, enter the following:
 - a. The database name, for example, BPEDB.
 - b. The host name where the database resides, for example, localhost.
 - c. The port number where the Oracle listener is listening, for example, 1521.
 - d. The user ID to connect to the database, for example, system.

- e. The password for the user ID.

After checking the connection, the database is prepared.

6. Check for any errors. If any errors occur, check the log file `setupEventCollector.log` that is located in the logs subdirectory of the profile directory. For example, on Windows, if your profile is named *myServer* and your profiles are stored in *install_root\profiles*, the log file is located in *install_root\profiles\myServer\logs*.
7. Install the event collector application. When you see the Commands Menu:
Commands Menu
 - 1) Prepare a database for the Event Collector
 - 2) Install the Event Collector application
 - 3) Remove the Event Collector application and related objects
 - 4) Change configuration settings of an installed Event Collector
- 0) Exit Commands Menu

Select option 2 to install the Business Process Choreographer event collector application. You see the JNDI name prompt:

Specify the JNDI name of the database where the WebSphere BPC Event Collector should store the collected events.

Enter '?' to get a list.

Your selection : [jdbc/BPEDB]

8. Enter the JNDI name that is used to connect to the database. You can also enter ? to get a list of all registered data sources.
9. Enter the name of the schema for the database tables in which the collected events will be stored. To use the schema specified in the data source definition, enter a space character or leave the field empty.
10. Enter the JMS user ID to authenticate with the Common Event Infrastructure (CEI) bus. If the CEI bus has security disabled, you can leave this empty. If you specify a user ID, also enter a password at the next prompt. All required objects are created and the enterprise application is installed. Success is indicated by the message:
WebSphere Business Process Choreographer Event Collector
installed successfully!
11. If there were no error messages, enter *y* to save the configuration. Otherwise, enter *n* to discard the changes and keep your original configuration. If there were errors, check the log file named `setupEventCollector.log`, which is located in the logs directory of the profile, for example, on Windows, if your profile is named *myServer* and your profiles are stored in *install_root\profiles*, the log file is located in *install_root\profiles\myServer\logs*.
12. If CEI logging is not enabled on the server, you see the following:
Checking if CEI event logging is enabled ...

Warning: The Business process container of *server_name* has CEI event logging disabled.
To allow the Event Collector to work correctly, CEI event logging is required.
Do you want to enable the CEI event logging on *server_name*? (y/n)

To enable CEI logging, enter *y*, otherwise enter *n*.
13. If you are prompted to start the application, enter *y* to start the application or *n* to not start it.
14. To activate all settings, restart the server.

The Business Process Choreographer event collector is installed and configured.

Continue configuring at step 2 on page 85. If you want to, you can use option 4 in the command menu to change configuration parameters for the event collector, this is described in support document 7008553.

Configuring Business Process Choreographer Observer

This describes how to use a script to configure Business Process Choreographer Observer, which allows you to observe the states of running business processes and human tasks.

- You have configured Business Process Choreographer using the administrative console or the `bpeconfig` script.
 - You have configured the Business Process Choreographer event collector but have not yet configured Business Process Choreographer Observer.
 - You have enabled the Common Event Infrastructure (CEI) logging for the container where the business applications run that you want to observe.
 - The server must be running.
1. Change to the Business Process Choreographer subdirectory where the configuration scripts are located.

Type the following command:

```
cd install_root/ProcessChoreographer/config
```

2. Start the script to set up the event consumer.

Type the following command:

```
setupObserver.sh [-conntype SOAP | RMI | JMS | NONE]
                 ( [-node nodeName] -server serverName ) | ( -cluster clusterName )
                 [ -remove [-silent]]
```

Where:

conntype *SOAP | RMI | JMS | NONE*

The connection mode that wsadmin tool uses.

node *nodeName*

The name of the node. This parameter is optional. The default value is the local node.

server *serverName*

The name of the server. If you do not specify the option `-conntype none`, this parameter is optional.

cluster *clusterName*

clusterName If you do not specify the option `-conntype none`, this parameter is optional.

remove

Specify this option to remove the Business Process Choreographer event collector. If you do not specify this option, the default is that the Business Process Choreographer event collector will be configured.

silent This option can only be used with the `remove` option. It causes the script to not output any prompts. This parameter is optional.

Note:

In a stand-alone server environment:

- Include the `-conntype NONE` option only if the application server is not running.

- If the server is running and global security is enabled, include the `-username` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

You see the Commands Menu:

Commands Menu

Working on node '*nodeName*', server '*serverName*'.

- 1) Install the Observer application
- 2) Remove the Observer application and related objects
- 3) Change configuration settings of an installed Observer

0) Exit Menu

3. Select option 1 to install Business Process Choreographer Observer. You see the JNDI name prompt:

Specify the JNDI name of the database containing the event tables.

Enter '?' to get a list.

Your selection : [jdbc/BPEDB]

4. Enter the JNDI name for the data source that is used by the Business Process Choreographer event collector. You can also enter ? to get a list of all registered data sources.
5. Enter the name of the schema for the database tables in which the collected events are stored. If you leave this field empty or enter a space character, the default schema is used, which is the schema of the user ID that is specified in the properties of the data source that you specified in step 4. All required objects are created and the enterprise application is installed. Success is indicated by the message:
WebSphere BPC Observer installed successfully!
6. If there were no error messages, enter `y` to save the configuration. Otherwise, enter `n` to discard the changes and keep your original configuration. If there were errors, check the log file named `setupObserver.log`, which is located in the logs directory of the profile, for example, in `WebSphere/V6R0M0/AppServer/profiles/default/logs`.
7. If you are prompted, enter `y` to start the application or `n` to not start it.
8. To activate all settings, restart the server.

Business Process Choreographer Observer is installed and configured. It can be reached using the URL `http://host:port/bpcobserver/`. Where *host* is the name of the host where your application server is running, and *port* is the port number for your application server (the default is 9080).

Continue configuring at step 8 on page 13. If you want to, you can use option 3 in the command menu to change configuration parameters for the Business Process Choreographer Observer, this is described in support document 7008553.

Activating Business Process Choreographer

After configuring the business process container and human task container, you must restart the servers where they were installed.

To activate Business Process Choreographer:

1. If you installed the business process container and human task container on a cluster of application servers, restart the cluster.

2. If you installed the business process container and human task container on one application server, restart the application server.
3. To verify that the business process container and human task container applications started successfully, make sure that no error messages exist in the `SystemOut.log` file for the application server. On a cluster, check the log for all application servers in the cluster.

Business Process Choreographer is running.

You are ready to verify that Business Process Choreographer is working.

Verifying that Business Process Choreographer works

Run the Business Process Choreographer installation verification application.

The application server, database system, and messaging service must be running.

1. Using either the administrative console or the `wsadmin` command, install the application in `install_root/installableApps/bpcivt.ear`. Errors will occur at this stage if the Business Process Choreographer database cannot be accessed. These problems can be caused if the database system is not running, if any database clients are not correctly configured, or if errors were made defining the data source, for example, entering an invalid user ID or password. After the enterprise application is installed, it is in the state stopped, and any process and task templates that it contains are in the state started. No process or task instances can be created until the application is started.
2. Select the application `BPCIVTApp` and click **Start** to start the application. At this point, the input queues are read for the first time. Errors will occur at this stage if the queue manager is not running, or if any mistakes were made defining the JMS provider or JMS resources.
3. Verify that the application works. Using a Web browser, open the following page:
`http://app_server_host:port_no/bpcivt`

Where `app_server_host` is the network name for the host of the application server and `port_no` is the port number used by the virtual host to which you mapped the IVT Web module when installing the file `bpcivt.ear`. The port number depends on your system configuration. You should see a message indicating success.

4. **Optional:** Stop and remove the `BPCIVTApp` application.

Business Process Choreographer works.

Understanding the startup behavior of the business process container

This topic explains why the business process container is unavailable until all enterprise applications are started.

When the business process container is started or restarted, no messages in the internal queue are processed until all enterprise applications are started. It is not possible to change this behavior. The time that business process container is unavailable during a restart depends on how long it takes until all enterprise applications are started. This behavior is necessary to prevent the business process engine from navigating processes with associated enterprise applications that are not running.

Starting to process messages in the internal queue before all applications are started would result in `ClassNotFoundException` exceptions.

Configuring Business Process Choreographer to use an LDAP user registry

This describes how to change an existing Business Process Choreographer configuration to use a Lightweight Directory Access Protocol (LDAP) user registry.

1. Configure the LDAP user registry.
 - a. Click **Security** → **Global security** make sure that the **Enable global security** is not enabled.
 - b. Under **User registries**, select **LDAP**.
 - c. Set the user name and password used to run WebSphere Process Server for security purposes. In the **Server user ID** field type the user name, and in the **Server user password** field, enter the corresponding password. This ID is not the LDAP administrator user ID. This user ID must exist in the LDAP registry.
 - d. From the **Type** list choose the specific LDAP that you want to use as your user registry.
 - e. In the **Host** field, enter the host name of the LDAP server.
 - f. In the **Port** field, enter the port number on which the LDAP server is listening.
 - g. Enter the **Base Distinguished Name**.

This value specifies the base distinguished name of the directory service, indicating the starting point for LDAP searches of the directory service. For authorization purposes, this field is case sensitive. This specification implies that if a token is received (for example, from another cell or Domino server) the base distinguished name (DN) in the server must match the base DN from the other cell or Domino server exactly. If case sensitivity is not a consideration for authorization, enable the **Ignore case** field. This field is required for all LDAP directories except for the Domino Directory, where this field is optional.
 - h. Enter the **Bind Distinguished Name**. Enter the user ID that the application server will use to bind to the LDAP server. For example, you can use the same user ID that you entered for the **Server user ID**.
 - i. Enter the **Bind Password**. Enter the password for the user ID you specified for the **Bind Distinguished Name**.
 - j. Leave the remaining parameters with the default values and confirm your changes. Click **OK**.
2. Configure the Lightweight Third Party Authentication (LTPA) mechanism. Under **Authentication**, open **Authentication mechanisms** and select **LTPA**. In the fields **Password** and **Confirm password**, enter a password of your choice, then click **OK**.
3. Enable global security, Java 2 security, LTPA and LDAP.
 - a. Select **Enable global security**.
 - b. Select **Enforce Java 2 security**.
 - c. For **Active authentication mechanism**, select **Lightweight Third Party Authentication (LTPA)**.
 - d. For **Active user registry**, select **Lightweight Directory Access Protocol (LDAP) user registry**.

- e. Click **OK** and save your changes.
4. Restart WebSphere Process Server.
5. Log on to the administrative console using the user ID that you specified for the **Server user ID** in step 1c on page 92.
6. Add new user mapping for the JMSAPIUser role for the business process container application.
 - a. In the administrative console, locate the business process container application. Click **Applications** → **Enterprise Applications** → **BPEContainer_<your_node>_<your_server>**.
 - b. Under **Additional Properties**, click **Map RunAs roles to users**.
 - c. For **username**, enter a valid user ID that is defined in the LDAP user registry.
 - d. For **password**, enter the password for the user ID.
 - e. Select the check box in front of the table row for **JMSAPIUser**.
 - f. Click **Apply**. This associates the user ID with the role. The user ID is added to the table.
 - g. Click **OK** and save your changes.
7. Add security role mappings for system administrator and system monitor for the business process container application.
 - a. In the administrative console, locate the business process container application. Click **Applications** → **Enterprise Applications** → **BPEContainer_<your_node>_<your_server>**.
 - b. Under **Additional Properties**, click **Map security roles to users/group**.
 - c. Select the check boxes in front of the table rows for **BPESystemAdministrator** and **BPESystemMonitor**.
 - d. Click **Lookup Users**
 - e. For **Search String**, enter the character *, and click **Search**.
 - f. In the **Available** list, select the entry for a user or group that will , then click >> .
 - g. Click **OK**.
 - h. Select the check box in front of **BPESystemAdministrator**, if any other check boxes are selected, clear them.
 - i. Click **Lookup Groups**.
 - j. Remove the group, by selecting the group that is displayed in the **Selected** field, click <<, click **OK**.
 - k. Click **OK** and save your changes.
8. Add a new user name mapping for the EscalationUser role for the human task container application.
 - a. In the administrative console, locate the human task container application. Click **Applications** → **Enterprise Applications** → **TaskContainer_<your_node>_<your_server>**.
 - b. Under **Additional Properties**, click **Map RunAs roles to users**.
 - c. For **username**, enter a valid user ID that is defined in the LDAP user registry.
 - d. For **password**, enter the password for the user ID.
 - e. Select the check box in front of the table row for **EscalationUser**.
 - f. Click **Apply**. This associates the user ID with the role. The user ID is added to the table.

- g. Click **OK** and save your changes.
9. Add security role mappings for system administrator and system monitor for the human task container application.
 - a. In the administrative console, locate the business process container application. Click **Applications** → **Enterprise Applications** → **TaskContainer_<your_node>_<your_server>**.
 - b. Under **Additional Properties**, click **Map security roles to users/group**.
 - c. Select the check boxes in front of the table rows for **TaskSystemAdministrator** and **TaskSystemMonitor**.
 - d. Click **Lookup Users**
 - e. For **Search String**, enter the character *, and click **Search**.
 - f. In the **Available** list, select the entry for the user ID that you specified in step 1c on page 92, then click >> .
 - g. Click **OK**.
 - h. Select the check box in front of **TaskSystemAdministrator**, if any other check boxes are selected, clear them.
 - i. Click **Lookup Groups**.
 - j. Remove the group, by selecting the group that is displayed in the **Selected** field, click <<, click **OK**.
 - k. Click **OK** and save your changes.
10. Change the sample LDAP staff plug-in configuration.
 - a. In the administrative console, click **Resources** → **Staff plug-in provider** → **LDAP Staff Plugin Provider**.
 - b. Under **Additional Properties**, click **Staff Plugin Configuration**.
 - c. Click **LDAP Staff Plugin Configuration sample**
 - d. Under **Additional Properties**, click **Custom properties**.
 - e. Set the value of the **BaseDN** property to the same value that you entered for **Base Distinguished Name (DN)** in step 1g on page 92.
 - f. Set the value of the **ProviderURL** to the URL for the LDAP server. For example, `ldap://host:port`, where *host* and *port* are the values that you entered in steps 1e on page 92 and 1f on page 92.
 - g. Click **OK** and save your changes.
11. Change the authentication data entries for the J2EE Connector Architecture (J2C).
 - a. In the administrative console, click **Security** → **Global Security**.
 - b. Under **JAAS Configuration**, click **J2C Authentication data** .
 - c. Change each user alias, and set the user ID and password to values for a valid user ID that is defined in the LDAP user registry. Do not change any database aliases.
 - d. Save your changes.
12. Restart the servers.
 - For ND: Stop the cluster, all node agents, and the deployment manager and restart them.
 - For a single server: Restart the server.

Note: Use the server user ID that you specified in step 1c on page 92 to stop the servers, node agents and deployment manager.
13. Verify that the Business Process Choreographer applications are running.

- a. In the administrative console, click **Applications** → **Enterprise Applications**.
- b. Verify that the applications `BPEContainer_node_server`, and `TaskContainer_node_server` are running.

Now all staff queries will be made against the selected LDAP server.

Federating a stand-alone node that has Business Process Choreographer configured

This describes how to federate a server in a stand-alone profile that runs applications that contain business processes or human tasks or both to a new deployment manager cell.

The deployment manager is running, and you know its host name and port number. Business Process Choreographer is configured on the server in a stand-alone profile. The Business Process Choreographer database in the stand-alone profile must be remotely accessible from the deployment manager cell. For this reason, your server cannot be based on the sample Business Process Choreographer configuration that uses embedded Cloudscape.

You have one or more applications, which contain business processes or human tasks, running on a stand-alone server, and you want to federate this server into a network deployment environment. One of the following applies:

- You want to move an application from a test environment to a production environment.
 - You want to use ND to administer your server and application.
1. If the node includes a large number of applications, increase the timeout for the administrative connector.
 2. From the command line, run the `addNode` command with the `-includeapps` and `-includebuses` options. For details about this command and possible errors that can occur, refer to the WebSphere Application Server for z/OS information center, `addNode` command. For example, if the deployment manager has a host name of `dmgr_host` and uses port `dmgr_port`, enter the command:


```
addNode dmgr_host dmgr_port -includeapps -includebuses
```

For example, if the deployment manager has a host name of `any.hostname.com` and uses port `9043`, your profile name is `ProcSvr07`, your user ID is `admin`, and your password is `secret`, enter the command:

```
addNode any.hostname.com 9043 -profileName ProcSvr07 -username admin
      -password secret -includeapps -includebuses
```

If any of the prerequisites are not met, an error message is displayed. Otherwise, the server is stopped and the server is federated into a new deployment manager cell.

3. Set the JDBC provider's WebSphere variables on the deployment manager node. The variable for the path for the JDBC driver that you are using must be set. That is one of the following:
 - `DB2UNIVERSAL_JDBC_DRIVER_PATH` (for DB2)
 - `UNIVERSAL_JDBC_DRIVER_PATH` (for DB2)
 - `CONNECTJDBC_JDBC_DRIVER_PATH` (for SQL Server)
 - `ORACLE_JDBC_DRIVER_PATH` (for Oracle)
4. Start the server to activate the changes.

5. If you cannot access the business applications that are running on the server, use the administrative console on the deployment manager to make sure that the virtual host and alias definitions for the application server match the new cell.

Your applications are now running on the same server, but the server is now in a cell that can be administered using the deployment manager.

If required, you can promote the server to a cluster.

Related tasks

“Promoting a server that has Business Process Choreographer configured to a cluster”

This describes how to promote a server that runs applications that contain business processes or human tasks or both to a cluster.

Promoting a server that has Business Process Choreographer configured to a cluster

This describes how to promote a server that runs applications that contain business processes or human tasks or both to a cluster.

You have one or more applications, which contain business processes or human tasks, running on a standalone server in an ND cell, and you want to create a cluster from this server. For example, to increase the workload capacity, availability, or to consolidate multiple applications that have their own profiles.

1. Create the cluster. Perform *Creating a cluster, using the existing server that has Business Process Choreographer configured on it*.

Restriction:

- If you have an existing server that has Business Process Choreographer configured on it, it must be the first server in the cluster.
 - If you want to create the cluster using one of the templates provided, use the `defaultProcessServerTemplate`.
2. Activate the changes. If the server is not running start it. If the server is running, stop and start it.

Your applications are now running on the same server, but the server is now a member of the cluster, and the resources have cluster scope.

You can add more members to the cluster to share the workload and increase availability.

Chapter 3. Removing the Business Process Choreographer configuration

Use this task to remove the business process container, human task container, Business Process Choreographer Explorer, and the associated resources.

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 human tasks or business processes, stop and uninstall all human task and business process templates, then uninstall the application.
3. Perform one of the following actions:
 - To uninstall the business process container, human task container, Business Process Choreographer Explorer, and the associated resources, perform “Using a script to remove the Business Process Choreographer configuration.”
 - If you want to reuse parts of the existing configuration, perform “Using the administrative console to remove the Business Process Choreographer configuration” on page 99.

The Business Process Choreographer configuration has been removed.

Using a script to remove the Business Process Choreographer configuration

Use this task to remove the business process container, task container, Business Process Choreographer Explorer and Business Process Choreographer Observer configuration, and the associated resources.

Before you can remove the Business Process Choreographer configuration, you must stop all process and task templates, delete all process and task instances, then stop and remove the configuration for all enterprise applications that contain business processes or human tasks.

1. Change to the Business Process Choreographer sample directory: Type the following:

```
cd install_root/ProcessChoreographer/config
```
2. Run the script `bpeunconfig.jacl`. In the following cases, also specify the appropriate options:
 - For stand-alone servers, stop the application server and use the `-conntype NONE` option. This step ensures that any Cloudscape databases are not locked and can be removed automatically.
 - In a Network Deployment (ND) environment, run the script, as follows:
 - If the deployment manager is not running, run the script on the deployment manager, using the `-conntype NONE` option.
 - If the deployment manager is running, stop the application server from which the configuration is to be removed, then run the script, omitting the `-conntype NONE` option.

When the script is running on the application server node from which the Business Process Choreographer configuration is to be removed, the script can automatically delete any Cloudscape databases.

- If WebSphere security is enabled, specify also the user ID and password:
-userid *userID* -password *password*
- If you are not configuring the default profile, specify also the profile name:
-profileName *profileName*

When removing the configuration for a single server 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 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 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 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 are prompted for it.

3. **Optional:** Delete the database used by Business Process Choreographer.

For both the Business Process Choreographer database and the messaging database the following apply:

- The `bpeunconfig.jacl` script lists the databases that were used by the configuration that has been removed. You can then more easily identify the databases that are to be removed.
- When a Cloudscape database is used for the Business Process Choreographer database, the `bpeunconfig.jacl` script optionally removes the database, unless it is locked by a running application server. If the database is locked, stop the server, and use the `-conntype NONE` option.

4. **Optional:** Check the log file `install_root/profiles/profileName/logs/bpeunconfig.log`.

5. **Required:** Delete the database used by WebSphere default messaging. This database cannot be reused in a new configuration.

For both the Business Process Choreographer database and the messaging database the following apply:

- The `bpeunconfig.jacl` script displays a list of the databases that were used by the configuration that has been removed. The list of databases is also written to the `install_root/profiles/profileName/logs/bpeunconfig.log` log file. Use this information to identify the databases that are to be removed.

- When Cloudscape is the messaging database, the `bpeunconfig.jacl` script optionally removes the database, unless it is locked by a running application server. If the database is locked, stop the server, and use the `-conntype NONE` option.
6. **Optional:** For WebSphere MQ only, delete the queue manager used by Business Process Choreographer.
 7. **Optional:** Manually undo remaining settings that `bpeunconfig.jacl` does not undo. The following settings are not undone by the `bpeunconfig.jacl` script because it cannot determine whether the settings are still needed by other components:
 - enabling the `WorkAreaService`
 - enabling the `ApplicationProfileService`
 - enabling the `ObjectPoolService`
 - enabling the `StartupBeansService`
 - enabling the `CompensationService`
 - enabling the `WorkareaPartitionService`
 - enabling the WebSphere Security and Java 2 security
 - setting WebSphere variables
 - installing or adding target mappings for the `SchedulerCalendars` application
 8. If you configured Business Process Choreographer Observer, remove it and the event collector by running both setup scripts, but selecting the remove option, as described in the following:
 - a. “Configuring the Business Process Choreographer event collector” on page 86
 - b. “Configuring Business Process Choreographer Observer” on page 89

The Business Process Choreographer applications and associated resources (such as scheduler, data sources, listener ports, connection factories, queue destinations, activation specs, work area partition, mail session, and authentication aliases) have been removed.

Using the administrative console to remove the Business Process Choreographer configuration

Use this task to remove part or all of the business process container, task container, and Business Process Choreographer Explorer configuration, and the associated resources.

Before you can remove the Business Process Choreographer configuration, you must stop all process and task templates, delete all tasks and process instances, then stop and uninstall all enterprise applications that contain business processes or human tasks.

1. Uninstall the Business Process Choreographer enterprise applications.
 - a. Display the enterprise applications.
In the administrative console, select **Applications** → **Enterprise Applications**.
 - b. Identify the scope of the Business Process Choreographer installation.
Look for applications with the following names:
 - `BPEContainer_scope` is the business process container application.
 - `TaskContainer_scope` is the human task container application.

- BPCExplorer_scope is the Business Process Choreographer Explorer application.
- BPCObserver_scope is the Business Process Choreographer Observer application.
- BPCECollector_scope is the event collector application that is required by the Business Process Choreographer Observer.

Where the value of *scope* depends on your configuration:

- If Business Process Choreographer was configured on an application server, *scope* has the value *nodeName_serverName*.
 - If Business Process Choreographer was configured on a cluster, *scope* has the value *clusterName*.
- Optional:** If you installed the business process container, uninstall it. Select BPEContainer_scope, then click **Uninstall** → **OK** → **Save** → **Save**.
 - Optional:** If you installed the human task container, uninstall it.
 - 1) Select TaskContainer_scope, then click **Stop**.
 - 2) Select the application again, then click **Uninstall** → **OK** → **Save** → **Save**.
 - Optional:** If you installed Business Process Choreographer Explorer, uninstall it.
 - 1) Select BPCExplorer_scope, then click **Stop**.
 - 2) Select the application again, then click **Uninstall** → **OK** → **Save** → **Save**.
 - Optional:** If you installed Business Process Choreographer Observer, uninstall it and the Business Process Choreographer Common Event Infrastructure consumer.
 - 1) Select bpcobserver_scope and bpcobservereventconsumer_scope, then click **Stop**.
 - 2) Select the applications again, then click **Uninstall** → **OK** → **Save** → **Save**.
- Remove all or any of the following resources that you do not want to reuse:
 - Optional:** Find the Business Process Choreographer data source (the default name is BPEDataSourceDbType) and note its associated authentication data alias (if any) and Java Naming and Directory Interface (JNDI) name before removing it (for a single server, the default name is jdbc/BPEDB).
To find the data sources:
 - 1) Click **Resources** → **JDBC Providers**.
 - 2) If Business Process Choreographer was installed on an application server, select **Server**.
 - 3) If Business Process Choreographer was installed on a cluster, select the cluster.
 - 4) Click **Apply**.
 - 5) Select the appropriate JDBC provider, then click **Data sources**.
 - 6) If you are using an Oracle database management system, remove also a second data source: BPEDataSourceOracleNonXA.
 - Optional:** For a database other than a Cloudscape database, remove the JDBC provider of the data source identified in step 2, unless it contains further data sources that you still need.
 - Optional:** Remove the appropriate connection factories and queues.
 - For default messaging, before you remove the connection factories, note their associated authentication data aliases. Then remove the JMS connection factories and JMS queues.
 - 1) Click **Resources** → **JMS Providers** → **Default messaging**.

- 2) On the Default messaging provider pane perform one of the following:
 - If you configured Business Process Choreographer on a cluster, select **Cluster** then click **Apply**.
 - If you configured Business Process Choreographer on a server, select **Server** then click **Apply**.
- For WebSphere MQ, remove the JMS queue connection factories and JMS queue destinations.
 - 1) Click **Resources** → **JMS Providers** → **WebSphere MQ**.
 - 2) On the WebSphere MQ messaging provider pane, select **Server**. Then click **Apply**.

If you configured Business Process Choreographer on a cluster, you must repeat this for each server that is a member of the cluster.

For the business process container the JNDI names are normally as follows:

```
jms/BPECF
jms/BPECFC
jms/BPEIntQueue
jms/BPERetQueue
jms/BPEHldQueue
```

For the human task container the JNDI names are normally as follows:

```
jms/HTMCF
jms/HTMIntQueue
jms/HTMHldQueue
```

- d. **Optional:** If you are using WebSphere default messaging as the JMS provider, remove the activation specifications.
 - 1) Click **Resources** → **JMS Providers** → **Default messaging** → **JMS activation specification**.
 - 2) Remove the following activation specifications:


```
BPEInternalActivationSpec
HTMInternalActivationSpec
```
- e. **Optional:** If you are using WebSphere MQ as the JMS provider, remove the listener ports.
 - 1) Click **Servers** → **Application servers** → *serverName*.
 - 2) Under Communications, click **Messaging** → **Message Listener Service** → **Listener Ports**.
 - 3) On the Application servers pane, remove the following listener ports:


```
BPEInternalListenerPort
BPEApiListenerPort
BPEHoldListenerPort
HTMInternalListenerPort
```
- f. **Optional:** Delete the authentication data aliases.
 - If the data source identified in step 2 on page 100 had an authentication data alias, remove that alias.

Usually, the alias for the database is named *cellName/BPEAuthDataAliasdbType_Scope*, where:

```
cellName
    The name of the cell
dbType
    The database type
Scope
    One of the values given in step 1b on page 99
```


- If any of the connection factories identified in step 2c on page 100 have an authentication data alias, remove the alias.
Usually, the alias for the database is named *cellName/BPEAuthDataAliasJMS_Scope*, where:
cellName
The name of the cell
Scope
One of the values given in step 1b on page 99
The authentication data alias is in **Security** → **Global security** → **JAAS Configuration** → **J2C Authentication data**.
- g. **Optional:** Remove the scheduler configuration for the data source JNDI name.
 - 1) Click **Resources** → **Schedulers**.
 - 2) Select the scope of the Business Process Choreographer configuration; either **Server** or **cluster**. Then click **Apply**.
 - 3) On the Schedulers pane, note the work manager name, then select and delete the scheduler BPEScheduler.
- h. **Optional:** Remove the work manager.
 - 1) Click **Resources** → **Asynchronous beans** → **Work managers**.
 - 2) Select the scope of the Business Process Choreographer configuration; either **Server** or **cluster**. Then click **Apply**.
 - 3) On the Work managers pane, select and delete the work manager whose name you noted in step 2g.
- i. **Optional:** Remove the work area partition.
 - 1) Click **Servers** → **Application servers** → *serverName*.
 - 2) Under Business Process Services, click **Work area partition service**.
 - 3) On the Application servers pane, select and delete the work area partition BPECompensation.
- j. **Optional:** Remove the mail session.
 - 1) Click **Resources** → **Mail Providers**.
 - 2) On the Mail Providers pane, select **Cell**. Then click **Apply**.
 - 3) Click **Built-in Mail Provider**.
 - 4) Under Additional Properties, select **Mail sessions**.
 - 5) Select and delete *HTMailSession_Scope*, where *Scope* is the scope identified in step 1b on page 99
- k. In a cluster, repeat the removal of any server level resources for each cluster member.
- l. Save your configuration changes.
- m. Restart the application server.
- 3. **Optional:** If you use WebSphere default messaging for Business Process Choreographer, you can delete the bus member, bus, and data source:
 - a. Click **Service integration** → **Buses** → **BPC.cellName.Bus** → **Messaging engines**.
 - b. Select the messaging engine:
 - *nodeName.serverName-BPC.cellName.Bus* if you configured Business Process Choreographer on a server.
 - *clusterName-BPC.cellName.Bus* if you configured Business Process Choreographer in a cluster.

Note: If you configured Business Process Choreographer to use a remote messaging engine, *clusterName* might not match the name of the cluster where you configured Business Process Choreographer.

- c. In *Additional Properties*, select *Data store*, and note the JNDI name for the data source.
- d. Go to **Service integration** → **Buses** → **BPC.cellName.Bus** → **Bus members** and remove the bus member identified by one of the following:
 - *Node=nodeName*, *Server=serverName* if you configured Business Process Choreographer on a server.
 - *Cluster=clusterName* if you configured Business Process Choreographer on a cluster.
- e. **Optional:** If you removed the last member of the bus *BPC.cellName.Bus*, you can also remove the bus.
- f. **Optional:** Remove the data source. Click **Resources** → **JDBC Providers** → **Server** → **Apply** → **Cloudscape JDBC Provider** → **Data Sources**, then delete the data source that you noted in step 3c.
4. **Optional:** If you configured Business Process Choreographer on a cluster, delete the *BPC_REMOTE_DESTINATION_LOCATION* variable: Click **Environment** → **WebSphere Variables** → **Cluster** → **Apply**. Select the variable named *BPC_REMOTE_DESTINATION_LOCATION*, then click **Delete**.
5. Click **Save** to save all your deletions in the master configuration.
6. **Optional:** Delete the Business Process Choreographer database.
7. **Optional:** If you are using WebSphere MQ, delete the queue manager used by Business Process Choreographer.
8. If you use WebSphere default messaging for Business Process Choreographer, delete the data store for the message engine. If you use the default data store, you can delete the data store by deleting the following directory:
 - On Windows systems, delete

```
install_root\profiles\profileName\databases\com.ibm.ws.sib\  
nodeName.serverName-BPC.cellName.Bus
```
 - On UNIX and Linux systems, delete

```
install_root/profiles/profileName/databases/com.ibm.ws.sib/  
nodeName.serverName-BPC.cellName.Bus
```
9. **Optional:** If you configured the Business Process Choreographer Observer perform the following:
 - a. “Using the administrative console to remove the Business Process Choreographer event collector” for each event collector.
 - b. “Using the administrative console to remove Business Process Choreographer Observer” on page 105.

The Business Process Choreographer configuration has been removed.

Using the administrative console to remove the Business Process Choreographer event collector

Use this task to remove the Business Process Choreographer event collector configuration, and the associated resources that are required by the Business Process Choreographer Observer.

1. Display the enterprise applications.

In the administrative console, select **Applications** → **Enterprise Applications**.

2. Uninstall the Business Process Choreographer event collector application. Select the check box in front of `BPCECollector_node_name_server_name`, click **Uninstall** → **OK** → **Save**.
3. Delete the `BPCTransformerQueueDestination` destination:
 - a. Click **Service integration** → **Buses** → **CommonEventInfrastructure_Bus** .
 - b. Under **Destination Resources**, click **Destinations**.
 - c. Select the check box for `BPCTransformerQueueDestination_node_name_server_name`.
 - d. Click **Delete**.
4. Delete the `BPCCEIConsumerQueueConnectionFactory` JMS queue connection factory:
 - a. Click **Resources** → **JMS Providers** → **Default messaging (server scope)** → **JMS queue connection factory**.
 - b. Select the check box for `BPCCEIConsumerQueueConnectionFactory`.
 - c. Click **Delete**.
5. Delete the JMS queues:
 - a. Click **Resources** → **JMS Providers** → **Default messaging** → **JMS queue**.
 - b. Select the check boxes for `BPCCEIConsumerQueue_serverName` and `BPCTransformerQueue_nodeName_serverName`.
 - c. Click **Delete**.
6. Delete the JMS activation specifications:
 - a. Click **Resources** → **JMS Providers** → **Default messaging** → **JMS activation specification**.
 - b. Select the check boxes for `BPCCEIConsumerActivationSpec` and `BPCTransformerActivationSpec`.
 - c. Click **Delete**.
7. Delete the event profile group for server scope `BFMEEvents`:
 - a. Click **Resources** → **Common Event Infrastructure Provider** → **Event Group Profile List**.
 - b. Select the **Event groups list** link.
 - c. Select **Event Group Profiles**.
 - d. Select the check box for `BFMEEvents`.
 - e. Click **Delete**.
8. Delete the authentication data alias `BPCEventCollectorJMSAuthenticationAlias_nodeName_serverName`. Find your authentication data alias in **Security** → **Global security** → **JAAS Configuration** → **J2C Authentication data** and delete it.
9. Click **Save** to save your changes in the master configuration.
10. Drop the database, schema, and table space used by Business Process Choreographer Observer by running the following scripts that on Windows platforms are found in the directory `install_root\dbscripts\ProcessChoreographer\database_type`, and on Linux and UNIX platforms in the directory `install_root\dbscripts\ProcessChoreographer\database_type`:
 - `dropSchema_Observer.sql`
 - `dropTablespace_Observer.sql`
 - `dropDataBase_Observer.sql`

The Business Process Choreographer event collector configuration has been removed.

Using the administrative console to remove Business Process Choreographer Observer

Use this task to remove Business Process Choreographer Observer configuration, and the associated resources.

1. Display the enterprise applications.

In the administrative console, select **Applications** → **Enterprise Applications**.

2. Identify the scope of the Business Process Choreographer Observer installation.

Look for application named `BPCObserver_scope`.

- If Business Process Choreographer Observer was installed on an application server, *scope* has the value of *nodeName_serverName*.
- If Business Process Choreographer Observer was installed on a cluster, *scope* has the value of *clusterName*.

3. Uninstall the Business Process Choreographer Observer application.

a. Select `BPCObserver_scope`, then click **Stop**.

b. Select the application again, then click **Uninstall** → **OK** → **Save** → **Save**.

4. Click **Save** to save your changes in the master configuration.

The Business Process Choreographer Observer configuration has been removed.

Chapter 4. Administering

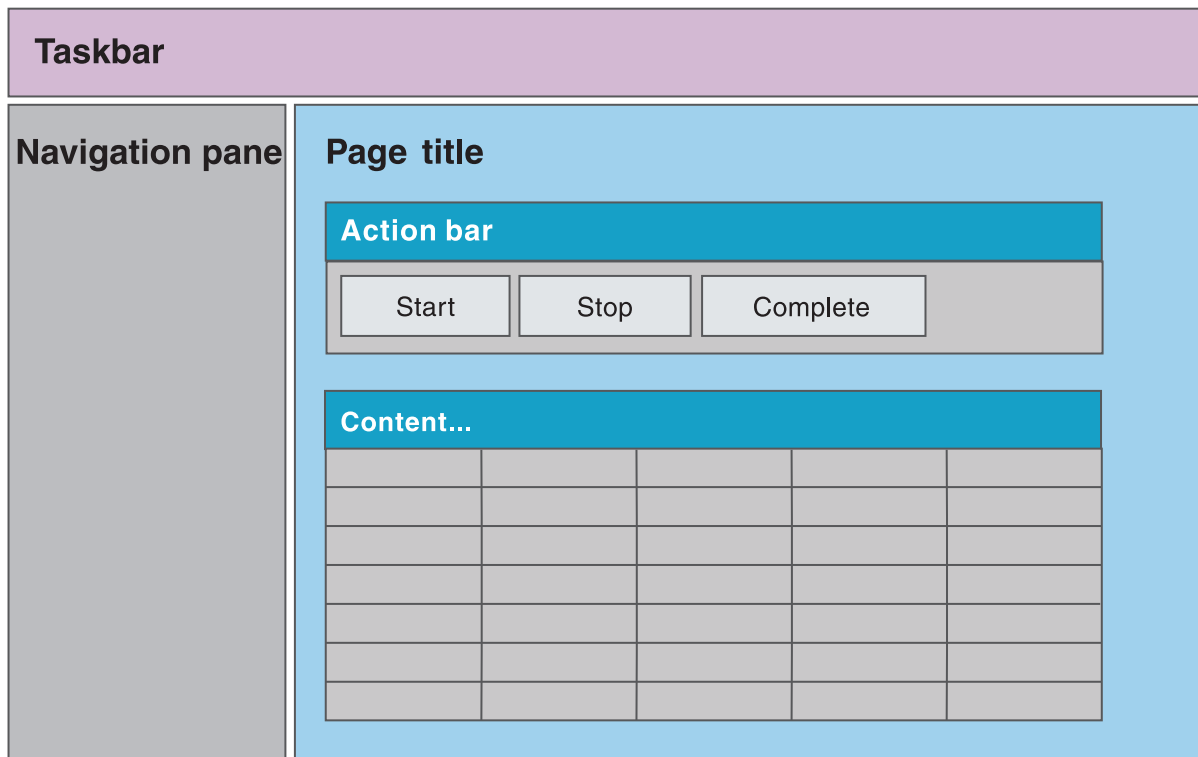
Using Business Process Choreographer Explorer

If you are a business process or a human task administrator, you can use Business Process Choreographer Explorer to administer business process and human task objects, such as process instances, failed activities, and work assignments. If you are a business user, you can use Business Process Choreographer Explorer to work with your assigned tasks.

Business Process Choreographer Explorer user interface

Business Process Choreographer Explorer is a stand-alone Web application that provides a set of administration functions for managing business processes and human tasks. The interface consists of a taskbar, a navigation pane, and the workspace.

The following figure shows the layout of the Business Process Choreographer Explorer user interface.



The user interface has the following main areas.

Taskbar

For all users, the taskbar offers options for logging out of Business Process Choreographer Explorer, and accessing online help. If you have system administrator rights, the taskbar also includes the following options:

Customize

Select this option to add views to and remove views from the navigation pane for this instance of Business Process Choreographer Explorer. You can also define the view that your users see when they log in.

Define views

Select this option to define customized views for your user group.

Navigation pane

The navigation pane on the left side of the user interface contains links to views that you use to administer objects, for example, process instances that you started, or human tasks that you are authorized to administer. The default user interface contains links to predefined views for business processes and tasks.

The system administrator can customize the content of the default navigation pane by adding and removing predefined views from the navigation pane and defining custom views to add to the navigation pane. All users can define personalized views from the navigation pane.

Workspace

The workspace on the right side of the user interface contains pages that you use to view and administer business-process and human-task related objects. You access these pages by clicking the links in the navigation pane, by clicking an action in the action bar, or by clicking links within the workspace pages.


Business Process Choreographer Explorer navigation pane

Use the navigation pane to access views that you use to administer business process and human task objects, such as process instances and work assignments. The default user interface contains links to predefined views for business processes and tasks. You can also define your own views and add these to the navigation pane.



Available actions




The following actions are available in the navigation pane:

- Navigate to a view.
Click the view name to navigate to that view.
- Collapse and expand a group.
Click the plus sign (+) beside an item in the navigation pane to expand it, or click the minus sign (-) to collapse the item. You can also click the item itself to toggle between its expanded and collapsed state.
- Search.

Click the **Search** icon (), to search for objects, or to define a personalized view.

Additional actions are available from the pop-up menu depending on the view type. An icon indicates whether a pop-up menu is available.

- To delete the view, click the **Delete** icon ().
- To modify the view, click the **Edit** icon ().

- To create a copy of the view and modify the copy, click the **Copy** icon ().
- To move the view up or down in the list, click the **Up** icon () or the **Down** icon ().

Predefined views in the default navigation pane

The default navigation pane contains the following groups of views. The views that are shown in the navigation pane in your Business Process Choreographer Explorer might differ depending on whether your system administrator has added views to, or removed views from the navigation pane.

Process templates

The process templates group contains the following view:

My Process Templates

This view shows a list of process templates. From this view you can display information about the process template and its structure, display a list of process instances that are associated with a template, and start process instances.

Process instances

The process instances group contains the following views:

Started By Me

This view shows the process instances that you started. From this view, you can monitor the progress of the process instance, and list the activities, processes, or tasks that are related to it.

Administered By Me

This view shows the process instances that you are authorized to administer. From this view, you can act on the process instance, for example, to suspend and resume a process, or monitor the progress of the activities in a process instance.

Critical Processes

This view shows process instances in the running state that contain activities in the stopped state. From this view, you can act on the process instances, or list the activities and then act on them.

Terminated Processes

This view shows process instances that are in the terminated state. From this view, you can act on these process instances.

Failed Compensations

This view shows the compensation actions that have failed for microflows.

Activity instances

The activity instances group does not contain any views in the default navigation pane.

Task templates

The task templates group contains the following view:

My Task Templates

This view shows a list of task templates. From this view you can create and start a task instance, and display a list of task instances that are associated with a template.

Task instances

The task instances group contains the following views:

My Tasks

This view shows a list of the task instances that you are authorized to work with. From this view, you can work on a task instance, release a task instance that you claimed, or transfer a task instance to another user.

All Tasks

This view shows all of the tasks for which you are the owner, potential owner, or editor. From this view, you can work on a task instance, release a task instance that you claimed, or transfer a task instance to another user.

Initiated By Me

This view shows the task instances that you initiated. From this view, you can work on a task instance, release a task instance that you claimed, or transfer a task instance to another user.

Administered By Me

This view shows the task instances that you are authorized to administer. From this view, you can act on the task instance, for example, to suspend and resume a process, to create work items for the task instance, or to display a list of the current work items for the task instance.



My Escalations


This view shows all of the escalations for the logged on user.

View types

The navigation pane can contain the following types of views. Depending on the view, additional actions are available from the pop-up menu.

- Predefined views in the default navigation pane.

These groups of views are available only if the navigation pane has not been changed by the system administrator in the Customize Navigation Tree and Login View page. A pop-up menu is not available for these views.
- Customized views and predefined views that were added to the navigation pane by the system administrator. Pop-up menus are available for the system administrator but not for business users.
 - The predefined views are indicated by the **Predefined view** icon: . A system administrator can use the pop-up menu to change the position of these views in the navigation pane.
 - The customized views are indicated by the **Custom view** icon: . A system administrator can delete, edit, copy, and move these views.
- Personalized views.

These views are indicated by the **Custom view** icon: . These views can be deleted, edited, copied, and moved by the user that created the view.

Starting Business Process Choreographer Explorer

Business Process Choreographer Explorer is a Web application that is installed as part of the configuration of the business process container. Before you can start using Business Process Choreographer Explorer from a Web browser, you must

have installed the business process container, human task container, and the Business Process Choreographer Explorer application, and the application must be running.

To start Business Process Choreographer Explorer, complete the following steps.

1. Direct your Web browser to the Business Process Choreographer Explorer URL. The URL takes the following form. The value of the URL depends on how the virtual host and context root were configured for your installation.

`http://app_server_host:port_no/context_root`

Where:

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 Business Process Choreographer Explorer. The port number depends on your system configuration. The default port number is 9080

context_root

The root directory for the Business Process Choreographer Explorer application on the application server. The default is bpc.

2. If security is enabled, you must enter a user ID and password, then click **Login**.

The initial page of the Business Process Choreographer Explorer is displayed. By default, this is the page that shows the My Tasks view.

Customizing Business Process Choreographer Explorer

Business Process Choreographer Explorer provides a user interface for administrators to manage business processes and human tasks, and for business users to work with their assigned tasks. Because this is a generic interface, you might want to customize this interface to address the business needs of your user groups.

You can customize the user interface in various ways.

Customizing the Business Process Explorer interface for different user groups

The navigation pane in the default Business Process Choreographer Explorer user interface contains a set of links to predefined views. The My Tasks view is the default view that is shown after you log in. If you have one of the Business Process Choreographer system administrator roles, you can customize the links that are shown in the navigation pane, the view that your users see after they log in, and the information that is shown in the views.

For example, the default user interface for Business Process Choreographer Explorer does not include views for working with business state machines. You can add predefined views to work with process templates and process instances for business state machines.

Or, you might want to offer users that deal with customer orders a different interface to the one that you offer the users dealing with customer service enquiries. You can customize an instance of Business Process Choreographer Explorer so that it meets the workflow patterns of a particular group of users.

In Business Process Choreographer Explorer, complete the following steps to customize the default user interface.

1. Customize the navigation pane and the default login view for this instance.
 - a. Click **Customize** in the taskbar.
 - b. In the Customize Navigation Tree and Login View page, select the views to include in and deselect the views to remove from the navigation pane.
 - c. Select the view that your users see when they log into Business Process Choreographer Explorer.

The list contains the views that you selected in the previous step and any customized views that you created from the Define Views page.
 - d. To save your changes, click **Save**.

To return the views for this instance to the default views, click **Restore defaults**. This action resets the navigation pane to the list of predefined views. Customized views in the navigation pane are not affected by this action.
2. **Optional:** Customize the views.

You can specify the information that is shown in the views for this Business Process Choreographer Explorer instance.

 - a. Click **Define Views** in the taskbar.
 - b. In the Search and Define Views page, select the type of view that you want to customize, for example, process templates.
 - c. In the Search and Define Views page for the view, select the properties to include in the view.

If this is a task instance view or a process instance view for administrative purposes, select **Administrative View** to show all of the items for which the logged-on user has a work item and to add administrative actions to the action bar in the view. If the logged-on user is a system administrator, all of the items that match the search criteria are shown regardless of whether the system administrator has work items for these items.

If you do not select **Administrative View**, the new view shows all of the items that the logged-on user is authorized to see.
 - d. Enter a display name for the view in the **List Name** field and click **Save**.

The new view appears in your navigation pane. Members of your user group see the new view when they next log into Business Process Choreographer Explorer.

Customizing views for process templates for business state machines:

Although a predefined view is provided for the process templates for business state machines, you might want to define your own views for this type of template.

To create customized views, you must have BPCSystemAdministrator authorization.

1. Click **Define Views** in the taskbar.
2. In the Search and Define Views page, select **Search and Define Views for Process Templates**.
3. Click the List Properties tab.
 - a. In the List Columns for Custom Properties, add a custom property with the following settings:
 - In the **Property Name** field, type generatedBy

- In the **Property Value** field, type `BusinessStateMachine`
 - In the **Display Name** field, type a display name for the column
- b. Add other custom properties, or add columns to or remove columns from the list of selected columns.
4. Type a display name for the query in the **List Name** field, and click **Save**.
 5. Add the view to the navigation pane.
 - a. Click **Customize** in the taskbar.
 - b. In the Customize Navigation Tree and Login View page, select the new business state machine view.
 - c. To save your changes, click **Save**.

A link to the business state machines view is added to the Process Templates group in the navigation pane. Your users see this view the next time they log into Business Process Choreographer Explorer.

Customizing views for process instances for business state machines:

Although a predefined view is provided for the process instances for business state machines, you might want to define your own views for this type of process instance.

To create customized views, you must have `BPCSSystemAdministrator` authorization.

1. Click **Define Views** in the taskbar.
2. In the Search and Define Views page, select **Search and Define Views for Process Instances**.
3. Click the Custom Properties tab.
 - a. Add a custom property with the following settings.
Type `generatedBy` in the **Property Name** field, and `BusinessStateMachine` in the **Property Value** field.
 - b. Add other custom properties as needed.
4. Click the List Properties tab.
 - a. In the List Columns for Query Properties, add the following query properties.
 - To add business state information to the view, type name in the **Property Name** field, `DisplayState` in the **Variable Name** field, and `tns` in the **Namespace** field, where `tns` is the target namespace of the business state machine suffixed by `-process`. Also specify a display name for the column in the **Display Name** field.
 - To add correlation information to the view, provide the appropriate information in the **Property Name** field, the **Variable Name** field, and the **Namespace** field. These values are derived from the definition of the business state machine. Also provide a display name for the column in the **Display Name** field.

Property Name

The name of the correlation property that you defined for the business state machine.

Variable Name

If the correlation set is initiated by incoming parameters, the variable name has the following format:

operation_name_Input_operation_parameter_name

where *operation_name* is the name of the operation for the transition out of the initial state.

If the correlation set is initiated by outgoing parameters, the variable name has the following format:

operation_name_Output_operation_parameter_name

Namespace

The namespace of the query property, where *tns* is the target namespace of the business state machine suffixed by *-process*.


- b. Add other custom properties or query properties, or add columns to or remove columns from the list of selected columns.
5. Type a name for the query in the **List Name** field, and click **Save**.
6. Add the view to the navigation pane.
 - a. Click **Customize** in the taskbar.
 - b. In the Customize Navigation Tree and Login View page, select the new business state machine view.
 - c. To save your changes, click **Save**.

A link to the business state machines view is added to the Process Instances group in the navigation pane. Your users see this view the next time they log into Business Process Choreographer Explorer.

Personalizing the Business Process Choreographer Explorer interface

The navigation pane in the default Business Process Choreographer Explorer user interface contains a set of links to predefined views and views that are defined by your system administrator. You can add your own views to your navigation pane, for example, to monitor the progress of a specific task or process.

In Business Process Choreographer Explorer, complete the following steps to personalize your user interface.

1. In the section of the navigation tree where you want to define the new view, click the **Search** icon ().
2. In the Search and Define Views for the new view, select the properties to include in the view.

If this is a task instance view or a process instance view for administrative purposes, select **Administrative View** to show all of the items for which you have administrative rights and to add administrative actions to the action bar in the view.

If you do not select **Administrative View**, the new view shows all of the items that you are authorized to see.
3. Enter a display name for the view in the **List Name** field and click **Save**.

The new view appears in your navigation pane.

Changing the appearance of the default Web application

Business Process Choreographer Explorer provides a ready-to-use Web user interface based on JavaServer Pages (JSP) files and JavaServer Faces (JSF) files. You can adapt the user interface to fit a certain look and feel without writing any new code.

A cascading style sheet (CSS) controls how the Web interface is rendered. You can change the CSS, for example, so that the default interface conforms to guidelines for corporate identity.

1. **Optional:** Modify the header. The `Menubar.jsp` file is always displayed in the user interface. The default `Menubar.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 the elements in the header, the navigation pane, and the content pane.

Styles used in the Business Process Choreographer Explorer interface:

The `style.css` file contains styles that you can change to adapt the look and feel of the default user interface.

The `style.css` file contains styles for the following elements of the default user interface:

- “Body of a page”
- “Login page”
- “Menu bar” on page 116
- “Navigator” on page 116
- “Content panels” on page 116
- “Command bar” on page 117
- “Lists” on page 117
- “Details panel” on page 117
- “Message data” on page 117
- “Tabbed panes” on page 118
- “Search pages” on page 118
- “Error details” on page 118
- “Sorting” on page 118

This file is in the following directory:

`<profile_directory>\installedApps\<node_name>\<explorer_instance>\bpexplorer.war\theme`

Body of a page

Style name	Description
<code>.pageBody</code>	The main content area in a table layout with two columns: navigator and content.
<code>.pageBody td</code>	The individual cell in the overall page body layout.
<code>.pageBodyNavigator</code>	The column that contains the navigator.
<code>.pageBodyContent</code>	The column that contains the content.

Login page

Style name	Description
<code>.loginPanel</code>	The panel containing the login form.
<code>.loginTitle</code>	The title on the form.
<code>.loginText</code>	The instructional text.

Style name	Description
.loginForm	The form that contains the input controls.
.loginValues	The table that determines the layout of the controls.
.loginField	The labels used for the logon fields, for example, Name or Password.
.loginValue	The text input field.

Menu bar

Style name	Description
.menubar	The JSF subview.
.menuContainer	The container panel including the menu items, for example, labels, and links.
.menuItem	An item on the menu bar.
.menuitem a	A menu item that is a link.
.menuitem a:visited	A menu item that is a link that the user has visited.
.menuitem a:hover	Hovering over a menu item that is a link.

Navigator

Style name	Description
.navigator	JSF subview for navigator which contains the links to the lists.
.navigatorTitle	The title for each navigator box.
.navigatorFrame	The division for each navigator box, for example, to draw a border.
.taskNavigatorTitle	A class of titles for navigation boxes. They are used to distinguish between links to lists of business process objects and human task objects.
.navigatorItem	An item in the navigator box.
.navigatorItemList	An item that represents a list.
.expanded / .expanded div / .expanded a .expanded a:visited	Used when the navigator boxes are expanded.
.collapsed	Used when the navigator boxes are collapsed.

Content panels

Style name	Description
.panelContainer	The division panel that contains the list, details or messages. This element is embedded in the pageBodyContent column.
.panelTitle	The title for the displayed content, for example, My Tasks.
.panelHelp	The division container that contains the help text and the icon.
.panelGroup	The division container that contains the command bar and list, details or message.

Command bar

Style name	Description
.commandbarHeader	The title above the command bar.
.commandbar	The division container around the command-bar area.

Lists

Style name	Description
.listHeader	The style used in the header row of the list.
.list	The table that contains the rows.
.list tbody td, .list th	Styles for the header row.
.list tthead input, .list tbody input	Check boxes for lists.
.list tfoot td	Last row entry is the footer information.
.list tfoot div	The division container around the footer elements.
.list tfoot input	The input controls in the footer.
.list a / .list a:visited	For links rendered in the list.

Details panel

Style name	Description
.details	The division container around a details panel.
div.details	Details styles that are embedded in the division container.
table.details	Details styles that are embedded in the table container.
td.detailsProperty	The label for a property name.
td.detailsValue	The text for a property value.

Message data

Style name	Description
.messageData	The division container around a message.
.messageData table	The table container in which the message is placed.
.messageDataButton	Button style for Add and Remove buttons in the message form.
.messageData td / .messageData th	Body and header cells.
.messageDataOutput	For rendering read-only text.
.messageDataValidInput	For message values that are valid.
.messageDataInvalidInput	For message values that are not valid.

Tabbed panes

Style name	Description
.tabbedPane	The division container around all of the tabbed panes.
.tabHeader	The tab header of a tabbed pane.
.tabHeader ul	Each header is organized in an unordered list.
.tabHeader li	Each header label is a list item.
.tabHeader a / .tabHeader a:hover / .tabHeader a.tab	The header label as a link.
.tabHeader a.selectedTab	The selected tab header.
.tabPane	The division container that encloses a tabbed pane.
.tabPane table	A pane is always embedded in a panel grid. This action results in a table container around the pane.
.tabPane .list th, .tabPane .list tfoot div	Settings for lists on a tabbed pane.

Search pages

Style name	Description
.searchPanel	The tabbed pane for a search panel. See also tabbed panes.
.searchPanelFilter	The table container for a search form.
.searchLabel	The label for a search form control.
.searchListBox	The list box control for select options.

Error details

Style name	Description
.errorPage	The tabbed pane for an error page.
.errorLink / .errorLink a / .errorLink a:visited	Styles uses to render the button links on a page.
.errorDetails	Tabbed pane with error details.
.errorDetailsStack	Tabbed pane with an exception stack.
.errorDetailsStack table / .errorDetailsStack td	The exception stack that is shown as rows in a table.
.errorDetailsMessage	Text style for error message.

Sorting

Style name	Description
.ascending	Style for the list header class when the list is sorted by this column in ascending order.
.descending	Style for the list header class when the list is sorted by this column in descending order.
.unsorted	Style for the list header class when the list is not sorted by this column.

Customizing input and output forms

The Business Process Choreographer Explorer interface provides default input and output forms for displaying and entering business data. You can use JSP documents to customize these default input and output forms.

To include user-defined JavaServer Pages (JSP) documents in the Web client, you must specify them when you model a human task in WebSphere Integration Developer. For example, you can provide JSP documents for a specific task and its input and output messages, and for a specific user role or all user roles. At runtime, the user-defined JSP documents are included in the user interface to display output data and collect input data.

The customized forms are not self-contained Web pages; they are HTML fragments that Business Process Choreographer Explorer imbeds in an HTML form, for example, fragments for labels and input fields.

When a button is clicked on the page that contains the customized forms, the input is submitted and validated in Business Process Choreographer Explorer. The validation is based on the type of the properties provided and the locale used in the browser. If the input cannot be validated, the same page is shown again and information about the validation errors is provided in the `messageValidationErrors` request attribute.

To add customized forms to Business Process Choreographer Explorer, complete the following steps using WebSphere Integration Developer.

1. Create the customized forms.

The user-defined JSP documents for the input and output forms used in the Web interface access message data. Use Java snippets or the JSP execution language to access the business data from the request context.

2. Assign the JSP documents to a task.

Open the human task in the human task editor. In the client settings, specify the location of the user-defined JSP documents and the role to which the customized form applies, for example, administrator. The client settings for Business Process Choreographer Explorer are stored in the task template. At runtime these settings are retrieved with the task template.

3. Package the user-defined JSP documents in a Web archive (WAR file).

You can either include the WAR file in the enterprise archive with the module that contains the tasks or deploy the WAR file separately.

The customized forms are rendered in Business Process Choreographer Explorer at runtime.

User-defined JSP fragments:

The user-defined JSP fragments are imbedded in an HTML form tag. At runtime, Business Process Choreographer Explorer includes these fragments in the rendered page.

The user-defined JSP fragment for the input message is imbedded before the JSP fragment for the output message.

```
<html....>
  ...
  <form...>
```

```

        Input JSP (display task input message)

        Output JSP (display task output message)

    </form>
    ...
</html>

```

Because the user-defined JSP fragments are embedded in an HTML form tag, you can add input elements. The name of the input element must match the XML Path Language (XPath) expression of the data element. It is important to prefix the name of the input element with the provided prefix value:

```

<input id="address"
      type="text"
      name="{prefix}/selectPromotionalGiftResponse/address"
      value="{messageMap['/selectPromotionalGiftResponse/address']}"
      size="60"
      align="left" />

```

The prefix value is provided as a request attribute. The attribute ensures that the input name is unique in the enclosing form. The prefix is generated by Business Process Choreographer Explorer and it should not be changed:

```
String prefix = (String)request.getAttribute("prefix");
```

The prefix element is set only if the message can be edited in the given context. Output data can be displayed in different ways depending on the state of the human task. For example, if the task is in the claimed state, the output data can be modified. However, if the task is in the finished state, the data can be displayed only. In your JSP fragment, you can test whether the prefix element exists and render the message accordingly. The following JSTL statement shows how you might test whether the prefix element is set.

```

...
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%
...
<c:choose>
  <c:when test="{not empty prefix}">
    <!--Read/write mode-->
  </c:when>
  <c:otherwise>
    <!--Read-only mode-->
  </c:otherwise>
</c:choose>

```

Administering Business Process Choreographer

You can administer Business Process Choreographer using the administrative console or using scripts.

Using the administrative console to administer Business Process Choreographer

Describes the administrative actions that can be performed using the administrative console.

Administering the compensation service for a server

Use the administrative console to start the compensation service automatically, when the application starts, and to specify the location and maximum size of the recovery log.

The compensation service must be started on an application server, when business processes are run on that server. The compensation service is used to manage updates that might be made in a number of 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. On the Configuration tab, under Container Settings, click **Container services** → **Compensation service**. This action displays a panel with the compensation service properties.

Enable service at server startup

Specifies that, whenever the application server starts, it automatically tries to start the compensation service.

Make sure that this check box is selected. The compensation service must be enabled when you use business processes. If you run your business processes on a cluster, you must enable the compensation service for each server in the cluster.

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.

4. **Optional:** If necessary, change the compensation service properties.
5. Click **OK**.
6. To save your configuration, click **Save** in the Messages box of the administrative console window. Then click **Save** on the Application Servers Save pane.

Querying and replaying failed messages, using the administrative console

This describes how to check for and replay any messages for business processes or human tasks that could not be processed.

When a problem occurs while processing a message, it is moved to the retention queue or hold queue. This task describes how to determine whether any failed messages exist, and to send those messages 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*.
 - b. On the **Configuration** tab, in the Container Settings section, click one of the following sequences:
 - For business processes: **Business process container settings** → **Runtime Configuration**
 - For human tasks: **Human task container settings** → **Runtime Configuration**

The number of messages on the hold queue and retention queue are displayed under **General Properties**.

2. If either the hold queue or the retention queue contains messages, you can move the messages to the internal work queue.

Click one of the following options:

- For business processes: **Replay Hold Queue** or **Replay Retention Queue**
- For human tasks: **Replay Hold Queue**

Note: When security is enabled, the replay buttons are only visible to users who have operator authority.

Business Process Choreographer tries to service all replayed messages again.

Refreshing the failed message counts:

Use the administrative console to refresh the count of failed messages for business processes or human tasks.

The displayed number of messages on the hold queue and on the retention queue, and the number of message exceptions, remain static until refreshed. This task describes how to update and display the number of messages on those queues and the number of message exceptions.

1. Select the appropriate application server.

Click **Servers** → **Application servers** → *server_name*.

2. Refresh the message counts.

On the **Configuration** tab, in the Container Settings section, click one of the following sequences:

- For business processes: **Business process container settings** → **Runtime Configuration** → **Refresh Message Count**
- For human tasks: **Human task container settings** → **Runtime Configuration** → **Refresh Message Count**

The following updated values are displayed under **General Properties**:

- For business processes: The number of messages on the hold queue and on the retention queue
- For human tasks: The number of messages on the hold queue
- If any exceptions occurred while accessing the queues, the message text is displayed in the Message Exceptions field.

On this page, you can also replay the messages on these queues.

Failed message handling and quiesce mode:

Business Process Choreographer provides a facility for handling temporary infrastructure failures.

This section describes how the business process container handles failed messages. This contrasts with the simpler mechanism used by the human task container, described in “Failed message handling for human tasks” on page 125.

Long-running processes consist of a sequence of transactions. The transactions are separated by Java Message Service (JMS) messages, which the server sends to a message-driven bean. This bean passes the incoming messages to the process server, for processing. Each transaction consists of the following actions:

- Receive a message.

- Navigate, on behalf of the message.
- Send messages that trigger follow-on transactions.

The server might fail to process a message received by the message-driven bean for either of the following reasons:

- A specified number of consecutive messages cannot be processed. The infrastructure is therefore assumed to be unavailable.
- Only some messages can be processed. Any single message that cannot be processed is assumed to be damaged.

The responses to these causes are as follows:

Cause	Response
Unavailable infrastructure	The message-driven bean tries, for a specified time, to recover from that situation. It tries to keep all messages available until the server is again operational. This problem might be caused by a database failure, for example.
Damaged message	After a specified number of retries, the message is put into the hold queue, where it can be manipulated or reviewed. From the hold queue, it can also be moved back to the input queue, to retry the transaction.

The implementation for messages for business processes is as follows:

- If a message fails to be processed, the server puts it into a retention queue, where it is kept available, in case this is an infrastructure problem that is fixed within a specified time.
- When a message is in the retention queue, the options are as follows:
 - When a subsequent message can be processed successfully, all messages from the retention queue are moved back to the input queue of the message-driven bean. For each message, a count is maintained of how often the message has been sent to the retention queue. If this count exceeds the retry limit for a given message, the message is put in the hold queue.
 - If the next message fails to be processed, it is also put in the retention queue. This process continues until the threshold of maximum messages in the retention queue is reached. When this threshold is reached, the message-driven bean moves all messages from the retention queue to the input queue and switches into quiesce mode.

When the message-driven bean operates in quiesce mode, it periodically tries to process a message. Messages that fail to be processed are put back in the hold queue, without incrementing either the delivery count or the retention queue traversal count. As soon as a message can be processed successfully, the message-driven bean switches back into normal processing mode.

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 be transferred through the retention queue before being put on the hold queue.

To be put on the retention queue, the processing of a message must fail three times.

For example, if the retry limit is 5, a message must go through the retention queue five times (it must fail for $3 * 5 = 15$ times), before the last retry loop is started. If the last retry loop fails two more times, the message is put on the hold queue. This means that a message must fail $(3 * RetryLimit) + 2$ times before it is put on the hold queue.

In a performance-critical application running in a reliable infrastructure, the retry limit should be small: one or two, for example.

To locate this parameter in the administrative console, click **Servers** → **Application Servers** → *server_name*. Then, under the heading Business Process Container Settings, click **Business Process Container**.

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 enter quiesce mode as soon as one message fails, set the value to zero. To make the business process container more tolerant of infrastructure failures, increase the value.

To locate this parameter in the administrative console, click **Servers** → **Application Servers** → *server_name*. Then, under the heading Business Process Container Settings, click **Business Process Container**.

Retention queue

The retention queue holds failed messages that are replayed by moving them back to the business process container's internal work queue. A message is put on the retention queue if it fails three times. If the message fails $(3 * RetryLimit) + 2$ times, it is put on the hold queue. (For details of the retry limit, see "Retry limit" on page 123.) If the retention queue is full to the limit defined by the retention queue message limit and another message fails, the queue overflows, 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 contains messages that have failed $(3 * RetryLimit) + 2$ times. (For details of the retry limit, see "Retry limit" on page 123.) 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 can be done using the administrative console or using administrative commands.

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.

Failed message handling for human tasks

The human task container does not have a retention queue, nor retry limits. It only has a hold queue, on which failed messages are placed, and from which, they can be replayed.

Refreshing staff query results, using the administrative console

The results of a staff query are static. Use the administrative console to refresh staff queries.

Business Process Choreographer caches the results of staff assignments evaluated against a staff directory, such as an Lightweight Directory Access Protocol (LDAP) server, in the runtime database. If the staff directory changes, you can force the staff assignments to be evaluated again.

To refresh the staff queries:

1. Click **Servers** → **Application servers** → *server_name*.
2. On the **Configuration** tab, in the Container Settings section, click **Human task container settings** → **Runtime Configuration** → **Refresh Staff Queries**. All staff queries are refreshed.

Note: When security is enabled, the refresh button is only visible to users who have operator authority.

Refreshing the staff query results in this way can cause a high load on the application and database. Consider using the alternative methods listed below.

Related tasks

“Refreshing staff query results, using administrative commands” on page 135
The results of a staff query are static. Use the administrative commands to refresh staff queries.

“Refreshing staff query results, using the refresh daemon” on page 137
Use this method if you want to set up a regular and automatic refresh of all expired staff query results.

Enabling Common Base Events and the audit trail

Use this task to enable Business Process Choreographer events to be emitted to the Common Event Infrastructure as Common Base Events, or stored in the audit trail, or both.

You can change the state observers settings for the business process container or the human task container, permanently on the Configuration tab, or temporarily on the Runtime tab. Any choices you make on these Configuration or Runtime tabs affect all applications executing in the appropriate container. For changes to affect both the business process container and the human task container, you must change the settings separately for them both.

Changing the configured logging infrastructure:

Use this task to change the state observer logging for the audit log or common event infrastructure logging for the configuration.

Choices made on the Configuration tab are activated the next time the server is started. The chosen settings remain in effect whenever the server is started.

Make changes to the configuration, as follows:

1. Display the Business process container or Human task container pane.
Click **Servers** → **Application servers** → *server_name*. Then, under **Container Settings**, click one of the following sequences:
 - For business processes: **Business process container settings** → **Business process container**
 - For human tasks: **Human task container settings** → **Human task container**
2. In the General Properties section, select the logging to be implemented. The state observers are independent of each other: you can enable or disable either or both of them.

Enable Common Event Infrastructure logging

Select this check box to enable event emission that is based on the Common Event Infrastructure.

Enable audit logging

Select this check box to store the audit log events in the audit trail tables of the relational database.

3. Accept the change.
 - a. Click **Apply**.
 - b. In the Messages box, click **Save**.
 - c. On the Application servers pane, click **Save**.

The state observers are set, as you required. The changes take place after server restart.

Restart the container, to effect the changes.

Configuring the logging infrastructure for the session:

Use this task to change the state observer logging for the audit log or common event infrastructure logging for the session.

Choices made on the Runtime tab are effective immediately. The chosen settings remain in effect until the next time the server is started.

Make changes to the session infrastructure, as follows:

1. Display the Replay messages pane.
Click **Servers** → **Application servers** → *server_name*. Then, under Container Settings, click one of the following sequences:
 - For business processes: **Business process container settings** → **Runtime Configuration**
 - For human tasks: **Human task container settings** → **Runtime Configuration**
2. In the **State observer logging** section, select the logging to be implemented. The state observers are independent of each other: you can enable or disable either or both of them.

Enable Common Event Infrastructure logging

Select this check box to enable event emission that is based on the Common Event Infrastructure.

Enable audit logging

Select this check box to store the audit log events in the audit trail tables of the relational database.

Save runtime changes to configuration as well

Select this check box to update the configuration with the changes you made for this session.

3. Accept the change.

Click **OK**.

The state observers are set, as you required.

Event emission and storage:

Events for state changes can be generated for executing business processes, human tasks, or both.

Two infrastructures emit or store the events, such that those events can be retrieved by applications. Applications might use events to monitor business processes and to analyze the history of business processes, or human tasks, or both.

Task events, for example, can be emitted without having a business process involved. These events can be consumed by the audit trail and the Common Event Infrastructure (CEI). This applies to standalone tasks, purely human tasks, and tasks invoked by application components other than business processes.

Because the generation of events impacts system performance, you can select the infrastructure to be used to store and emit events:

Common Event Infrastructure

Events can be both stored and published to subscribing applications. To use this event infrastructure, make sure that the Common Event Infrastructure is installed and configured.

Use event emission that is based on the Common Event Infrastructure to retrieve events in the format of Common Base Events, through the application programming interface (API) of the Common Event Infrastructure. You can connect consuming applications either by subscription or by using the query-oriented interface of the Common Event Infrastructure.

Event emission that is based on the Common Event Infrastructure has considerably greater impact on system performance than have audit log events. However, it provides greater flexibility for consuming applications.

Audit trail

Events are stored as records of a table in a relational database.

This is a fast event-storing infrastructure that has little impact on performance. Consuming applications need Structured Query Language (SQL) queries to retrieve the events from the database.

You can select either, both, or neither of the infrastructures. Selecting an infrastructure does not imply that events are necessarily stored or emitted. The selection enables the infrastructure, while you can control the actual generation of events later, by additional mechanisms. However, the enablement of an infrastructure results in a basic overhead that affects system performance.

Using scripts to administer Business Process Choreographer

Describes the administrative actions that can be performed using scripts.

Deleting audit log entries, using administrative commands

Use the administrative commands to delete some or all audit log entries.

Before you begin this procedure, the following conditions must be met:

- The application server through which audit log entries are to be deleted must be running. That is, the `-conntype none` option of `wsadmin` cannot be used, because a server connection is required.
- When security is enabled, the user ID that you use must have operator authority.

You can use the `deleteAuditLog.py` script to delete audit log entries from the database.

1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

Enter the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Delete the entries in the audit log table.

Enter one or more of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin -lang jython -f deleteAuditLog.py  
                        -server serverName  
                        [-profile profileName]  
                        [options]
```

```
install_root/bin/wsadmin -lang jython -f deleteAuditLog.py  
                        -node nodeName  
                        -server serverName  
                        [-profile profileName]  
                        [options]
```

```
install_root/bin/wsadmin -lang jython -f deleteAuditLog.py  
                        -cluster clusterName  
                        [-profile profileName]  
                        [options]
```

Where:

-cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

-node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node.

-server *serverName*

The name of the server. Required if the cluster name is not specified.

-profileName *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

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, or the default number.

-time *timestamp*

Deletes all the audit log entries that are older than the time you specify for *timestamp*. The time used is coordinated universal time (UTC). Its 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.

The *-time* and *-processtime* options are mutually exclusive.

-processtime *timestamp*

Deletes all the audit log entries that belong to a process that finished before the time you specify for *timestamp*. Use the same time format as for the *-time* parameter.

The *-time* and *-processtime* options are mutually exclusive.

-slice *size*

Used with the *-all* parameter, *size* specifies the number of entries included in each transaction. The optimum 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.

Note: The jacl version of the cleanup unused staff query script, `deleteAuditLog.jacl`, is deprecated. It is available in the `util` subdirectory of the `ProcessChoreographer` directory and it takes the same parameters as described here, but the `-lang jython` option must be omitted.

Deleting process templates and task templates that are no longer valid

Use the administrative commands to delete, from the database, process templates and task templates that are no longer valid.

Before you begin this procedure, the application server on which templates are to be deleted must be running. That is, the `-conntype none` option of `wsadmin` cannot be used, because a server connection is required. No special authority is required to run this command, even if security is enabled.

Use the methods described here to remove, from the database, templates and all objects that belong to them if no corresponding valid application in the WebSphere configuration repository contains them. This situation can occur if an application installation was canceled or not stored to the Configuration Repository by the user. These templates usually have no impact. They are not shown in Business Process Choreographer Explorer.

There are rare situations in which these templates cannot be filtered. They must then be removed from the database with the following scripts.

You cannot use the scripts to remove templates of valid applications from the database. This condition is checked and a `ConfigurationError` exception is thrown if the corresponding application is valid.

1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

Enter the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Delete, from the database, business process templates or human task templates that are no longer valid.

To delete, a business process template that is no longer valid, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin.sh -lang jython -f deleteInvalidProcessTemplate.py
    -server serverName
    -template templateName
    -validFrom validFromString
    [-profileName profileName]
```

```
install_root/bin/wsadmin.sh -lang jython -f deleteInvalidProcessTemplate.py
    -server serverName
    -node nodeName
    -template templateName
    -validFrom validFromString
    [-profileName profileName]
```

```
install_root/bin/wsadmin.sh -lang jython -f deleteInvalidProcessTemplate.py
    -cluster clusterName
    -template templateName
    -validFrom validFromString
    [-profileName profileName]
```

To delete, a human task template that is no longer valid, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin.sh -lang jython -f deleteInvalidTaskTemplate.py
    -server serverName
    -template templateName
    -validFrom validFromString
    -nameSpace nameSpace
    [-profileName profileName]
```

```
install_root/bin/wsadmin.sh -lang jython -f deleteInvalidTaskTemplate.py
    -server serverName
    -node nodeName
    -template templateName
    -validFrom validFromString
    -nameSpace nameSpace
    [-profileName profileName]
```

```
install_root/bin/wsadmin.sh -lang jython -f deleteInvalidTaskTemplate.py
    -cluster clusterName
    -template templateName
    -validFrom validFromString
    -nameSpace nameSpace
    [-profileName profileName]
```

Where:

cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster. You can specify the cluster name or the server name and node name.

node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node. You can specify the server name and node name or the cluster name.

server *serverName*

The name of the server. Required if the cluster name is not specified. You can specify the server name and node name or the cluster name.

template *templateName*

The name of the process template or task template to be deleted.

validFrom *validFromString*

The date from which the template is valid (in UTC) as displayed in the administrative console. The string should have the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, 2005-01-31T13:40:50

nameSpace *nameSpace*

The target namespace of the task template.

profileName *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

Note: The jacl version of the cleanup unused staff query script, `deleteInvalidTaskTemplate.jacl`, is deprecated. It is available in the `util` subdirectory of the ProcessChoreographer directory and it takes the same parameters as described here, but the `-lang jython` option must be omitted.

Deleting completed process instances

Use an administrative command to selectively delete from the Business Process Choreographer database any top-level process instances that have reached an end state of finished, terminated, or failed.

Before you begin this procedure, the application server on which process instances are to be deleted must be running. That is, the `-conntype none` option of `wsadmin` cannot be used, because a server connection is required. No special authority is required to run this command, even if security is enabled.

A top-level process instance is considered completed if it is in one of the following end states: finished, terminated or failed. You specify criteria to selectively delete top-level process instances and all their associated data (such as activity instances, child process instances, and inline task instances) from the database.

1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

Type the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Delete process instances from the database.

Enter the following command:

```
install_root/bin/wsadmin -lang jython -f deleteCompletedProcessInstances.py  
  [[(-node nodeName) -server serverName) | (-cluster clusterName)]  
  (-all | -finished | -terminated | -failed )  
  [-templateName templateName [-validFrom timestamp]]  
  [-startedBy userID ]  
  [-completedBefore timestamp]  
  [-profileName profileName]
```

Where:

-node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node. You can specify the server name and node name or the cluster name.

-server *serverName*

The name of the server. Required if the cluster name is not specified. You can specify the server name and node name or the cluster name.

-cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster. You can specify the cluster name or the server name and node name.

-all | -finished | -terminated | -failed

Specifies which process instances are to be deleted according to their state. You can specify a combination of finished, terminated, failed, or all.

-templateName *templateName*

Optionally, specifies the name of the process template or human task template to be deleted. If this option is specified, you can also use the `validFrom`

-validFrom *timestamp*

The date from which the template is valid (in UTC) as displayed in the administrative console. This option can only be used with the `templateName` option. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, 2006-11-20T12:00:00

startedBy *userID*

Optionally, only deletes completed process instances that were started by the given User ID.

-completedBefore *timestamp*

Optionally, deletes completed process instances that completed before the given time. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, 2006-07-20T12:00:00

profileName *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

For example, to delete all of the process instances running on node *myNode* in server *myServer* that are in the state finished, and were started by the user Antje, run the following command:

```
wsadmin -lang jython -f deleteCompletedProcessInstances.py
        -node myNode -server myServer
        -finished
        -startedBy Antje
```

The completed process instances have been deleted from the database.

Deleting data from the observer database

Use an administrative command to selectively delete from the Business Process Choreographer Observer database all of the data for process instances that match specified conditions.

You can delete the observer information for process instances in three ways:

- Delete observer data for process instances that reached the end state deleted before a specified time.
 - Delete observer data for process instances of a specific process template version.
 - Delete observer data for a process instance regardless of its state if the last event was received before a specified time.
1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

Type the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Delete observer data for process instances from the database.

Enter the following command:

```
install_root/bin/wsadmin -lang jython
-f observerDeleteCompletedProcessInstances.py
[[[-node nodeName] -server serverName) | (-cluster clusterName)]
-dataSource dataSourceJNDIName
( -templateName templateName -validFrom timestamp)
| -completedBefore timestamp
| -force -state state -reachedBefore timestamp
[-profileName profileName]
```

Where:

-node *nodeName*

This name identifies the node. The default is the local node. This parameter is optional.

-server *serverName*

The name of the server. This is parameter optional. The default is the default server.

-cluster *clusterName*

The name of the cluster. This parameter is optional.

-datasource *datasourceJNDIName*

Identifies the database that the command will act on. This parameter is required because a server or cluster can have multiple observer databases.

-templateName *templateName*-**validFrom** *timestamp*

Optionally, specifies the name of the process template for which observer data will be deleted. If this option is specified, you must also specify the `-validFrom` option.

The *timestamp* string specifies the date from which the template is valid (in UTC) as displayed in the administrative console. It has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, 2006-11-20T12:00:00

-completedBefore *timestamp*

Optionally, deletes observer data for process instances that completed before the given time. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, 2006-07-20T12:00:00

-force -state *state* **-reachedBefore** *timestamp*

Optionally, forces the deletion of observer data for process instances that reached the given state before the given time. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, 2006-07-20T12:00:00

profileName *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

For example, to delete all process instances running on node *myNode* in server *myServer* that reached the state finished before midday on May 16, 2008, run the following command:

```
wsadmin -lang jython -f observerDeleteCompletedProcessInstances.py
-node myNode -server myServer
-force -state finished -reachedBefore 2008-05-16T12:00:00
```


If successful, the tool reports the number of instances for which observer data was deleted and the number of table entries that were deleted from the database. Otherwise, error information is reported and no changes are made to the database.

The observer data for the specified process instances has been deleted from the observer database.

Querying and replaying failed messages, using administrative commands

Use the administrative commands to determine whether there are any failed messages for business processes or human tasks, and, if there are, to retry processing them.

Before you begin this procedure, the following conditions must be met:

- The application server on which the messages are to be queried or replayed must be running. That is, the `-conntype none` option of the `wsadmin` script cannot be used, because a server connection is required.
- When security is enabled, you must have operator authority.

When a problem occurs while processing an internal message, this message ends up on the retention queue or hold queue. To determine whether any failed messages exist, and to send those messages to the internal queue again:

1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

Enter the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Query the number of failed messages on both the retention and hold queues. Enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin -lang jython -f queryNumberOfFailedMessages.py  
-cluster clusterName  
[ -bfm | -htm ]  
[-profile profileName]
```

```
install_root/bin/wsadmin -lang jython -f queryNumberOfFailedMessages.py  
-node nodeName  
-server serverName  
[ -bfm | -htm ]  
[-profile profileName]
```

Where:

cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node.

server *serverName*

The name of the server. Required if the cluster name is not specified.

bfm | htm

These keywords are optional. The default, if neither option is specified is to display all failed messages for both business processes and human tasks. If you only want to display the number of messages in the business process

container hold and retention queues, specify the `-bfm` option. If you only want to display the number of messages in the human task container hold queue, specify the `-htm` option.

profile *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

3. Replay all failed messages on the hold queue, retention queue, or both queues. Enter one of the following commands:

```
install_root/bin/wsadmin -lang jython -f replayFailedMessages.py -cluster clusterName -queue r
install_root/bin/wsadmin -lang jython -f replayFailedMessages.py -node nodeName -server serverName
install_root/bin/wsadmin -lang jython -f replayFailedMessages.py -server serverName -queue rep
```

Where:

queue *replayQueue*

Must have one of the following values:

holdQueue
retentionQueue
both

cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node.

server *serverName*

The name of the server. Required if the cluster name is not specified.

bfm | htm

These keywords are optional and mutually exclusive. The default, if neither option is specified is to replay failed messages for both business processes and human tasks. If you only want to replay the messages for business processes, specify the `-bfm` option. If you only want to replay messages for human tasks, specify the `-htm`.

profile *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

Note: The `jacl` version of the cleanup unused staff query script, `replayFailedMessages.jacl`, is deprecated. It is available in the `util` subdirectory of the `ProcessChoreographer` directory and it takes the same parameters as described here, but the `-lang jython` option must be omitted.

Refreshing staff query results, using administrative commands

The results of a staff query are static. Use the administrative commands to refresh staff queries.

Before you begin this procedure, the following conditions must be met:

- The application server on which the messages are to be queried or replayed must be running. That is, the `-conntype none` option of `wsadmin` cannot be used, because a server connection is required.
- When security is enabled, you must have operator authority.

Business Process Choreographer caches the results of staff assignments evaluated against a staff directory, such as an Lightweight Directory Access Protocol (LDAP) server, in the runtime database. If the staff directory changes, you can force the staff assignments to be evaluated again.

1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

Enter the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Force the staff assignment to be evaluated again.

Enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin -lang jython -f refreshStaffQuery.py  
-server serverName  
[-processTemplate templateName |  
(-taskTemplate templateName [-nameSpace nameSpace]) |  
-userList username{,username}...]  
[-profile profileName]
```

```
install_root/bin/wsadmin -lang jython -f refreshStaffQuery.py  
-node nodeName  
-server serverName  
[-processTemplate templateName |  
(-taskTemplate templateName [-nameSpace nameSpace]) |  
-userList username{,username}...]  
[-profile profileName]
```

```
install_root/bin/wsadmin -lang jython -f refreshStaffQuery.py  
-cluster clusterName  
[-processTemplate templateName |  
(-taskTemplate templateName [-nameSpace nameSpace]) |  
-userList username{,username}...]  
[-profile profileName]
```

Where:

cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node.

server *serverName*

The name of the server. Required if the cluster name is not specified.

processTemplate *templateName*

The name of the process template. Staff assignments that belong to this process template are refreshed.

taskTemplate *templateName*

The name of the task template. Staff assignments that belong to this task template are refreshed.

nameSpace *nameSpace*

The namespace of the task template.

userList *userName*

A comma-separated list of user names. Staff assignments that contain the specified names are refreshed.

profileName *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

Note: If you do not specify any *templateName* nor *userList*, all staff queries that are stored in the database are refreshed. You might want to avoid this for performance reasons.

Note: The jacl version of the refresh staff query script, `refreshStaffQuery.jacl`, is deprecated. It is available in the `util` subdirectory of the `ProcessChoreographer` directory and it takes the same parameters as described here, but the `-lang jython` option must be omitted.

Refreshing staff query results, using the refresh daemon

Use this method if you want to set up a regular and automatic refresh of all expired staff query results.

Staff queries are resolved by the specified staff plug-in provider repository. The result is stored in the Business Process Choreographer database. To optimize the authorization performance, the retrieved query results are cached. The cache content is checked for currency when the staff query refresh daemon is invoked.

In order to keep staff query results up to date, a daemon is provided that refreshes expired staff query results on a regular schedule. The daemon refreshes all cached staff query results that have expired.

1. To go to the custom properties page for the human task container:
Click **Servers** → **Application servers** → *Server_Name* then on the **Configuration** tab in the **Container Settings** section, click **Human task container settings** → **Runtime Configuration**.
2. In the field **Staff query refresh schedule** enter the schedule using the syntax as supported by the WebSphere CRON calendar. This value determines when the daemon will refresh any expired staff query results. The default value is `"0 0 1 * * ?"`, which causes a refresh every day at 1 am.
3. In the field **Timeout for staff query result** enter a new value in seconds. This value determines how long a staff query result is considered to be valid. After this time period, the staff query result is considered to be no longer valid, and the staff query will be refreshed the next time that the daemon runs. The default is one hour.
4. Click **OK**.
5. Save the changes and restart the human task container application to make the changes effective.
The new expiration time value applies only to new staff queries, it does not apply to existing staff queries.

Removing unused staff query results, using administrative commands

Use the administrative commands to remove unused staff query results from the database.

Before you begin this procedure, the following conditions must be met:

- The application server, through which unused staff queries are to be deleted, must be running. That is, the `-conntype none` option of `wsadmin` cannot be used, because a server connection is required.
- When security is enabled, you must have operator authority.

Business Process Choreographer maintains lists of user names in the runtime database for staff expressions that have been evaluated. Although the process instances and human tasks that used the staff expressions have 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 Business Process Choreographer subdirectory where the administration scripts are located.

Enter the following command:

```
cd install_root/ProcessChoreographer/admin
```

2. Remove the unused staff lists.

Enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin -lang jython -f cleanupUnusedStaffQueryInstances.py  
-server serverName  
[-profile profileName]
```

```
install_root/bin/wsadmin -lang jython -f cleanupUnusedStaffQueryInstances.py  
-node nodeName  
-server serverName  
[-profile profileName]
```

```
install_root/bin/wsadmin -lang jython -f cleanupUnusedStaffQueryInstances.py  
-cluster clusterName  
[-profile profileName]
```

Where:

cluster *clusterName*

The name of the cluster. Required if the business process container is configured for a WebSphere cluster.

node *nodeName*

Optional when specifying the server name. This name identifies the node. The default is the local node.

server *serverName*

The name of the server. Required if the cluster name is not specified.

profileName *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

The number of entries deleted from the database is displayed.

Note: The jacl version of the cleanup unused staff query script, `cleanupUnusedStaffQueryInstances.jacl`, is deprecated. It is available in the `util` subdirectory of the `ProcessChoreographer` directory and it takes the same parameters as described here, but the `-lang jython` option must be omitted.

Administering business processes and human tasks

Business processes and human tasks are deployed and installed as part of an enterprise application. You can use the administrative console or the administrative commands to administer process templates and task templates, and Business Process Choreographer Explorer to work with process instances and task instances. Use Business Process Choreographer Observer to report on business processes and human tasks.

About business processes

A business process is a set of business-related activities that are invoked in a specific sequence to achieve a business goal.

A process that is defined in the Web Services Business Process Execution Language (WS-BPEL) comprises:

- The activities that are the individual business steps 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.
 - Structured activities are activities that contain other activities.
- The partner links, also known as interface partners or reference partners, that specify external entities and partners that interact with the process or vice versa using WSDL interfaces.
- The variables that store 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 that is contained in messages that are exchanged with the process.
- Fault handlers that deal with exceptional situations that can occur when a business process runs.
- Event handlers that receive and process unsolicited messages in parallel to the normal execution process.
- Compensation handlers that specify the compensation logic for a single activity or a group of activities.

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

Business Process Choreographer also supports the IBM® extensions to the BPEL language, such as:

- Human task activities for human interaction. These inline participating tasks can be almost any step in the business process that involves a person, for example, completing a form, approving a document or drawing, writing a letter, and so on.
- Script activities for running inline Java code. The Java code can access all of the BPEL variables, correlation properties, and partner links, as well as process and activity contexts.
- Information service activities to directly access WebSphere Information Server and relational databases.
- Valid-from timestamps for process model versioning.
- Common Event Infrastructure (CEI) logging.

- Explicit checkpointing to support multiple activities in one transaction.
- Timeouts for activities.

Business process types

Business processes can be either long-running or microflows.

Long-running processes

A long-running business process is interruptible, and each step of the process can run in its own physical transaction. Long-running business processes can wait for external stimuli. Examples of external stimuli are events that are sent by another business process in a business-to-business interaction, responses to asynchronous invocations, or the completion of a human task.

A long-running process has the following characteristics:

- Runs as several transactions
- Consists of synchronous and asynchronous services
- Stores each intermediate process state, which makes the process forward-recoverable

Microflows

A microflow runs in one physical thread from start to finish without interruption. Microflows are sometimes referred to as non-interruptible business processes. Microflows can have different transactional capabilities. A microflow can run within a global transaction or as part of an activity session.

A microflow has the following characteristics:

- Runs in one transaction
- Normally runs for a short time
- Does not store run-time values in the database
- Consists of only synchronous services and non-interruptible subprocesses, which means that a microflow cannot contain:
 - Human tasks
 - Wait activities
 - Multiple initiating receive activities
 - Non-initiating receive activities

A microflow should not invoke the following services or activities:

- Long-running services
- Activities bound to asynchronous protocols

Additional BPEL activities

Business Process Choreographer includes support for additional activities that are extensions to the Web Services Business Process Execution Language (BPEL) invoke activity.

These additional activities include the Java snippet activity and the information service activity.

Java snippet activity

A Java snippet activity (script extensions of the BPEL invoke activity) allows you to specify Java code as part of the process implementation. This Java code has access to the enclosing BPEL environment, for example, it can work with BPEL variables, partner links, correlation sets, and custom properties. These objects are either data objects or Java objects that represent simple types. You can use BPEL variables in

Java snippets in the same way as local Java variables in the enclosing Java method.

Information service activity

An information service activity provides direct interaction with IBM Information Server and relational databases. The following kinds of information service activity are available:

Information server

With this activity kind, information services that were created in Information Server can be invoked from a business process.

SQL snippet

The SQL snippet allows you to process SQL statements, including Data Definition Language (DDL) statements, from a business process. For example, an SQL select statement can issue queries and assign query results to process variables by reference (set reference). These set references can be used by other activities in the process without moving all of the related data into the process space.

Retrieve set

The retrieve set allows data that is defined by a set reference to be loaded into a process variable. The data is returned as a business object.

Atomic SQL sequence

The atomic SQL sequence allows you to define multiple SQL snippets and retrieve set statements in an information server activity. The statements are processed in one transaction in the order in which you defined them.

Life cycle management and versioning behavior of subprocesses

A process that is started by another process is known as a *subprocess*. The way in which the life cycle of subprocesses can be managed and the versioning behavior of subprocesses depend on how these processes are modeled.

For modularity and reuse, it often makes sense to apply the programming concept of encapsulation to business process modeling, that is to implement one or more steps of the business logic as a separate process and to invoke this process from the main process. A subprocess can also start another process. This can lead to an arbitrarily deep hierarchy of process instances. When these processes are deployed, all of the process templates in the process-to-process relationship must be deployed to the same Business Process Choreographer database.

Life cycle management

A subprocess can have a peer-to-peer relationship or a parent-child relationship with the calling process. This relationship determines the behavior of a subprocess when an action that manages the process life cycle is invoked for the calling process. The life cycle actions comprise suspend, resume, terminate, delete, and compensation. Actions that manage the process life cycle can be taken only on top-level process instances.

The calling process-subprocess relationship is determined by the autonomy attribute of the subprocess. This attribute can have one of the following values:

Peer A peer process is considered to be a *top-level process*. A top-level process is a process instance that either is not invoked by another process instance or is invoked by another process instance, but it has peer autonomy. If the subprocess is part of a peer-to-peer relationship, life cycle actions on the calling process instance are not applied to the subprocess instance.

However, for long-running processes with a creating operation that implements a one-way interface, the value of the autonomy attribute is automatically set to peer during runtime. If the autonomy attribute is set to child, this value is ignored at runtime.

Child If the subprocess is part of a parent-child relationship, life cycle actions on the parent process instance are applied to the subprocess instance. For example, if the parent process instance is suspended, all of the subprocess instances with child autonomy are suspended, too.

A microflow always runs as a child process. However, if there is another component between the two processes, it might prevent a parent-child relationship from being established, for example, an interface map component that is wired between the two process components.

Versioning behavior

The version of a process that is used is determined by whether the process is used in an *early-binding* scenario or a *late-binding* scenario.

Early binding

In an early-binding scenario, the decision on which version of the subprocess is invoked is made during deployment. The calling process invokes a dedicated, statically-bound subprocess according to the Service Component Architecture (SCA) wiring. The versioning of the process is ignored.

An example of early-binding is an SCA wire. For example, if you wire a stand-alone reference to a process component, every invocation of the process using this reference is targeted to the specific version that is represented by the process component.

Late binding

In a late-binding scenario, the decision on which subprocess template is invoked happens when the calling process instance needs to invoke the subprocess. In this case, the version of the subprocess that is currently valid is used. A newer version of a process supersedes all of the previous versions of the process. Existing process instances continue to run with the process template with which they were associated when they started. This leads to the following categories of process templates:

- Process templates that are no longer current might still be valid for existing long-running process instances
- Current process templates are used for new process instances
- Process templates that become valid in the future according to their valid-from date and time.

To apply late-binding when a subprocess is invoked, the parent process must specify the name of the subprocess template from which the valid subprocess is to be chosen at the reference partner. The valid-from attribute of the process is used to determine the subprocess template that is currently valid. Any SCA wiring is ignored.

An example of late-binding is when a new process is invoked in Business Process Choreographer Explorer. The instance that is created is always based on the most recent version of the process template with a valid-from date that is not in the future.

When a new version of a process model is created and the existing process model is used in late-binding scenarios, you must avoid making changes that will lead to

compatibility problems when the new version of the process becomes valid and, for example, a parent process invokes an instance of the new version of the subprocess. The following are incompatible changes that you must avoid:

- Modifying the correlation sets
- Changing any interface that is used by the parent process to communicate with the subprocess

Data exchange between business processes and services

A business process can consume service component architecture (SCA) services or it can be consumed by other SCA services. The way in which Web Services Description Language (WSDL) message data is exchanged between the SCA service and the process depends on how the process was modeled.

A business process consumes a service

The consumption of a service in a business process is implemented using a Business Process Execution Language (BPEL) invoke activity in the process model. The data that is passed to the SCA service is retrieved from one or more BPEL variables. Usually, the data is passed by value, which means that the invoked service works with a copy of the data.

Under certain circumstances, data can be passed by reference. Passing data by reference can help to improve the performance of business processes.

If **all** of the following conditions are met, the data is passed by reference to the business process:

- The invocation of the service is synchronous.
- The BPEL process and the invoked service are in the same module.
- The data is exchanged in one of the following ways:
 - One or more BPEL variables are declared using XML schema types or elements. The WSDL message parts are mapped individually between the service invocation and the variables.

```
<variable name="inputPart1Var" type="ws:inputPart1Type">  
<variable name="inputPart2Var" type="ws:inputPart2Type">
```

The Web service activity uses the parameter extension to refer to the BPEL variables. In the SCA interaction, the WSDL is treated as a wrapper for the data that is passed by reference.

```
<invoke ....>  
  <wpc:input>  
    <wpc:parameter name="ws:inputPart1" variable="inputPart1Var"/>  
    <wpc:parameter name="ws:inputPart2" variable="inputPart2Var"/>  
    ...  
  </wpc:input>  
</invoke ....>
```

- One or more BPEL variables are declared using XML schema types or elements. The Web service interaction complies with the document-literal wrapped style; parameter elements are mapped between the wrapper document and the variables.

```
<variable name="inputParm1Var" type="ws:inputParm1ElemType">  
<variable name="inputParm2Var" type="ws:inputParm2ElemType">
```

The Web service activity uses the parameter extension to refer to the BPEL variables. This is the default behavior for processes that are created in WebSphere Integration Developer. In the SCA interaction, the wrapper holds the parameters that are passed by reference.

```

<invoke ....>
  <wpc:input>
    <wpc:parameter name="ws:inputParm1" variable="inputParm1Var"/>
    <wpc:parameter name="ws:inputParm2" variable="inputParm2Var"/>
    ...
  </wpc:input>
</invoke ....>

```

If the invoked service modifies the data, these changes are applied to the corresponding BPEL variables. However, as a best practice the invoked service should not update the data because any changes that are made to the data are not persistent. For long-running processes the changes are discarded when the current transaction commits, and for microflows the changes are discarded when the process ends. In addition, an event is not generated when the variable is updated by the invoked service.

A business process is consumed by a service

A business process that is consumed by other services contains receive activities, pick activities, or event handlers in the process model. The data that is passed to the process is written to one, or more BPEL variables. Usually, the data is passed by value, which means that the process works with a copy of the data.

However, if **all** of the following conditions are met, the data is passed by reference:

- The invocation of the business process is synchronous.
- The service and the invoked business process are in the same module.
- The data is exchanged in one of the following ways:
 - One or more BPEL variables are declared using XML schema types or elements. The WSDL message parts are mapped individually between the service invocation and the variables.

```

<variable name="outputPart1Var" type="ws:outputPart1Type">
<variable name="outputPart2Var" type="ws:outputPart2Type">

```

The activity uses the parameter extension to refer to the BPEL variables. In the SCA interaction, the WSDL is treated as a wrapper for the data that is passed by reference. For a receive activity, the corresponding BPEL snippet might look like the following example:

```

<receive ....>
  <wpc:output>
    <wpc:parameter name="ws:outputPart1" variable="outputPart1Var"/>
    <wpc:parameter name="ws:outputPart2" variable="outputPart2Var"/>
    ...
  </wpc:output>
</receive ....>

```

- One or more BPEL variables are declared using XML schema types or elements. The Web service interaction complies with the document-literal wrapped style; parameter elements are mapped between the wrapper document and the variables.

```

<variable name="outputParm1Var" type="ws:outputParm1ElemType">
<variable name="outputParm2Var" type="ws:outputParm2ElemType">

```

The activity uses the parameter extension to refer to the BPEL variables. This is the default behavior for processes that are created in WebSphere Integration Developer. In the SCA interaction, the wrapper holds the parameters that are passed by reference. For a receive activity, the corresponding BPEL snippet might look like the following example:

```

<receive ....>
  <wpc:output>
    <wpc:parameter name="ws:outputParm1" variable="outputParm1Var"/>
    <wpc:parameter name="ws:outputParm2" variable="outputParm2Var"/>
    ...
  </wpc:output>
</receive ....>

```

If the invoked process modifies the BPEL variables, the input data from the calling service is also modified.

About human tasks

A human task is a component that involves a person interacting with a service or another person.

The interaction can be initiated either by a person or by an automated service. A service that is initiated by a person can be either an automated implementation or a service that is provided by another person. A human task that is invoked by an automated service can be replaced easily by an automated implementation, and vice versa.

Tasks can be used to implement staff activities in business processes that require human interactions, such as manual exception handling and approvals. All other exception handling is modeled natively in Web Services Business Process Execution Language (WS-BPEL, abbreviated to BPEL), by using faults and fault handlers, or compensation.

Who can interact with a task can be determined using one of the supported staff directories. Work items are created for users who have a reason to view or interact with the task.

Business Process Choreographer supports the following types of staff directories:

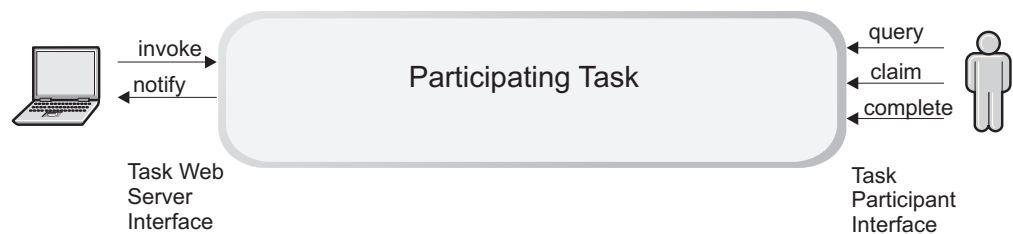
- Lightweight Directory Access Protocol (LDAP)
- WebSphere user registry

Kinds of human tasks

The kinds of human tasks are as follows:

Participating tasks

Support Web-service-to-person interactions, which enable a person to implement a service. For example, a participating task can be a human task activity in a business process.

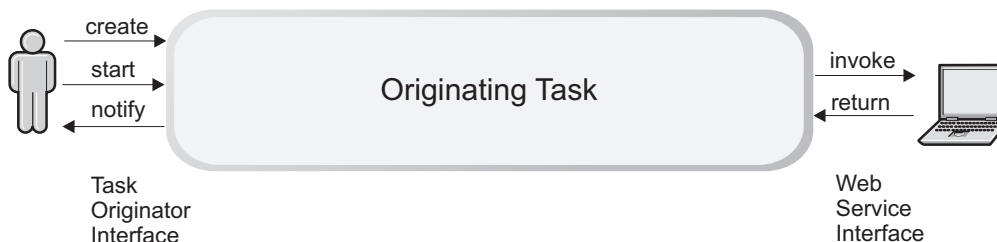


Administrative tasks

Administrative tasks are similar to participating tasks, except that they are used by administrators to solve technical problems that occur in processes. Currently, you can use administration tasks for business processes only.

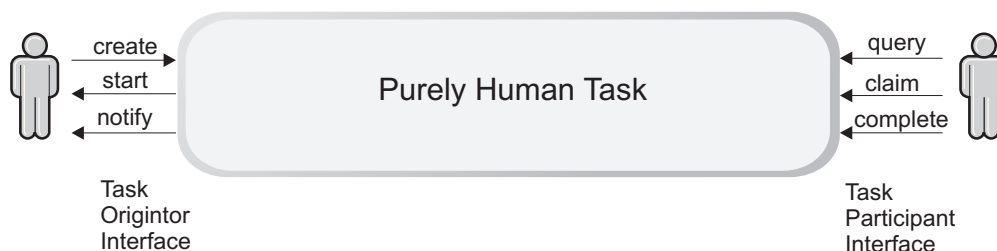
Originating tasks

Support person-to-computer interactions, which enables people to create and start services through a graphical user interface. For example, a user can start a business process, or send it an event by means of an originating task.



Purely human tasks

Support person-to-person interactions, which enable a person to share work with other people in a structured and controlled way. Purely human tasks do not interact with business processes or other Web services.



Relationship of human tasks to business processes

A human task can be related to a business process in one of the following ways:

Inline tasks

An inline task is defined as a part of the business process. It is not visible as an Service Component Architecture (SCA) component, and it can share context data with the process.

Stand-alone tasks

A stand-alone task is an SCA component that implements human interaction as a service (participating task), leverages the person-to-service interaction with a graphical user interface (originating task), or supports the structured collaboration between people (purely human task). Task components can be combined with other services, including business processes.

The following table shows the differences between these two implementation types.

Inline tasks	Stand-alone tasks
Part of the business process.	Independent of the business process. This implementation can also be used in scenarios that do not include business processes.
The life cycle of the task is usually controlled by the process.	The life cycle is independent of the process.

Inline tasks	Stand-alone tasks
A participating task is a human task activity in a process.	A participating task is an invoke activity in the process.
Inline tasks can access process context data, for example, variables, staff assignments, or custom properties.	Stand-alone tasks cannot access process context data.
Task descriptions, display names, and documentation for participating and originating tasks support only one language.	Task descriptions, display names, and documentation for participating and originating tasks support multiple languages.
Inline tasks are not visible as SCA components, and therefore they are not reusable (cannot be wired).	Stand-alone tasks are reusable. Participating and originating tasks are visible as SCA components (can be wired).
Supported kinds of tasks: participating tasks, originating tasks, and administrative tasks.	Supported kinds of tasks: participating tasks, originating tasks, and purely human tasks.

Subtasks

A subtask is an additional unit of work that is split out from a parent task. The subtask model can be selected from a template or it can be defined at runtime. Input data is provided by the person that creates or starts the subtask. The parent task waits until all of its subtasks are finished. The owner or editor of the parent task consolidates the subtask output data, and then completes the parent task.

If the subtask fails to complete within a specified period of time, the parent task can be escalated. The escalation indicates that the parent task is still waiting for subtasks to complete.

Subtasks can be purely human tasks or originating tasks.

Follow-on tasks

A follow-on task is a task that is created to complete an existing task. The follow-on task model can be selected from a template or it can be defined at runtime. Input data is provided by the person that creates or starts the follow-on task. The output and fault message types of the follow-on task must be the same as those of the previous task. The previous task is put into the forwarded state and it does not report completion to the service or person that invoked it.

When the follow-on task finishes, it reports its output or fault data to the service or person that invoked the original task. Escalations of the previous task continue to run and escalate. The follow-on task has its own escalations.

Follow-on tasks can be only purely human tasks.

Escalations

An escalation is a course of action that is executed when a task is not completed satisfactorily within a specific period of time. For example, if tasks are not claimed or are not completed within a defined time limit. You can specify one, or more, escalations for a task. These escalations can be started either in parallel or as a chain of escalations.

Escalations are initialized when the associated task reaches a certain state in its life cycle. After a defined duration, the task state is verified, and if it does not meet the modeled expectation, the escalation action is invoked. The following escalation actions are supported:

- Create work items for a set of users
- Send e-mails to the designated recipients
- Send notification events to registered consumers

Administering process templates and process instances

Use the administrative console or the administrative commands to administer process templates. Use Business Process Choreographer Explorer to work with process instances.

Process templates define business processes within an enterprise application. When an enterprise application that contains process templates is installed, deployed, and started, the process templates are put into the started state. You can use the administrative console or the administrative commands to stop and start process templates. The process templates that are started are shown in Business Process Choreographer Explorer.

A process instance can be a long-running process or a microflow. Use Business Process Choreographer Explorer to display information about process templates and process instances, or act on process instances. These actions can be, for example, starting process instances; and for long-running processes other process life cycle actions, such as suspending, resuming, or terminating process instances; or repairing activities.

Business process administration—frequently asked questions

Answers to a set of frequently asked questions about administering business processes.

- “What happens if a process template is in the started state, but the application it belongs to is in the stopped state?”
- “How do I stop new process instances being created?”
- “What happens to running instances when a newer process template becomes valid?” on page 149
- “What happens to a running instance if the template it was created from is stopped?” on page 149
- “How can I tell if any process instances are still running?” on page 149
- “Why can’t I stop a business process application if it has any process instances?” on page 149

What happens if a process template is in the started state, but the application it belongs to is in the stopped state?

If a currently valid process template is in the started state, but the application is in the stopped state, no new process instances are created from the template. Existing process instances cannot be navigated while the application is in the stopped state.

How do I stop new process instances being created?

Using the administrative console, select a process template, and click **Stop**. This action puts the process template into the stopped state, and no more instances are created from the template. After the template stops, any attempts to create a

process instance from the template result in an `EngineProcessModelStoppedException` error.

What happens to running instances when a newer process template becomes valid?

If a process template is no longer valid, this fact has no effect on running instances that were instantiated from the template. Existing process instances continue to run to completion. Old and new instances run in parallel until all of the old instances have finished, or until they have been terminated.

What happens to a running instance if the template it was created from is stopped?

Changing the state of a process template to 'stopped' only stops new instances being created. Existing process instances continue running until completion in an orderly way.

How can I tell if any process instances are still running?

Log on to the Business Process Choreographer Explorer as a process administrator, and go to the Process Instances Administered By Me page. This page displays any running process instances. If necessary, you can terminate and delete these process instances.

Why can't I stop a business process application if it has any process instances?

For a process instance to run, its corresponding application must also be running. If the application is stopped, the navigation of the process instance cannot continue. For this reason, you can only stop a business process application if it has no process instances.

Authorization roles for business processes

Actions that you can take on business processes depend on your authorization role. This role can be a J2EE role or an instance-based role.

A role is a set of employees who share the same level of authority. Java 2 Platform, Enterprise Edition (J2EE) roles are set up when the business process container is configured. Instance-based roles are assigned to processes and activities when the process is modeled. Role-based authorization requires that global security is enabled in WebSphere Application Server.

J2EE roles

The following J2EE roles are supported:

- `J2EE BPSystemAdministrator`. Users assigned to this role have all privileges. This role is also referred to as the system administrator for business processes.
- `J2EE BPSystemMonitor`. Users assigned to this role can view the properties of all business process objects. This role is also referred to as the system monitor for business processes.

You can use the administrative console to change the assignment of users and groups to these roles.

Setting up Roles using RACF security: These RACF® permissions apply when the following security fields are specified:

- **com.ibm.security.SAF.authorization= true**
 RDEFINE EJBROLE BPESystemAdministrator UACC(NONE)
 PERMIT BPESystemAdministrator CLASS(EJBROLE) ID(userid) ACCESS(READ)
 RDEFINE EJBROLE BPESystemMonitor UACC(NONE)
 PERMIT BPESystemMonitor CLASS(EJBROLE) ID(userid) ACCESS(READ)
- **com.ibm.security.SAF.delegation= true**
 RDEFINE EJBROLE JMSAPIUser UACC(NONE) APPLDATA(' userid')

You can use Security Authorization Facility (SAF)-based authorization (for example, using the RACF EJBROLE profile) to control access by a client to Java 2 Platform, Enterprise Edition (J2EE) roles in EJB and Enterprise applications, including the business process container. For more information on using SAF, see System Authorization Facility for role-based authorization in the WebSphere Application Server for z/OS information center.

Instance-based roles

A process instance or an activity is not assigned directly to a staff member in the process model, instead it is assigned to one of the available roles. Any staff member that is assigned to an instance-based role can perform the actions for that role. The association of users to instance-based roles is determined at runtime using staff resolution.

The following instance-based roles are supported:

- For processes: reader, starter, administrator
- For activities: reader, editor, potential starter, potential owner, owner, administrator

These roles are authorized to perform the following actions:

Role	Authorized actions
Activity reader	View the properties of the associated activity instance, and its input and output messages.
Activity editor	Actions that are authorized for the activity reader, and write access to messages and other data associated with the activity.
Potential activity starter	Actions that are authorized for the activity reader. Members of this role can send messages to receive or pick activities.
Potential activity owner	Actions that are authorized for the activity reader. Members of this role can claim the activity.
Activity owner	Work on and complete an activity. Members of this role can transfer owned work items to an administrator or a potential owner.
Activity administrator	Repair activities that are stopped due to unexpected errors, and force terminate long-running activities.
Process starter	View the properties of the associated process instance, and its input and output messages.
Process reader	View the properties of the associated process instance and its input and output messages. Process readers can also view the properties, and input and output messages for any activities that are contained in the process instance, but they cannot see any information about its subprocesses.

Role	Authorized actions
Process administrator	Members of this role can administer process instances and intervene in a process that has started; create, delete, and transfer work items. Members of this role also have activity administrator authorization.

Do not delete the user ID of the process starter from your user registry if the process instance still 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>

Stopping and starting process templates with the administrative console

You can use the administrative console to start and stop each installed process template individually.

If global security is enabled, verify that your user ID has operator authorization. The server on which the application is installed must be running.

You must stop a process template, for example, before you can uninstall the business process application to which it belongs. The following steps describe how to use the administrative console to stop and start process templates.

1. Select the application that you want to manage.
In the navigation pane of the administrative console, click **Applications** → **Enterprise Applications**, and then the application that you want to manage.
2. In the Configuration page for the enterprise application under **Related Items**, click **EJB Modules**, and then an Enterprise JavaBeans module.
3. In the Configuration page for the EJB module under **Additional Properties**, click **Business processes**, and then a process template.
4. Stop the process template.
Existing instances of the process templates continue to run until they end normally. However, you cannot create process instances from a stopped template.
5. Start the process template that is in the stopped state.

Stopping and starting process templates with administrative commands

Administrative commands provide an alternative to the administrative console for stopping and starting process templates.

If global security is enabled, verify that your user ID has operator authorization.

You must stop a business template, for example, before you can uninstall the business process application to which it belongs. The following steps describe how to use the administrative commands to stop and start process templates.

1. Change to the Business Process Choreographer samples directory. Type the following:
`cd install_root/ProcessChoreographer/sample`
2. Stop the process template.
`install_root/bin/wsadmin -f bpcTemplates.jacl
-stop application_name`

Where *application_name* is the name of the application to which the template belongs.

Existing instances of the process templates continue to run until they end normally. When the application stops, you cannot create process instances from the stopped templates.

3. Start the process template.

```
install_root/bin/wsadmin -f bpcTemplates.jacl  
                        -start application_name
```

The process template starts. You can use Business Process Choreographer Explorer to start process instances from the process template.

Managing the process life cycle

After a process starts, it goes through various states until it ends. As a process administrator, you can take various actions on a process throughout its life cycle.

Starting a new process instance:

You can start a new process instance from any of the process templates that you are authorized to use.

All of the installed process templates are shown in the list of process templates in Business Process Choreographer Explorer. To start a new process instance, complete the following steps.

1. Display the process templates that you are authorized to use.

Click **My Process Templates** under Process Templates in the navigation pane.

2. Select the check box next to the process template and click **Start Instance**.

This action displays the Process Input Message page.

If the process has more than one operation, this action displays a page that contains all of the available operations. Select the operation that is to start the process instance.

3. Provide the input data to start the process instance.

If the process is a long-running process, you can type in a process instance name. If you do not specify a name, a system-generated name is assigned to the new process instance.

Complete the input for the process input message.

4. To start the process, click **Submit**.

The process instance is started. If the business process contains an activity that requires human interaction, tasks are generated for all of the potential owners. If you are one of these potential owners, this task appears in the list on the My Tasks page.

If the process is a long-running process, a process output message is displayed immediately after the process finishes. Not all processes have output messages, for example, if the process implements a one-way operation, an output message is not displayed.

Monitoring the progress of a process instance:

You can monitor the progress of a process instance to determine whether you need to take action so that the process can run to completion.

In Business Process Choreographer Explorer, complete the following steps to monitor the progress of a process instance.

1. Display a list of process instances.
For example, click **Administered By Me** under Process Instances in the navigation pane.
2. Select the check box next to the process instance and click **View Process State**.
This action displays the Process State page. This page shows the activities, the links including the transition and join conditions for the links, the fault handlers, the compensation handlers, and the event handlers that are defined for the process. Activities that are shown in bold are defined as business relevant in the process model. State information is shown for these activities.
3. To act on an activity, click the activity.
This action displays the Activity page where you can perform these actions.

Suspending and resuming process instances:

You can suspend a long-running, top-level process instance. You might want to do this, for example, so that you can configure access to a back-end system that is used later in the process, or to fix a problem that is causing the process instance to fail. When the prerequisites for the process are met, you can resume running the process instance.

To suspend and resume process instances, you must have process administrator authorization.

To suspend a process instance, the process instance must be in either the running or failing state. To resume a process, the process instance must be in the suspended state.

To suspend or resume a process instance, complete the following steps in Business Process Choreographer Explorer.

1. Display a list of process instances.
For example, click **Administered By Me** under Process Instances in the navigation pane.
2. Suspend the process.
Select the check box next to the process instance and click **Suspend**.
This action suspends the specified top-level process instance. The process instance is put into the suspended state. Subprocesses with the autonomy attribute set to `child` are also suspended if they are in the running, failing, terminating, or compensating state. However, you can still complete any active activities and tasks that belong to the process instance.
3. Resume the process instance.
Select a process instance that is in the suspended state and click **Resume**. The process instance and its subprocess are put into the states they had before they were suspended, for example, running. The process instance and its subprocesses resume.

Terminating process instances:

You might want to terminate a process instance, for example, 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, you must have process administrator authorization.

In Business Process Choreographer Explorer, complete the following steps to terminate a process instance. If compensation is defined for the business process model, you can choose to terminate the process instance with compensation.

1. Display the process instances that you can administer.
Click **Administered By Me** under Process Instances in the navigation pane.
2. Select the check box next to the process instance that you want to stop.
 - To terminate the process instance with compensation, click **Compensate**.
This action terminates the process instance and starts compensation processing.
 - To terminate the process instance without compensation, click **Terminate**.
This action stops the process instance immediately without waiting for any outstanding activities or tasks. Process instances that are terminated are not compensated.

Deleting process instances:

Process templates can be modeled so that process instances are not automatically deleted when they complete. You can explicitly delete these process instances after they complete.

To delete a process instance, you must have process administrator authorization. The process instance must be in the finished or terminated state.

Completed processes instances are automatically deleted from the Business Process Choreographer database if the corresponding property is set for the process template in the process model.

You might want to keep process instances in your database, for example, to query data from process instances that are not written to the audit log, or if you want to defer the deletion of processes to off-peak times. However, old process instance data that is no longer needed can impact disk space and performance. Therefore, you should regularly delete process instance data that you no longer need or want to maintain. Make sure that you run this maintenance task at off-peak times.

You can delete completed process instances using either Business Process Choreographer Explorer, for example, to delete individual process instances, or the `deleteCompletedProcessInstances` administrative script to delete several process instances at once.

In Business Process Choreographer Explorer, complete the following steps to delete a process instance.

1. Display the process instances that you administer.
Click **Administered By Me** under Process Instances in the navigation pane.
2. Select the process instance that you want to delete and click **Delete**.

This action deletes the selected process instance from the database.

Repairing processes and activities

If the process runs into problems, you can analyze the process and then repair the activities.

Business Process Choreographer Explorer provides various views for the process administrator to monitor the processes that are currently running.

- To view process instances with activities in the stopped state, click **Critical Processes** under Process Instances in the navigation pane.
- To monitor the progress of a specific process instance, click **View Process State** in any view that displays a list of process instances.

You can now take action to repair the pending activities.

Restarting activities:

If you have repaired an activity, you can restart it using new input data.

The activity must be in the stopped state and the associated process instance must be in the running state.

To restart an activity, complete the following steps in Business Process Choreographer Explorer.

1. Navigate to the Activity page for the activity and click **Restart**.
2. Specify the input data that is needed to start the activity again.
If the process is to continue if an error occurs when the activity starts again, select **Continue on Error**.
3. Click **Restart**.

Forcing the completion of activities:

If you are aware that an activity is not going to complete in a timely manner, for example, because the invoked service is no longer available, you can force the completion of the activity so that the process flow can continue.

Generally, the activity must be in the stopped state. However, if the activity is a staff activity, it can also be in either the ready or the claimed state. The associated process instance must be in the running state.

To force the completion of an activity, complete the following steps in Business Process Choreographer Explorer.

1. Navigate to the Activity page for the activity and click **Force Complete**.
2. Specify the data that is needed to complete the activity.
3. Click **Force Complete** again.

Administering compensation for microflows:

When a microflow runs, it can encounter problems. For these situations, compensation might have been defined for the process in the process model. Compensation allows you to undo previous completed steps, for example, to reset data and states so that you can recover from these problems.

For microflows to be compensated, the compensation service must be started in the administrative console.

If a compensation action for a microflow fails, the process administrator must intervene to resolve the problem.

In Business Process Choreographer Explorer, complete the following steps to administer failed compensation actions.

1. Display a list of the compensation actions that failed.

Click **Failed Compensations** under Process Instances in the navigation pane.

The Failed Compensations page is displayed. This page contains information about why the named compensation action failed. This information can help you to decide what actions to take to correct the failed compensation.

2. Select the check box next to the activity and then click one of the available actions.

The following administrative actions are available:

Skip Skips the current compensating action and continues with compensating the microflow. This action might result in a non-compensated activity.

Retry If you have taken action to correct the failed compensation action, click **Retry** to try the compensation action again.

Stop Stops the compensation processing.

Compensation in business processes:

Compensation is the means by which operations in a process that have successfully completed can be undone.

Compensation processing starts because an error occurs in a running process instance for which compensation is defined in the process model. Compensation reverses the effects of operations that were committed up to when the error occurred to get back to a consistent state.

You can define compensation for long-running processes and for microflows in your process model.

Compensation for long-running processes

Compensation for long-running processes is also known as *business-level compensation*. This type of compensation is defined on the scope level. This means that either part of the process, or the entire process can be compensated.

Compensation is triggered by fault handlers or the compensation handler of a scope or a process; compensation is another navigation path of the process.

A long-running process automatically compensates child processes that have successfully completed when the enclosing parent scope is compensated. Within a process, only invoke and scope activities that complete successfully are compensated.

Compensation for microflows

Compensation for microflows is also known as *technical compensation*. This type of compensation is triggered when the work unit (the transaction or the activity session) that contains the microflow is rolled back. 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 microflow is a child of a compensable, long-running process, the undo actions of the microflow are made available to the parent process when the microflow completes. It can, therefore, potentially participate in the compensation of the parent process. For these types of microflows, it is a good practice to specify undo actions for all of the activities in the process when you define your process model.

If an error occurs during compensation processing, the compensation action requires manual resolution to overcome the error. You can use Business Process Choreographer Explorer to repair these compensation actions.

Administering task templates and task instances

Use the administrative console or the administrative commands to administer task templates. Use Business Process Choreographer Explorer to work with task instances.

Authorization roles for human tasks

Actions that you can take on human tasks depend on your authorization role. This role can be a J2EE role or an instance-based role.

A role is a set of employees who share the same level of authority. Java 2 Platform, Enterprise Edition (J2EE) roles are set up when the human task container is configured. Instance-based roles are assigned to human tasks and escalations when the task is modeled. Role-based authorization requires that global security is enabled in WebSphere Application Server.

J2EE roles

The following J2EE roles are supported:

- **J2EE TaskSystemAdministrator.** Users assigned to this role have all privileges. This role is also referred to as the system administrator for human tasks.
- **J2EE TaskSystemMonitor.** Users assigned to this role can view the properties of all of the task objects. This role is also referred to as the system monitor for human tasks.

You can use the administrative console to change the assignment of users and groups to these roles.

Setting up Roles using RACF security: These RACF permissions apply when the following security fields are specified:

- **com.ibm.security.SAF.authorization= true**
RDEFINE EJBROLE TaskSystemAdministrator UACC(NONE)
PERMIT TaskSystemAdministrator CLASS(EJBROLE) ID(userid) ACCESS(READ)
RDEFINE EJBROLE TaskSystemMonitor UACC(NONE)
PERMIT TaskSystemMonitor CLASS(EJBROLE) ID(userid) ACCESS(READ)
- **com.ibm.security.SAF.delegation= true**
RDEFINE EJBROLE JMSAPIUser UACC(NONE) APPLDATA('userid')

You can use Security Authorization Facility (SAF)-based authorization (for example, using the RACF EJBROLE profile) to control access by a client to Java 2 Platform, Enterprise Edition (J2EE) roles in EJB and Web applications, including the WebSphere Application Server administrative console application. For more information, see System Authorization Facility for role-based authorization in the WebSphere Application Server for z/OS information center.

Instance-based roles

A task instance or an escalation instance is not assigned directly to a staff member in the task model, instead it is assigned to one of the available roles. Any staff member that is assigned to an instance-based role can perform the actions for that role. The association of users to instance-based roles is determined at runtime using staff resolution.

The following instance-based roles are supported:

- For tasks: potential instance creator, originator, potential starter, starter, potential owner, owner, reader, editor, administrator
- For escalations: escalation receiver

These roles are authorized to perform the following actions:

Role	Authorized actions
Potential instance creator	Members of this role can create an instance of the task. If no potential instance creator is defined for the task template or the application components, then all users are considered to be a member of this role.
Originator	Members of this role have administrative rights until the task starts. When the task starts, the originator has the authority of a reader and can perform some administrative actions, such as suspending and resuming tasks, and transferring work items.
Potential starter	Members of this role can start an existing task instance. If a potential starter is not specified, the originator becomes the potential starter. For inline tasks without a potential starter, the default is everybody.
Starter	Members of this role have the authority of a reader and can perform some administrative actions, such as transferring work items.
Potential owner	Members of this role can claim a task. If no potential owner is defined for the task template or the application components, then all users are considered to be a member of this role. If staff resolution fails for this role, then the administrators are assigned as the potential owners.
Owner	Work on and complete a task.
Reader	View the properties of all of the task objects, but cannot work on them.
Editor	Members of this role can work with the content of a task, but cannot claim or complete it
Administrator	Members of this role can administer tasks, task templates, and escalations.
Escalation receiver	Members of this role have the authority of a reader for the escalation and the escalated task.

Stopping and starting task templates with the administrative console

Use the administrative console to start and stop task templates.

If global security is enabled, verify that user ID has operator authorization.

Task templates define Service Component Architecture (SCA) services that are represented as stand-alone tasks within an enterprise application. When an

enterprise application that contains task templates is installed, deployed, and started, the task templates are put into the start state.

1. Select the application that you want to manage.
In the navigation pane of the administrative console, click **Applications** → **Enterprise Applications**, and then the application that you want to manage.
2. In the Configuration page for the enterprise application under **Related Items**, click **EJB Modules**, and then an Enterprise JavaBeans module.
3. In the Configuration page for the EJB module under **Additional Properties**, click **Human tasks**, and then a process template.
4. To stop the task template, click **Stop**.
5. To start the task template, click **Start**.

Stopping and starting task templates with the administrative commands

Administrative commands provide an alternative to the administrative console for stopping and starting task templates.

If global security is enabled, verify that you are logged with a user ID that has operator authorization.

Task templates define Service Component Architecture (SCA) services that are represented as stand-alone tasks within an enterprise application. When an enterprise application that contains task templates is installed, deployed, and started, the task templates are put into the start state.

1. Change to the Business Process Choreographer samples directory. Type the following:

```
cd install_root/ProcessChoreographer/sample
```

2. Stop the task template.

```
install_root/bin/wsadmin -f bpcTemplates.jacl  
                        -stop application_name
```

Where *application_name* is the name of the application to which the template belongs. Existing instances of the task template continue to run until they end normally.

3. Start the task template.

```
install_root/bin/wsadmin -f bpcTemplates.jacl  
                        -start application_name
```

The task template starts. You can use Business Process Choreographer Explorer to work with task instances associated with the task template.

Creating and starting a task instance

You can create and start a task instance from any of the task templates that you are authorized to use.

All of the installed task templates are shown in the list of task templates in Business Process Choreographer Explorer. To create and start a task instance from a task template, complete the following steps.

1. Display the task templates that you are authorized to use.
Click **My Task Templates** under Task Templates in the navigation pane.
2. Select the check box next to the task template and click **Start Instance**.
This action displays the Task Input Message page.
3. Provide the input data to start the task instance.

4. To start the task instance, click **Submit**.

The task instance is ready to be worked on.

Working on your tasks

To work on a task, you must claim the task and then perform the actions that are needed to complete it.

You can claim a task that is in the ready state if you are a potential owner or the administrator of that task. If you claim a task, you become the owner of that task and are responsible for completing it.

Tasks for which you have the role of reader or editor also appear on your list of tasks.

To claim and complete a task with Business Process Choreographer Explorer, complete the following steps.

1. Display the tasks that have been assigned to you.
Click **Task Instances** → **My Tasks**.
This action displays the My Tasks page, which lists the tasks that have been assigned to you.
2. Claim the task on which you want to work.
Select the check box next to the task and click **Work on**.
This action displays the Task Message page.
3. Provide the information to complete the task.
If you need to interrupt your work, for example, because you need more information from a co-worker to complete the task, click **Save** to save the changes you made.
4. Click **Complete** to complete the task with the information that you provide.

The task that you completed is in the finished state. If you leave the task without completing it, the task remains in the claimed state.

Managing work assignments

After a task has started, you might need to manage work assignments for the task, for example, to better distribute the work load over the members of a work group.

A *work item* is the assignment of a business entity, such as a task or a process instance, to a person or a group of people for a particular reason. The assignment reason allows a person to play various roles in the business process scenario, for example, potential owner, editor, or administrator.

A task instance can have several work items associated with it because different people can have different roles. For example, John, Sarah, and Mike are all potential owners of a task instance and Anne is the administrator; work items are generated for all four people. John, Sarah, and Mike see only their own work items as tasks on their list of tasks. Because Anne is the administrator, she gets her own work item for the task and she can manage the work items generated for John, Sarah, and Mike.

Sometimes, you might need to change a task assignment after a task has been 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.

Transferring tasks that you own:

If you are the owner of a task, you might need to transfer the task to another user, for example, if someone else needs to provide information to complete the task.

In Business Process Choreographer Explorer, complete the following steps to transfer a task that you own.

1. Display the tasks that you own.
Click **My Tasks** under Task Instances in the navigation pane.
2. Select the check box next to the task that you want to transfer and click **Transfer**.
3. Transfer the task.
In the **New Owner** field, specify the user ID of the new task owner, and click **Transfer**. You can transfer the task only to another potential owner of the task or the task administrator.

The transferred task appears on the list of tasks belonging to the new task owner.

Transferring work items for which you are the starter, originator, or administrator of the task:

You might need to change a work assignment after work begins on the task. For example, you might want to transfer a work item to another user if the task owner is on vacation and the task must be completed before this person returns. The way in which you can transfer a work item depends on the role that you have and the state of the task.

To transfer a work item you must have one of the following roles and the task must be in one of the following states.

Role	Task state	Work items can be transferred to the following user roles:
Owner	Claimed	Potential owner, administrator.
Starter	Terminated, expired, finished, failed, or running	Potential starter, administrator.
Originator	Any task state	Potential instance creator, administrator. If the task is in the active state, it can be transferred to any user role.
Administrator	Ready, claimed, terminated, expired, finished, failed, or running	Any user role.

In Business Process Choreographer Explorer, complete the following steps to transfer a work item.

1. Display the task instances that you can administer.
Click **Administered By Me** under Task Instances in the navigation pane.
2. Display the work items for a task instance.
In the Task Instances Administered By Me page, select the check box next to the task instance and click **Work Items**.
3. Transfer the work item.

- a. In the **New Owner** field, specify the user ID of the new work-item owner.
- b. Select one or more work items and click **Transfer**.

The transferred work item with the new work-item owner appears in the list of work items.

Creating work items:

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 a long-running process if the organization has changed since the process started.

To create a work item for a task instance, you must have the appropriate role for the task. If you are the task administrator, you can create work items for the task instance if it is in one of the following states: ready, claimed, running, finished, or failed. If the task instance is derived from a task template, you can also create work items if the task is in the terminated or expired state.

In Business Process Choreographer Explorer, complete the following steps to create a work item.

1. Display the task instances that you administer.
Click **Administered By Me** under Task Instances in the navigation pane.
2. Select the check box next to the task instance for which you want to create a work item and click **Create Work Items**. The Create Work Items page is displayed.
3. Create the work items.
 - a. In the **New Owner** 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 that the assigned person can perform on the new work item.
 - c. Click **Create**.

A work item is created for each role that you specify for the new work-item owner. The new task appears on the list of tasks assigned to this person.

Deleting work items:

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 for the company.

To delete a work item for a task instance, you must have the appropriate role for the task. If you are the task administrator, you can delete the task instance if it is in one of the following states: ready, claimed, running, finished, or failed. If the task instance was derived from a task template, you can also delete the task in the terminated or expired state.

In Business Process Choreographer Explorer, complete the following steps to delete a work item.

1. Display the task instances that you administer.
Click **Administered By Me** under Task Instances in the navigation pane.

2. Display the work items for a task instance.
In the Task Instances Administered By Me page, select a task instance and click **Work Items**.
3. Delete the work items.
Select one or more work items and click **Delete**.

The work items are deleted.

Viewing task escalations

An escalation notifies the escalation receiver that a user might have problems completing their assigned task on time.

When a task becomes overdue, it might result in an escalation. An escalation can result in the following actions:

- A new work item is created, for example, for a manager to take action to support the resolution of the problem.
- If you specified e-mail settings when you configured the human task container, an e-mail is sent to a designated person to inform them about the escalated task.
- An event notification handler is called.

To view escalations, click **My Escalations** under Task Instances.

- To view information about an escalation, click the escalation ID.
- To view information about an escalated task, click the task name.

Sending e-mails for escalations:

When a task becomes overdue, it might result in an escalation. You can set up your system to send e-mails to designated people to inform them about the escalation.

Ensure that your Lightweight Directory Access Protocol (LDAP) configuration contains the e-mail addresses of the people that need to be notified about an escalation.

1. In WebSphere Integration Developer, perform the following actions for the task in the human task editor.
 - a. Under the task settings in the **Details** tab of the properties area, edit the value of the **JNDI name of staff plugin configuration** field.
Set the Java Naming and Directory Interface (JNDI) name to `bpe/staff/sampleldapconfiguration`, or the LDAP provider configuration JNDI name of your choice.
 - b. Under the escalation settings in the **Details** tab of the properties area, set the value of the **Notification type** field to E-mail.
 - c. Specify text for the body of the e-mail that is sent for the escalation.
If you do not specify any text, the default message text is used.
2. In WebSphere Process Server, perform the following actions.
 - a. Ensure that the simple mail transfer protocol (smtp) host is set.
In the administrative console, go to **Mail Providers** → **Built-in Mail Provider** → **Mail Sessions** → **HTMMailSession_server** to check this setting.
The smtp host is defined on a cell level.
 - b. Ensure that the sender e-mail address that you specify when you configure the human task container is a valid e-mail address.

If a problem occurs with escalation e-mails, check the SystemOut.log file for error messages.

Reporting on business processes and activities

During the processing of business processes and activities, an event is generated when the process, activity, or task changes state. These events are stored and made available for creating reports using Business Process Choreographer Observer, for example, to analyze process bottlenecks, or to evaluate the reliability of a service that is called from an activity.

You can work with predefined reports or create user-defined reports for processes and activities.

Predefined reports

Predefined lists and charts provide a drill-down approach to get you to event and state information. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

User-defined reports for processes and activities

User-defined process and activity reports are more flexible than the predefined lists and charts. In addition, you can store and retrieve your report definitions. For process reports, you can get information about the activities that belong to the process instances. For activity reports, you can also get information about the process instances involved.

Chapter 5. Developing

Developing client applications for business processes and tasks

You can use a modeling tool to build and deploy business processes and tasks. These processes and tasks are interacted with at runtime, for example, a process is started, or tasks are claimed and completed. You can use Business Process Choreographer Explorer to interact with processes and tasks, or the Business Process Choreographer APIs to develop customized clients for these interactions.

These clients can be Enterprise JavaBeans™ (EJB) clients, Web service clients, or Web clients that exploit the Business Process Choreographer Explorer JavaServer Faces (JSF) components. Business Process Choreographer provides Enterprise JavaBeans (EJB) APIs and interfaces for Web services for you to develop these clients. The EJB API can be accessed by any Java application, including another EJB application. The interfaces for Web services can be accessed from either Java environments or Microsoft® .Net environments.

Developing EJB client applications for business processes and human tasks

The EJB APIs provide a set of generic methods for developing EJB client applications for working with the business processes and human tasks that are installed on a WebSphere Process Server.

With these Enterprise JavaBeans (EJB) APIs, you can create client applications to do the following:

- Manage the life cycle of processes and tasks from starting them through to deleting them when they complete
- Repair activities and processes
- Manage and distribute the workload over members of a work group

The EJB APIs are provided as two stateless session enterprise beans:

- BusinessFlowManagerService interface provides the methods for business process applications
- HumanTaskManagerService interface provides the methods for task-based applications

For more information on the EJB APIs, see the Javadoc in the `com.ibm.bpe.api` package and the `com.ibm.task.api` package.

The following steps provide an overview of the actions you need to take to develop an EJB client application.

1. Decide on the functionality that the application is to provide.
2. Decide which of the session beans that you are going to use.

Depending on the scenarios that you want to implement with your application, you can use one, or both, of the session beans.

3. Determine the authorization authorities needed by users of the application.

The users of your application must be assigned the appropriate authorization roles to call the methods that you include in your application, and to view the objects and the attributes of these objects that these methods return. When an

instance of the appropriate session bean is created, WebSphere Application Server associates a context with the instance. The context contains information about the caller's principal ID, group membership list, and roles. This information is used to check the caller's authorization for each call.

The Javadoc contains authorization information for each of the methods.

4. Decide how to render the application.

The EJB APIs can be called locally or remotely.

5. Develop the application.

- a. Access the EJB API.
- b. Use the EJB API to interact with processes or tasks.
 - Query the data.
 - Work with the data.

Accessing the EJB APIs

The Enterprise JavaBeans (EJB) APIs are provided as two stateless session enterprise beans. Business process applications and task applications access the appropriate session enterprise bean through the home interface of the bean.

The BusinessFlowManagerService interface provides the methods for business process applications, and the HumanTaskManagerService interface provides the methods for task-based applications. The application can be any Java application, including another Enterprise JavaBeans (EJB) application.

Accessing the remote session bean:

An EJB client application accesses the appropriate remote session bean through the home interface of the bean.

The session bean can be either the BusinessFlowManager session bean for process applications or the HumanTaskManager session bean for task applications.

1. Add a reference to the remote session bean to the application deployment descriptor. Add the reference to one of the following files:
 - 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

The reference to the remote home interface for process applications is shown in the following example:

```
<ejb-ref>
  <ejb-ref-name>ejb/BusinessFlowManagerHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
  <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
```

The reference to the remote home interface for task applications is shown in the following example:

```
<ejb-ref>
  <ejb-ref-name>ejb/HumanTaskManagerHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.ibm.task.api.HumanTaskManagerHome</home>
  <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
```

If you use WebSphere Integration Developer to add the EJB reference to the deployment descriptor, the binding for the EJB reference is automatically

created when the application is deployed. For more information on adding EJB references, refer to the WebSphere Integration Developer documentation.

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 or the TaskContainer application runs, complete the following actions:

- a. For process applications, package the `<install_root>/ProcessChoreographer/client/bpe137650.jar` file with the enterprise archive (EAR) file of your application.
- b. For task applications, package the `<install_root>/ProcessChoreographer/client/task137650.jar` file with the EAR file of your application.
- c. If you use complex data types in your business process or human task and your client does not run in an EJB application or a Web application, package the corresponding XSD or WSDL files with the EAR file of your application.
- d. Set the **Classpath** parameter in the manifest file of the application module to include the JAR file.

The application module can be a J2EE application, a Web application, or an EJB application.

3. Retrieve a reference to the home interface of the remote session bean from Java Naming and Directory Interface (JNDI).

The following example shows this step for a process application:

```
// Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

// Lookup the remote home interface of the BusinessFlowManager bean
Object result =
    initialContext.lookup("java:comp/env/ejb/BusinessFlowManagerHome");

// Convert the lookup result to the proper type
BusinessFlowManagerHome processHome =
    (BusinessFlowManagerHome)javax.rmi.PortableRemoteObject.narrow
    (result,BusinessFlowManagerHome.class);
```

The home interface of the 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 session bean.

The following example shows this step for a process application:

```
BusinessFlowManager process = processHome.create();
```

Access to the session bean does not guarantee that the caller can perform all of the actions provided by the bean; the caller must also be authorized for these actions. When an instance of the session bean is created, a context is associated with the instance of the session bean. The context contains the caller's principal ID, group membership list, and indicates whether the caller has one of the Business Process Choreographer J2EE roles. The context is used to check the caller's authorization for each call, even when global security is not set. If global security is not set, the caller's principal ID has the value UNAUTHENTICATED.

5. Call the business functions exposed by the service interface.

The following example shows this step for a process application:

```
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();
```

Tip: To prevent database deadlocks, avoid running statements similar to the following in parallel transactions:

```
// Obtain user transaction interface
UserTransaction transaction=
    (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//read lock on the activity instance
process.getActivityInstance(aiid);
//write lock on the activity instance
process.claim(aiid);

transaction.commit();
```

Example

Here is an example of how steps 3 through 5 might look for a task application.

```
//Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

//Lookup the remote home interface of the HumanTaskManager bean
Object result =
    initialContext.lookup("java:comp/env/ejb/HumanTaskManagerHome");

//Convert the lookup result to the proper type
HumanTaskManagerHome taskHome =
    (HumanTaskManagerHome)javax.rmi.PortableRemoteObject.narrow
    (result,HumanTaskManagerHome.class);

...
//Access the remote interface of the session bean.
HumanTaskManager task = taskHome.create();

...
//Call the business functions exposed by the service interface
task.callTask(tkiid,input);
```

Accessing the local session bean:

An EJB client application accesses the appropriate local session bean through the home interface of the bean.

The session bean can be either the LocalBusinessFlowManager session bean for process applications or the LocalHumanTaskManager session bean for human task applications.

1. Add a reference to the local session bean to the application deployment descriptor. Add the reference to one of the following files:
 - 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

The reference to the local home interface for process applications is shown in the following example:

```
<ejb-local-ref>
<ejb-ref-name>ejb/LocalBusinessFlowManagerHome</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
<local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
```

The reference to the local home interface for task applications is shown in the following example:

```
<ejb-local-ref>
<ejb-ref-name>ejb/LocalHumanTaskManagerHome</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
<local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>
```

If you use WebSphere Integration Developer to add the EJB reference to the deployment descriptor, the binding for the EJB reference is automatically created when the application is deployed. For more information on adding EJB references, refer to the WebSphere Integration Developer documentation.

2. Retrieve a reference to the local home interface of the local session bean from Java Naming and Directory Interface (JNDI).

The following example shows this step for a process application:

```
// Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

// Lookup the local home interface of the LocalBusinessFlowManager bean

LocalBusinessFlowManagerHome processHome =
    (LocalBusinessFlowManagerHome)initialContext.lookup
    ("java:comp/env/ejb/LocalBusinessFlowManagerHome");
```

The home interface of the local 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 local session bean.

The following example shows this step for a process application:

```
LocalBusinessFlowManager process = processHome.create();
```

Access to the session bean does not guarantee that the caller can perform all of the actions provided by the bean; the caller must also be authorized for these actions. When an instance of the session bean is created, a context is associated with the instance of the session bean. The context contains the caller's principal ID, group membership list, and indicates whether the caller has one of the Business Process Choreographer J2EE roles. The context is used to check the caller's authorization for each call, even when global security is not set. If global security is not set, the caller's principal ID has the value `UNAUTHENTICATED`.

4. Call the business functions exposed by the service interface.

The following example shows this step for a process application:

```
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();
```

Tip: To prevent database deadlocks, avoid running statements similar to the following in parallel transactions:

```
// Obtain user transaction interface
UserTransaction transaction=
    (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//read lock on the activity instance
process.getActivityInstance(aiid);
//write lock on the activity instance
process.claim(aiid);

transaction.commit();
```

Example

Here is an example of how steps 2 through 4 might look for a task application.

```
//Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

//Lookup the local home interface of the LocalHumanTaskManager bean
LocalHumanTaskManagerHome taskHome =
    (LocalHumanTaskManagerHome)initialContext.lookup
    ("java:comp/env/ejb/LocalHumanTaskManagerHome");

...
//Access the local interface of the local session bean
LocalHumanTaskManager task = taskHome.create();

...
//Call the business functions exposed by the service interface
task.callTask(tkid,input);
```

Querying business-process and task-related objects

The client applications work with business-process and task-related objects. You can query business-process and task-related objects in the database to retrieve specific properties of these objects.

During the configuration of Business Process Choreographer, a relational database is associated with both the business process container and the task container. The

database stores all of the template (model) and instance (runtime) data for managing business processes and tasks. You use SQL-like syntax to query this data.

You can perform a one-off query to retrieve a specific property of an object. You can also save queries that you use often and include these stored queries in your application.

Queries on business-process and task-related objects:

Use the query method or the queryAll method of the service API to retrieve stored information about business processes and tasks.

The query method returns objects according to the caller's authorization. The query result set contains the properties of only those objects for which the caller has an associated work item. The queryAll method returns the selected data for all of the objects in the database.

Predefined database views are provided for you to query the object properties.

The query is made up of the following elements:

- Select clause
- Where clause
- Order-by clause
- Skip-tuples parameter
- Threshold parameter
- Time-zone parameter

The syntax of the query depends on the object type. The following table shows the syntax for each of the different object types.

Table 2.

Object	Syntax
Process template	ProcessTemplateData[] queryProcessTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);
Task template	TaskTemplate[] queryTaskTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);
Business-process and task-related data	QueryResultSet query (java.lang.String selectClause, java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer skipTuples java.lang.Integer threshold, java.util.TimeZone timezone);

For example, a list of work items IDs that are accessible to the caller of the function is retrieved by:

```
QueryResultSet result = process.query("DISTINCT WORK_ITEM.WIID",
                                     null, null, null, null, null);
```

The query interface also contains a `queryAll` method. You can use this method to retrieve the data for all of the objects that are stored in the database, for example, for monitoring purposes. The caller of the `queryAll` method must have one of the following Java 2 Platform, Enterprise Edition (J2EE) roles: `BPESystemAdministrator`, `BPESystemMonitor`, `TaskSystemAdministrator`, or `TaskSystemMonitor`. Authorization checking using the corresponding work item of the object is not applied.

You can include both custom properties and variable properties in queries. If you include several custom properties or variable properties in your query, this results in self-joins on the corresponding database table. Depending on your database system, these `query()` calls might have performance implications.

You can also store queries in the Business Process Choreographer database using the `createStoredQuery` method. You provide the query criteria when you define the stored query. The criteria are applied dynamically when the stored query runs, that is, the data is assembled at runtime. If the stored query contains parameters, these are also resolved when the query runs.

For more information on the Business Process Choreographer APIs, see the Javadoc in the `com.ibm.bpe.api` package for process-related methods and in the `com.ibm.task.api` package for task-related methods.

Select clause:

The `select` clause in the query function identifies the object properties that are to be returned by a query.

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()`, or `MAX()`.

To select the properties of multiple name-value pairs, such as custom properties and properties of variables that can be queried, add a one-digit counter to the view name. This counter can take the values 1 through 9.

Examples of 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 of the IDs of objects, without duplicates, for which the caller has a work item.
- `"ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON"`
Gets the names of the activities the caller has work items for and their assignment reasons.
- `"ACTIVITY.STATE, PROCESS_INSTANCE.STARTER"`

Gets the states of the activities and the starters of their associated process instances.

- "DISTINCT TASK.TKIID, TASK.NAME"
Gets all of the IDs and names of tasks, without duplicates, for which the caller has a work item.
- "TASK_CPROP1.STRING_VALUE, TASK_CPROP2.STRING_VALUE"
Gets the values of the custom properties that are specified further in the where clause.
- "QUERY_PROPERTY1.STRING_VALUE, QUERY_PROPERTY2.INT_VALUE"
Gets the values of the properties of variables that can be queried. These parts are specified further in the where clause.
- "COUNT(DISTINCT TASK.TKIID)"
Counts the number of work items for unique tasks that satisfy the where clause.

Where clause:

The where clause in the query function describes the filter criteria to apply to the query domain.

The syntax of a where clause is the same as an SQL where clause. You do not need to explicitly add an SQL from clause or join predicates to the where clause, these constructs are added automatically when the query runs. If you do not want to apply filter criteria, 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 that are defined for the queried database.

The following rules also apply:

- Specify object ID constants as ID('string-rep-of-oid').
- Specify binary constants as BIN('UTF-8 string').
- 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.
- If the value of the property in the comparison statement contains single quotation marks ('), double the quotation marks, for example, "TASK_CPROP.STRING_VALUE='d''automatisation''".
- Refer to properties of multiple name-value pairs, such as custom properties, by adding a one-digit suffix to the view name. For example:
"TASK_CPROP1.NAME='prop1' AND "TASK_CPROP2.NAME='prop2' "
- 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').

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 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
"ACTIVITY.BUSINESS_RELEVANCE = TRUE"
- Using custom properties
"TASK_CPROP1.NAME = 'prop1' AND " TASK_CPROP1.STRING_VALUE = 'v1' AND
TASK_CPROP2.NAME = 'prop2' AND " TASK_CPROP2.STRING_VALUE = 'v2' "

Order-by clause:

The order-by clause in the query function specifies 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.

Sort criteria are applied to the server, that is, the locale of the server is used for sorting. 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.

Skip-tuples parameter:

The skip-tuples parameter specifies the number of query-result-set tuples from the beginning of the query result set that are to be ignored and not to be returned to the caller.

Use this parameter with the threshold parameter to implement paging in a client application, for example, to retrieve the first 20 items, then the next 20 items, and so on.

If this parameter is set to `null` and the threshold parameter is not set, all of the qualifying tuples are returned.

Example of a skip-tuples parameter

- `new Integer(5)`

Specifies that the first five qualifying tuples are not to be returned.

Threshold parameter:

The threshold parameter in the query function restricts the number of objects returned from the server to the client in the query result set.

Because query result sets in production scenarios can contain thousands or even millions of items, it is a best practice to always specify a threshold. The threshold parameter can be useful, for example, in a graphical user interface where only a small number of items should be displayed at one time. If you set the threshold parameter accordingly, the database query is faster and less data needs to transfer from the server to the client.

If this parameter is set to `null` and the skip-tuples parameter is not set, all of the qualifying objects are returned.

Example of a threshold parameter

- `new Integer(50)`

Specifies that 50 qualifying tuples are to be returned.

Timezone parameter:

The time-zone parameter in the query function defines the time zone for time-stamp constants in the query.

Time zones can differ between the client that starts the query and the server that processes the query. Use the time-zone parameter to specify the time zone 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 time zone that is 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 time-zone parameters

- ```
process.query("ACTIVITY.AIID",
 "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
 null,
 null,
 java.util.TimeZone.getDefault());
```

Returns object IDs for activities that started later than 17:40 local time on 1 January 2005.

- `process.query("ACTIVITY.AIID",  
"ACTIVITY.STARTED > TS('2005-01-01T17:40')",  
null, null, null);`

Return object IDs for activities that started later than 17:40 UTC on 1 January 2005. This specification is, for example, 6 hours earlier in Eastern Standard Time.

*Parameters in stored queries:*

A stored query is a query that is stored in the database and identified by a name. The qualifying tuples are assembled dynamically when the query is run. To make stored queries reusable, you can use parameters in the query definition that are resolved at runtime.

For example, you have defined custom properties to store customer names. You can define queries to return the tasks that are associated with a particular customer, ACME Co. To query this information, the where clause in your query might look similar to the following example:

```
String whereClause =
"TASK.STATE = TASK.STATE.STATE_READY
AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = 'ACME Co.'";
```

To make this query reusable so that you can also search for the customer, BCME Ltd, you can use parameters for the values of the custom property. If you add parameters to the task query, it might look similar to the following example:

```
String whereClause =
"TASK.STATE = TASK.STATE.STATE_READY
AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = '@param1'";
```

The @param1 parameter is resolved at runtime from the list of parameters that is passed to the query method. The following rules apply to the use of parameters in queries:

- Parameters can only be used in the where clause.
- Parameters are strings.
- Parameters are replaced at runtime using string replacement. If you need special characters you must specify these in the where clause or passed-in at runtime as part of the parameter.
- Parameter names consist of the string @param concatenated with an integer number. The lowest number is 1, which points to the first item in the list of parameters that is passed to the query API at runtime.
- A parameter can be used multiple times within a where clause; all occurrences of the parameter are replaced by the same value.

*Query results:*

A query result set contains the results of a query.

The elements of the result 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         | <code>getString</code>                                                                                                        |
| OID            | <code>getOID</code>                                                                                                           |
| Timestamp      | <code>getTimestamp</code><br><code>getString</code>                                                                           |
| Integer        | <code>getInteger</code><br><code>getShort</code><br><code>getLong</code><br><code>getString</code><br><code>getBoolean</code> |
| Boolean        | <code>getBoolean</code><br><code>getShort</code><br><code>getInteger</code><br><code>getLong</code><br><code>getString</code> |
| byte[]         | <code>getBinary</code>                                                                                                        |

### 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 method 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"

```

*Predefined views for queries on business-process and human-task objects:*

Predefined database views are provided for business-process and human-task objects. Use these views when you query reference data for these objects.

When you use the predefined views, you do not need to explicitly add join predicates for view columns, these constructs are added automatically for you. You can use the generic query function of the service API (BusinessFlowManagerService or HumanTaskManagerService) to query this data. You can also use the corresponding method of the HumanTaskManagerDelegate API or your predefined queries provided by your implementations of the ExecutableQuery interface.

*ACTIVITY view:*

Use this predefined database view for queries on activities.

*Table 3. Columns in the ACTIVITY view*

| 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 are:<br><br>KIND_INVOKE (21)<br>KIND_RECEIVE (23)<br>KIND_REPLY (24)<br>KIND_THROW (25)<br>KIND_RETHROW (46)<br>KIND_TERMINATE (26)<br>KIND_WAIT (27)<br>KIND_COMPENSATE (29)<br>KIND_SEQUENCE (30)<br>KIND_EMPTY (3)<br>KIND_SWITCH (32)<br>KIND_WHILE (34)<br>KIND_PICK (36)<br>KIND_FLOW (38)<br>KIND_SCOPE (40)<br>KIND_SCRIPT (42)<br>KIND_STAFF (43)<br>KIND_ASSIGN (44)<br>KIND_CUSTOM (45)<br>KIND_FOR_EACH_PARALLEL (49)<br>KIND_FOR_EACH_SERIAL (47) |
| COMPLETED       | Timestamp | The time the activity is completed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| ACTIVATED       | Timestamp | The time the activity is activated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| FIRST_ACTIVATED | Timestamp | The time at which the activity was activated for the first time.                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Table 3. Columns in the ACTIVITY view (continued)

| Column name        | Type      | Comments                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STARTED            | Timestamp | The time the activity is started.                                                                                                                                                                                                                                                                                                                                            |
| STATE              | Integer   | The state of the activity. Possible values are:<br><br>STATE_INACTIVE (1)<br>STATE_READY (2)<br>STATE_RUNNING (3)<br>STATE_PROCESSING_UNDO (14)<br>STATE_SKIPPED (4)<br>STATE_FINISHED (5)<br>STATE_FAILED (6)<br>STATE_TERMINATED (7)<br>STATE_CLAIMED (8)<br>STATE_TERMINATING (9)<br>STATE_FAILING (10)<br>STATE_WAITING (11)<br>STATE_EXPIRED (12)<br>STATE_STOPPED (13) |
| OWNER              | String    | Principal ID of the owner.                                                                                                                                                                                                                                                                                                                                                   |
| DESCRIPTION        | String    | If the activity template description contains placeholders, this column contains the description of the activity instance with the placeholders resolved.                                                                                                                                                                                                                    |
| TEMPLATE_NAME      | String    | Name of the associated activity template.                                                                                                                                                                                                                                                                                                                                    |
| TEMPLATE_DESCR     | String    | Description of the associated activity template.                                                                                                                                                                                                                                                                                                                             |
| BUSINESS_RELEVANCE | Boolean   | Specifies whether the activity is business relevant. Possible values are:<br><br><b>TRUE</b> The activity is business relevant. You can view the activity status in Business Process Choreographer Explorer.<br><br><b>FALSE</b> The activity is not business relevant.                                                                                                      |
| EXPIRES            | Timestamp | The date and time when the activity is due to expire. If the activity has expired, the date and time when this event occurred.                                                                                                                                                                                                                                               |

*ACTIVITY\_ATTRIBUTE* view:

Use this predefined database view for queries on custom properties for activities.

Table 4. Columns in the ACTIVITY\_ATTRIBUTE view

| Column name | Type   | Comments                                                    |
|-------------|--------|-------------------------------------------------------------|
| AIID        | ID     | The ID of the activity instance that has a custom property. |
| NAME        | String | The name of the custom property.                            |

Table 4. Columns in the ACTIVITY\_ATTRIBUTE view (continued)

| Column name | Type   | Comments                          |
|-------------|--------|-----------------------------------|
| VALUE       | String | The value of the custom property. |

ACTIVITY\_SERVICE view:

Use this predefined database view for queries on activity services.

Table 5. Columns in the ACTIVITY\_SERVICE view

| Column name    | Type   | Comments                                                 |
|----------------|--------|----------------------------------------------------------|
| EIID           | ID     | The ID of the event instance.                            |
| AIID           | ID     | The ID of the activity waiting for the event.            |
| 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 namespace.                                |
| OPERATION      | String | The operation name of the service.                       |

APPLICATION\_COMP view:

Use this predefined database view to query the application component ID and default settings for tasks.

Table 6. Columns in the APPLICATION\_COMP view

| Column name        | Type    | Comments                                                                                                                                                                                                                                                                                                                                                        |
|--------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACOID              | String  | The ID of the application component.                                                                                                                                                                                                                                                                                                                            |
| BUSINESS_RELEVANCE | Boolean | The default task business-relevance policy of the component. This value can be overwritten by a definition in the task template or the task. The attribute affects logging to the audit trail.<br>Possible values are:<br><b>TRUE</b> The task is business relevant and it is audited.<br><b>FALSE</b> The task is not business relevant and it is not audited. |
| NAME               | String  | Name of the application component.                                                                                                                                                                                                                                                                                                                              |
| SUPPORT_AUTOCLAIM  | Boolean | The default automatic-claim policy of the component. If this attribute is set to TRUE, the task can be automatically claimed if a single user is the potential owner. This value can be overwritten by a definition in the task template or task.                                                                                                               |

Table 6. Columns in the APPLICATION\_COMP view (continued)

| Column name        | Type    | Comments                                                                                                                                                                                                                                       |
|--------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SUPPORT_CLAIM_SUSP | Boolean | The default setting of the component that determines whether suspended tasks can be claimed. If this attribute is set to TRUE, suspended tasks can be claimed. This value can be overwritten by a definition in the task template or the task. |
| SUPPORT_DELEGATION | Boolean | The default task delegation policy of the component. If this attribute is set to TRUE, the work item assignments for the task can be modified. This means that work items can be created, deleted, or transferred.                             |
| SUPPORT_FOLLOW_ON  | Boolean | The default follow-on task policy of the component. If this attribute is set to TRUE, follow-on tasks can be created for tasks. This value can be overwritten by a definition in the task template or the task.                                |
| SUPPORT_SUB_TASK   | Boolean | The default subtask policy of the component. If this attribute is set to TRUE, subtasks can be created for tasks. This value can be overwritten by a definition in the task template or the task.                                              |

ESCALATION view:

Use this predefined database view to query data for escalations.

Table 7. Columns in the ESCALATION view

| Column name | Type    | Comments                                                                                                                                                                                                                                                                                                                 |
|-------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ESIID       | String  | The ID of the escalation instance.                                                                                                                                                                                                                                                                                       |
| ACTION      | Integer | The action triggered by the escalation. Possible values are:<br><br><b>ACTION_CREATE_WORK_ITEM (1)</b><br>Creates a work item for each escalation receiver.<br><br><b>ACTION_SEND_EMAIL (2)</b><br>Sends an e-mail to each escalation receiver.<br><br><b>ACTION_CREATE_EVENT (3)</b><br>Creates and publishes an event. |



Table 7. Columns in the ESCALATION view (continued)

| Column name        | Type      | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACTIVATION_STATE   | Integer   | An escalation instance is created if the corresponding task reaches one of the following states:<br><br><b>ACTIVATION_STATE_READY (2)</b><br>Specifies that the human or participating task is ready to be claimed.<br><br><b>ACTIVATION_STATE_RUNNING (3)</b><br>Specifies that the originating task is started and running.<br><br><b>ACTIVATION_STATE_CLAIMED (8)</b><br>Specifies that the task is claimed.<br><br><b>ACTIVATION_STATE_WAITING_FOR_SUBTASK (20)</b><br>Specifies that the task is waiting for the completion of subtasks. |
| ACTIVATION_TIME    | Timestamp | The time when the escalation is activated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| AT_LEAST_EXP_STATE | Integer   | The state of the task that is expected by the escalation. If a timeout occurs, the task state is compared with the value of this attribute. Possible values are:<br><br><b>AT_LEAST_EXPECTED_STATE_CLAIMED (8)</b><br>Specifies that the task is claimed.<br><br><b>AT_LEAST_EXPECTED_STATE_ENDED (20)</b><br>Specifies that the task is in a final state (FINISHED, FAILED, TERMINATED or EXPIRED).<br><br><b>AT_LEAST_EXPECTED_STATE_SUBTASKS_COMPLETED (21)</b><br>Specifies that all of the subtasks of the task are complete.            |
| ESTID              | String    | The ID of the corresponding escalation template.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| FIRST_ESIID        | String    | The ID of the first escalation in the chain.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| INCREASE_PRIORITY  | Integer   | Indicates how the task priority will be increased. Possible values are:<br><br><b>INCREASE_PRIORITY_NO (1)</b><br>The task priority is not increased.<br><br><b>INCREASE_PRIORITY_ONCE (2)</b><br>The task priority is increased once by one.<br><br><b>INCREASE_PRIORITY_REPEATED (3)</b><br>The task priority is increased by one each time the escalation repeats.                                                                                                                                                                         |
| NAME               | String    | The name of the escalation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| STATE              | Integer   | The state of the escalation. Possible values are:<br><br>STATE_INACTIVE (1)<br>STATE_WAITING (2)<br>STATE_ESCALATED (3)<br>STATE_SUPERFLUOUS (4)                                                                                                                                                                                                                                                                                                                                                                                              |

Table 7. Columns in the ESCALATION view (continued)

| Column name | Type   | Comments                                              |
|-------------|--------|-------------------------------------------------------|
| TKIID       | String | The task instance ID to which the escalation belongs. |

ESCALATION\_CPROP view:

Use this predefined database view to query custom properties for escalations.

Table 8. Columns in the ESCALATION\_CPROP view

| Column name  | Type   | Comments                                                |
|--------------|--------|---------------------------------------------------------|
| ESIID        | String | The escalation ID.                                      |
| NAME         | String | The name of the property.                               |
| DATA_TYPE    | String | The type of the class for non-string custom properties. |
| STRING_VALUE | String | The value for custom properties of type String.         |

ESCALATION\_DESC view:

Use this predefined database view to query multilingual descriptive data for escalations.

Table 9. Columns in the ESCALATION\_DESC view

| Column name  | Type   | Comments                                                                |
|--------------|--------|-------------------------------------------------------------------------|
| ESIID        | String | The escalation ID.                                                      |
| LOCALE       | String | The name of the locale associated with the description or display name. |
| DESCRIPTION  | String | A description of the task template.                                     |
| DISPLAY_NAME | String | The descriptive name of the escalation.                                 |

PROCESS\_ATTRIBUTE view:

Use this predefined database view for queries on custom properties for processes.

Table 10. Columns in the PROCESS\_ATTRIBUTE view

| Column name | Type   | Comments                                                   |
|-------------|--------|------------------------------------------------------------|
| PIID        | ID     | The ID of the process instance that has a custom property. |
| NAME        | String | The name of the custom property.                           |
| VALUE       | String | The value of the custom property.                          |

PROCESS\_INSTANCE view:

Use this predefined database view for queries on process instances.

Table 11. Columns in the PROCESS\_INSTANCE view

| Column name | Type | Comments                 |
|-------------|------|--------------------------|
| PTID        | ID   | The process template ID. |

Table 11. Columns in the *PROCESS\_INSTANCE* view (continued)

| Column name    | Type      | Comments                                                                                                                                                                                                                                                                                                                                               |
|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIID           | ID        | The process instance ID.                                                                                                                                                                                                                                                                                                                               |
| NAME           | String    | The name of the process instance.                                                                                                                                                                                                                                                                                                                      |
| STATE          | Integer   | The state of the process instance. Possible values are:<br><br>STATE_READY (1)<br>STATE_RUNNING (2)<br>STATE_FINISHED (3)<br>STATE_COMPENSATING (4)<br>STATE_INDOUBT (10)<br>STATE_FAILED (5)<br>STATE_TERMINATED (6)<br>STATE_COMPENSATED (7)<br>STATE_COMPENSATION_FAILED (12)<br>STATE_TERMINATING (8)<br>STATE_FAILING (9)<br>STATE_SUSPENDED (11) |
| CREATED        | Timestamp | The time the process instance is created.                                                                                                                                                                                                                                                                                                              |
| STARTED        | Timestamp | The time the process instance started.                                                                                                                                                                                                                                                                                                                 |
| COMPLETED      | Timestamp | The time the process instance completed.                                                                                                                                                                                                                                                                                                               |
| PARENT_NAME    | String    | The name of the parent process instance.                                                                                                                                                                                                                                                                                                               |
| TOP_LEVEL_NAME | String    | The name of the top-level process instance. If there is no top-level process instance, this is the name of the current process instance.                                                                                                                                                                                                               |
| STARTER        | String    | The principal ID of the starter of the process instance.                                                                                                                                                                                                                                                                                               |
| DESCRIPTION    | String    | If the description of the process template contains placeholders, this column contains the description of the process instance with the placeholders resolved.                                                                                                                                                                                         |
| TEMPLATE_NAME  | String    | The name of the associated process template.                                                                                                                                                                                                                                                                                                           |
| TEMPLATE_DESCR | String    | Description of the associated process template.                                                                                                                                                                                                                                                                                                        |

*PROCESS\_TEMPLATE* view:

Use this predefined database view for queries on process templates.

Table 12. Columns in the *PROCESS\_TEMPLATE* view

| Column name      | Type      | Comments                                                                      |
|------------------|-----------|-------------------------------------------------------------------------------|
| PTID             | ID        | The process template ID.                                                      |
| NAME             | String    | The name of the process template.                                             |
| VALID_FROM       | Timestamp | The time from when the process template can be instantiated.                  |
| TARGET_NAMESPACE | String    | The target namespace of the process template.                                 |
| APPLICATION_NAME | String    | The name of the enterprise application to which the process template belongs. |
| VERSION          | String    | User-defined version.                                                         |

Table 12. Columns in the *PROCESS\_TEMPLATE* view (continued)

| Column name    | Type      | Comments                                                                                                                                                                                                                                                                                                                               |
|----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 are:<br><br>STATE_STARTED (1)<br>STATE_STOPPED (2)                                                                                                                                                                                    |
| EXECUTION_MODE | Integer   | Specifies how process instances that are derived from this process template can be run. Possible values are:<br><br>EXECUTION_MODE_MICROFLOW (1)<br>EXECUTION_MODE_LONG_RUNNING (2)                                                                                                                                                    |
| DESCRIPTION    | String    | Description of the process template.                                                                                                                                                                                                                                                                                                   |
| COMP_SPHERE    | Integer   | Specifies the compensation behavior of instances of microflows in the process template; either an existing compensation sphere is joined or a compensation sphere is created.<br><br>Possible values are:<br><br>COMP_SPHERE_REQUIRED (2)<br>COMP_SPHERE_REQUIRES_NEW (3)<br>COMP_SPHERE_SUPPORTS (4)<br>COMP_SPHERE_NOT_SUPPORTED (1) |

*QUERY\_PROPERTY* view:

Use this predefined database view for queries on process-level variables.

Table 13. Columns in the *QUERY\_PROPERTY* view

| Column name   | Type    | Comments                                                                                                                                               |
|---------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIID          | ID      | The process instance ID.                                                                                                                               |
| VARIABLE_NAME | String  | The name of the process-level variable.                                                                                                                |
| NAME          | String  | The name of the query property.                                                                                                                        |
| NAMESPACE     | String  | The namespace of the query property.                                                                                                                   |
| GENERIC_VALUE | String  | A string representation for property types that do not map to one of the defined types: STRING_VALUE, NUMBER_VALUE, DECIMAL_VALUE, or TIMESTAMP_VALUE. |
| STRING_VALUE  | String  | If a property type is mapped to a string type, this is the value of the string.                                                                        |
| NUMBER_VALUE  | Integer | If a property type is mapped to an integer type, this is the value of the integer.                                                                     |

Table 13. Columns in the QUERY\_PROPERTY view (continued)

| Column name     | Type      | Comments                                                                                 |
|-----------------|-----------|------------------------------------------------------------------------------------------|
| DECIMAL_VALUE   | Decimal   | If a property type is mapped to a floating point type, this is the value of the decimal. |
| TIMESTAMP_VALUE | Timestamp | If a property type is mapped to a timestamp type, this is the value of the timestamp.    |

TASK view:

Use this predefined database view for queries on task objects.

Table 14. Columns in the TASK view

| Column name        | Type      | Comments                                                                                                                                                                                                                                                                                                                   |
|--------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TKIID              | ID        | The ID of the task instance.                                                                                                                                                                                                                                                                                               |
| ACTIVATED          | Timestamp | The time when the task was activated.                                                                                                                                                                                                                                                                                      |
| APPLIC_DEFAULTS_ID | ID        | The ID of the application component that specifies the defaults for the task.                                                                                                                                                                                                                                              |
| APPLIC_NAME        | String    | The name of the enterprise application to which the task belongs.                                                                                                                                                                                                                                                          |
| BUSINESS_RELEVANCE | Boolean   | Specifies whether the task is business relevant. The attribute affects logging to the audit trail. Possible values are:<br><br><b>TRUE</b> The task is business relevant and it is audited.<br><br><b>FALSE</b> The task is not business relevant and it is not audited.                                                   |
| COMPLETED          | Timestamp | The time when the task completed.                                                                                                                                                                                                                                                                                          |
| CONTAINMENT_CTX_ID | ID        | The containment context for this task. This attribute determines the life cycle of the task. When the containment context of a task is deleted, the task is also deleted.                                                                                                                                                  |
| CTX_AUTHORIZATION  | Integer   | Allows the task owner to access the task context. Possible values are:<br><br><b>AUTH_NONE</b><br>No authorization rights for the associated context object.<br><br><b>AUTH_READER</b><br>Operations on the associated context object require reader authority, for example, reading the properties of a process instance. |
| DUE                | Timestamp | The time when the task is due.                                                                                                                                                                                                                                                                                             |
| EXPIRES            | Timestamp | The date when the task expires.                                                                                                                                                                                                                                                                                            |
| FIRST_ACTIVATED    | Timestamp | The time when the task was activated for the first time.                                                                                                                                                                                                                                                                   |
| FOLLOW_ON_TKIID    | ID        | The ID of the instance of the follow-on task.                                                                                                                                                                                                                                                                              |

Table 14. Columns in the TASK view (continued)

| Column name        | Type      | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HIERARCHY_POSITION | Integer   | Possible values are:<br><b>HIERARCHY_POSITION_TOP_TASK (0)</b><br>The top-level task in the task hierarchy.<br><b>HIERARCHY_POSITION_SUB_TASK (1)</b><br>The task is a subtask in the task hierarchy.<br><b>HIERARCHY_POSITION_FOLLOW_ON_TASK (2)</b><br>The task is a follow-on task in the task hierarchy.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| IS_AD_HOC          | Boolean   | Indicates whether this task was created dynamically at runtime or from a task template.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| IS_ESCALATED       | Boolean   | Indicates whether an escalation of this task has occurred.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| IS_INLINE          | Boolean   | Indicates whether the task is an inline task in a business process.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| IS_WAIT_FOR_SUB_TK | Boolean   | Indicates whether the parent task is waiting for a subtask to reach an end state.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| KIND               | Integer   | The kind of task. Possible values are:<br><b>KIND_HUMAN (101)</b><br>States that the task is created and processed by a human.<br><b>KIND_WPC_STAFF_ACTIVITY (102)</b><br>States that the task is a human task that is a staff activity of a WebSphere Business Integration Server Foundation, version 5 business process.<br><b>KIND_ORIGINATING (103)</b><br>States that the task supports person-to-computer interactions, which enables people to create, initiate, and start services.<br><b>KIND_PARTICIPATING (105)</b><br>States that the task supports computer-to-person interactions, which enable a person to implement a service.<br><b>KIND_ADMINISTRATIVE (106)</b><br>States that the task is an administrative task. |
| LAST_MODIFIED      | Timestamp | The time when the task was last modified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| LAST_STATE_CHANGE  | Timestamp | The time when the state of the task was last modified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| NAME               | String    | The name of the task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| NAME_SPACE         | String    | The namespace that is used to categorize the task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| ORIGINATOR         | String    | The principal ID of the task originator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| OWNER              | String    | The principal ID of the task owner.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

Table 14. Columns in the TASK view (continued)

| Column name        | Type      | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PARENT_CONTEXT_ID  | String    | The parent context for this task. This attribute provides a key to the corresponding context in the calling application component. The parent context is set by the application component that creates the task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| PRIORITY           | Integer   | The priority of the task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| STARTED            | Timestamp | The time when the task was started (STATE_RUNNING, STATE_CLAIMED).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| STARTER            | String    | The principal ID of the task starter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| STATE              | Integer   | The state of the task. Possible values are:<br><b>STATE_READY (2)</b><br>States that the task is ready to be claimed.<br><b>STATE_RUNNING (3)</b><br>States that the task is started and running.<br><b>STATE_FINISHED (5)</b><br>States that the task finished successfully.<br><b>STATE_FAILED (6)</b><br>States that the task did not finish successfully.<br><b>STATE_TERMINATED (7)</b><br>States that the task has been terminated because of an external or internal request.<br><b>STATE_CLAIMED (8)</b><br>States that the task is claimed.<br><b>STATE_EXPIRED (12)</b><br>States that the task ended because it exceeded its specified duration.<br><b>STATE_FORWARDED (101)</b><br>States that task completed with a follow-on task. |
| SUPPORT_AUTOCLAIM  | Boolean   | Indicates whether this task is claimed automatically if it is assigned to a single user.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| SUPPORT_CLAIM_SUSP | Boolean   | Indicates whether this task can be claimed if it is suspended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| SUPPORT_DELEGATION | Boolean   | Indicates whether this task supports work delegation through creating, deleting, or transferring work items.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| SUPPORT_FOLLOW_ON  | Boolean   | Indicates whether this task supports the creation of follow-on tasks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| SUPPORT_SUB_TASK   | Boolean   | Indicates whether this task supports the creation of subtasks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| SUSPENDED          | Boolean   | Indicates whether the task is suspended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| TKTID              | ID        | The task template ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| TOP_TKIID          | ID        | The top parent task instance ID if this is a subtask.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| TYPE               | String    | The type used to categorize the task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

TASK\_CPROP view:

Use this predefined database view to query custom properties for task objects.

Table 15. Columns in the TASK\_CPROP view

| Column name  | Type   | Comments                                        |
|--------------|--------|-------------------------------------------------|
| TKIID        | String | The task instance ID.                           |
| NAME         | String | The name of the property.                       |
| STRING_VALUE | String | The value for custom properties of type String. |

TASK\_DESC view:

Use this predefined database view to query multilingual descriptive data for task objects.

Table 16. Column in the TASK\_DESC view

| Column name  | Type   | Comments                                                                |
|--------------|--------|-------------------------------------------------------------------------|
| TKIID        | String | The task instance ID.                                                   |
| LOCALE       | String | The name of the locale associated with the description or display name. |
| DESCRIPTION  | String | A description of the task.                                              |
| DISPLAY_NAME | String | The descriptive name of the task.                                       |

TASK\_TEMPL view:

This predefined database view holds data that you can use to instantiate tasks.

Table 17. Columns in the TASK\_TEMPL view

| Column name        | Type      | Comments                                                                                                                                                                                                                                                                          |
|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TKTID              | String    | The task template ID.                                                                                                                                                                                                                                                             |
| VALID_FROM         | Timestamp | The time when the task template becomes available for instantiation.                                                                                                                                                                                                              |
| APPLIC_DEFAULTS_ID | String    | The ID of the application component that specifies the defaults for the task template.                                                                                                                                                                                            |
| APPLIC_NAME        | String    | The name of the enterprise application to which the task template belongs.                                                                                                                                                                                                        |
| BUSINESS_RELEVANCE | Boolean   | Specifies whether the task template is business relevant. The attribute affects logging to the audit trail. Possible values are:<br><br><b>TRUE</b> The task is business relevant and it is audited.<br><br><b>FALSE</b> The task is not business relevant and it is not audited. |
| CONTAINMENT_CTX_ID | ID        | The containment context for this task template. This attribute determines the life cycle of the task template. When a containment context is deleted, the task template is also deleted.                                                                                          |



Table 17. Columns in the TASK\_TEMPL view (continued)

| Column name        | Type    | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CTX_AUTHORIZATION  | Integer | Allows the task owner to access the task context. Possible values are:<br><br><b>AUTH_NONE</b><br>No authorization rights for the associated context object.<br><br><b>AUTH_READER</b><br>Operations on the associated context object require reader authority, for example, reading the properties of a process instance.                                                                                                                                                                                                                                             |
| IS_AD_HOC          | Boolean | Indicates whether this task template was created dynamically at runtime or when the task was deployed as part of an EAR file.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| IS_INLINE          | Boolean | Indicates whether this task template is modeled as a task within a business process.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| KIND               | Integer | The kind of tasks that are derived from this task template. Possible values are:<br><br><b>KIND_HUMAN (101)</b><br>Specifies that the task is created and processed by a human.<br><br><b>KIND_ORIGINATING (103)</b><br>Specifies that a human can assign a task to a computer. In this case, a human invokes an automated service.<br><br><b>KIND_PARTICIPATING (105)</b><br>Specifies that a service component (such as a business process) assigns a task to a human.<br><br><b>KIND_ADMINISTRATIVE (106)</b><br>Specifies that the task is an administrative task. |
| NAME               | String  | The name of the task template.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| NAMESPACE          | String  | The namespace that is used to categorize the task template.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| PRIORITY           | Integer | The priority of the task template.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| STATE              | Integer | The state of the task template. Possible values are:<br><br><b>STATE_STARTED (1)</b><br>Specifies that the task template is available for creating task instances.<br><br><b>STATE_STOPPED (2)</b><br>Specifies that the task template is stopped. Task instances cannot be created from the task template in this state.                                                                                                                                                                                                                                              |
| SUPPORT_AUTOCLAIM  | Boolean | Indicates whether tasks derived from this task template can be claimed automatically if they are assigned to a single user.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| SUPPORT_CLAIM_SUSP | Boolean | Indicates whether tasks derived from this task template can be claimed if they are suspended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

Table 17. Columns in the TASK\_TEMPL view (continued)

| Column name        | Type    | Comments                                                                                                                             |
|--------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------|
| SUPPORT_DELEGATION | Boolean | Indicates whether tasks derived from this task template support work delegation using creation, deletion, or transfer of work items. |
| SUPPORT_FOLLOW_ON  | Boolean | Indicates whether the task template supports the creation of follow-on tasks.                                                        |
| SUPPORT_SUB_TASK   | Boolean | Indicates whether the task template supports the creation of subtasks.                                                               |
| TYPE               | String  | The type used to categorize the task template.                                                                                       |

TASK\_TEMPL\_CPROP view:

Use this predefined database view to query custom properties for task templates.

Table 18. Columns in the TASK\_TEMPL\_CPROP view

| Column name  | Type   | Comments                                                |
|--------------|--------|---------------------------------------------------------|
| TKTID        | String | The task template ID.                                   |
| NAME         | String | The name of the property.                               |
| DATA_TYPE    | String | The type of the class for non-string custom properties. |
| STRING_VALUE | String | The value for custom properties of type String.         |

TASK\_TEMPL\_DESC view:

Use this predefined database view to query multilingual descriptive data for task template objects.

Table 19. Columns in the TASK\_TEMPL\_DESC view

| Column name  | Type   | Comments                                                                |
|--------------|--------|-------------------------------------------------------------------------|
| TKTID        | String | The task template ID.                                                   |
| LOCALE       | String | The name of the locale associated with the description or display name. |
| DESCRIPTION  | String | A description of the task template.                                     |
| DISPLAY_NAME | String | The descriptive name of the task template.                              |

WORK\_ITEM view:

Use this predefined database view for queries on work items and authorization data for process, tasks, and escalations.

Table 20. Columns in the WORK\_ITEM view

| Column name | Type    | Comments                                         |
|-------------|---------|--------------------------------------------------|
| WIID        | ID      | The work item ID.                                |
| OWNER_ID    | String  | The principal ID of the owner.                   |
| GROUP_NAME  | String  | The name of the associated group worklist.       |
| EVERYBODY   | Boolean | Specifies whether everybody owns this work item. |

Table 20. Columns in the WORK\_ITEM view (continued)

| Column name       | Type      | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OBJECT_TYPE       | Integer   | <p>The type of the associated object. Possible values are:</p> <p><b>OBJECT_TYPE_ACTIVITY (1)</b><br/>Specifies that the work item was created for an activity.</p> <p><b>OBJECT_TYPE_PROCESS_INSTANCE (3)</b><br/>Specifies that the work item was created for a process instance.</p> <p><b>OBJECT_TYPE_TASK_INSTANCE (5)</b><br/>Specifies that the work item was created for a task.</p> <p><b>OBJECT_TYPE_TASK_TEMPLATE (6)</b><br/>Specifies that the work item was created for a task template.</p> <p><b>OBJECT_TYPE_ESCALATION_INSTANCE (7)</b><br/>Specifies that the work item was created for an escalation instance.</p> <p><b>OBJECT_TYPE_APPLICATION_COMPONENT (9)</b><br/>Specifies that the work item was created for an application component.</p> |
| OBJECT_ID         | ID        | The ID of the associated object, for example, the associated process or task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ASSOC_OBJECT_TYPE | Integer   | The type of the object referenced by the ASSOC_OID attribute, for example, task, process, or external objects. Use the values for the OBJECT_TYPE attribute.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| ASSOC_OID         | ID        | The ID of the object associated object with 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   | <p>The reason for the assignment of the work item. Possible values are:</p> <p>REASON_POTENTIAL_STARTER (5)<br/>REASON_POTENTIAL_INSTANCE_CREATOR (11)<br/>REASON_POTENTIAL_STARTER (1)<br/>REASON_EDITOR (2)<br/>REASON_READER (3)<br/>REASON_ORIGINATOR (9)<br/>REASON_OWNER (4)<br/>REASON_STARTER (6)<br/>REASON_ESCALATION_RECEIVER (10)<br/>REASON_ADMINISTRATOR (7)</p>                                                                                                                                                                                                                                                                                                                                                                                       |
| CREATION_TIME     | Timestamp | The date and time when the work item was created.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## Filtering data using variables in queries:

A query result returns the objects that match the query criteria. You might want to filter these results on the values of variables.

You can define variables that are used by a process at runtime in its process model. For these variables, you declare which parts can be queried.

For example, John Smith, calls his insurance company's service number to find out the progress of his insurance claim for his damaged car. The claims administrator uses the customer ID to find the claim.

1. **Optional:** List the properties of the variables in a process that can be queried.

Use the process template ID to identify the process. You can skip this step if you know which variables can be queried.

```
List variableProperties = process.getQueryProperties(ptid);
for (int i = 0; i < variableProperties.size(); i++)
{
 QueryProperty queryData = (QueryProperty)variableProperties.get(i);
 String variableName = queryData.getVariableName();
 String name = queryData.getName();
 int mappedType = queryData.getMappedType();
 ...
}
```

2. List the process instances with variables that match the filter criteria.

For this process, the customer ID is modeled as part of the variable `customerClaim` that can be queried. You can therefore use the customer's ID to find the claim.

```
QueryResultSet result = process.query
("PROCESS_INSTANCE.NAME, QUERY_PROPERTY.STRING_VALUE",
 "QUERY_PROPERTY.VARIABLE_NAME = 'customerClaim' AND " +
 "QUERY_PROPERTY.NAME = 'customerID' AND " +
 "QUERY_PROPERTY.STRING_VALUE like 'Smith%'",
 null, null, null, null);
```

This action returns a query result set that contains the process instance names and the values of the customer IDs for customers whose IDs start with Smith.

## Managing stored queries:

Stored queries provide a way to save queries that are run often. The stored query can be either a query that is available to all users (public query), or a query that belongs to a specific user (private query).

A stored query is a query that is stored in the database and identified by a name. A private and a public stored query can have the same name; private stored queries from different owners can also have the same name.

You can have stored queries for business process objects, task objects, or a combination of these two object types.

### *Managing public stored queries:*

Public stored queries are created by the system administrator. These queries are available to all users.

As the system administrator, you can create, view, and delete public stored queries. If you do not specify a user ID in the API call, it is assumed that the stored query is a public stored query.

1. Create a public stored query.

For example, the following code snippet creates a stored query for process instances and saves it with the name `CustomerOrdersStartingWithA`.

```
process.createStoredQuery("CustomerOrdersStartingWithA",
 "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
 "PROCESS_INSTANCE.NAME LIKE 'A%'",
 "PROCESS_INSTANCE.NAME",
 null, null);
```

The result of the stored query is a sorted list of all the process-instance names that begin with the letter A and their associated process instance IDs (PIID).

2. Run the query defined by the stored query.

```
QueryResultSet result = process.query("CustomerOrdersStartingWithA",
 new Integer(0));
```

This action returns the objects that fulfill the criteria. In this case, all of the customer orders that begin with A.

3. List the names of the available public stored queries.

The following code snippet shows how to limit the list of returned queries to just the public queries.

```
String[] storedQuery = process.getStoredQueryNames(StoredQueryData.KIND_PUBLIC);
```

4. **Optional:** Check the query that is defined by a specific stored query.

A stored private query can have the same name as a stored public query. If these names are the same, the private stored query is returned. The following code snippet shows how to return only the public query with the specified name. If you want to run this query for task-based objects, specify `StoredQuery` as the returned object type instead of `StoredQueryData`.

```
StoredQueryData storedQuery = process.getStoredQuery
 (StoredQueryData.KIND_PUBLIC, "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

5. Delete a public stored query.

The following code snippet shows how to delete the stored query that you created in step 1.

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

*Managing private stored queries for other users:*

Private queries can be created by any user. These queries are available only to the owner of the query and the system administrator.

As the system administrator, you can manage private stored queries that belong to a specific user.

1. Create a private stored query for the user ID Smith.

For example, the following code snippet creates a stored query for process instances and saves it with the name `CustomerOrdersStartingWithA` for the user ID Smith.

```
process.createStoredQuery("Smith", "CustomerOrdersStartingWithA",
 "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
 "PROCESS_INSTANCE.NAME LIKE 'A%'",
 "PROCESS_INSTANCE.NAME",
 null,null, null, null);
```

The result of the stored query is a sorted list of all the process-instance names that begin with the letter A and their associated process instance IDs (PIID).

2. Run the query defined by the stored query.

```
QueryResultSet result = process.query("Smith", "CustomerOrdersStartingWithA",
 null,null, null, null);
 new Integer(0));
```

This action returns the objects that fulfill the criteria. In this case, all of the customer orders that begin with A.

3. Get a list of the names of the private queries that belong to a specific user.

For example, the following code snippet shows how to get a list of private queries that belongs to the user Smith.

```
String[] storedQuery = process.getStoredQueryNames("Smith");
```

4. View the details of a specific query.

The following code snippet shows how to view the details of the CustomerOrdersStartingWithA query that is owned by the user Smith.

```
StoredQuery storedQuery = process.getStoredQuery
 ("Smith", "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

5. Delete a private stored query.

The following code snippet shows how to delete a private query that is owned by the user Smith.

```
process.deleteStoredQuery("Smith", "CustomerOrdersStartingWithA");
```

#### *Working with your private stored queries:*

If you are not a system administrator, you can create, run, and delete your own private stored queries. You can also use the public stored queries that the system administrator created.

1. Create a private stored query.

For example, the following code snippet creates a stored query for process instances and saves it with a specific name. If a user ID is not specified, it is assumed that the stored query is a private stored query for the logged-on user.

```
process.createStoredQuery("CustomerOrdersStartingWithA",
 "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
 "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. Run the query defined by the stored query.

```
QueryResultSet result = process.query("CustomerOrdersStartingWithA",
 new Integer(0));
```

This action returns the objects that fulfill the criteria. In this case, all of the customer orders that begin with A.

3. Get a list of the names of the stored queries that the logged-on user can access.

The following code snippet shows how to get both the public and the private stored queries that the user can access.

```
String[] storedQuery = process.getStoredQueryNames();
```

4. View the details of a specific query.

The following code snippet shows how to view the details of the CustomerOrdersStartingWithA query that is owned by the user Smith.

```
StoredQuery storedQuery = process.getStoredQuery
("CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

5. Delete a private stored query.

The following code snippet shows how to delete a private stored query.

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

## Developing applications for business processes

A business process is a set of business-related activities that are invoked in a specific sequence to achieve a business goal. Examples are provided that show how you might develop applications for typical actions on processes.

A business process can be either a microflow or a long-running process:

- Microflows are short running business processes that are executed synchronously. After a very short time, the result is returned to the caller.
- Long-running, interruptible processes are executed as a sequence of activities that are chained together. The use of certain constructs in a process causes interruptions in the process flow, for example, invoking a human task, invoking a service using an synchronous binding, or using timer-driven activities.

Parallel branches of the process are usually navigated asynchronously, that is, activities in parallel branches are executed concurrently. Depending on the type and the transaction setting of the activity, an activity can be run in its own transaction.

### Authorization roles for business processes:

Actions that you can take on business processes depend on your authorization role. This role can be a J2EE role or an instance-based role.

A role is a set of employees who share the same level of authority. Java 2 Platform, Enterprise Edition (J2EE) roles are set up when the business process container is configured. Instance-based roles are assigned to processes and activities when the process is modeled. Role-based authorization requires that global security is enabled in WebSphere Application Server.

### J2EE roles

The following J2EE roles are supported:

- J2EE BPESystemAdministrator. Users assigned to this role have all privileges. This role is also referred to as the system administrator for business processes.
- J2EE BPESystemMonitor. Users assigned to this role can view the properties of all business process objects. This role is also referred to as the system monitor for business processes.

You can use the administrative console to change the assignment of users and groups to these roles.

**Setting up Roles using RACF security:** These RACF permissions apply when the following security fields are specified:

- **com.ibm.security.SAF.authorization= true**  

```
RDEFINE EJBROLE BPESystemAdministrator UACC(NONE)
PERMIT BPESystemAdministrator CLASS(EJBROLE) ID(userid) ACCESS(READ)
RDEFINE EJBROLE BPESystemMonitor UACC(NONE)
PERMIT BPESystemMonitor CLASS(EJBROLE) ID(userid) ACCESS(READ)
```
- **com.ibm.security.SAF.delegation= true**  

```
RDEFINE EJBROLE JMSAPIUser UACC(NONE) APPLDATA(' userid')
```

You can use Security Authorization Facility (SAF)-based authorization (for example, using the RACF EJBROLE profile) to control access by a client to Java 2 Platform, Enterprise Edition (J2EE) roles in EJB and Enterprise applications, including the business process container. For more information on using SAF, see System Authorization Facility for role-based authorization in the WebSphere Application Server for z/OS information center.

### Instance-based roles

A process instance or an activity is not assigned directly to a staff member in the process model, instead it is assigned to one of the available roles. Any staff member that is assigned to an instance-based role can perform the actions for that role. The association of users to instance-based roles is determined at runtime using staff resolution.

The following instance-based roles are supported:

- For processes: reader, starter, administrator
- For activities: reader, editor, potential starter, potential owner, owner, administrator

These roles are authorized to perform the following actions:

| Role                       | Authorized actions                                                                                                             |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Activity reader            | View the properties of the associated activity instance, and its input and output messages.                                    |
| Activity editor            | Actions that are authorized for the activity reader, and write access to messages and other data associated with the activity. |
| Potential activity starter | Actions that are authorized for the activity reader. Members of this role can send messages to receive or pick activities.     |
| Potential activity owner   | Actions that are authorized for the activity reader. Members of this role can claim the activity.                              |
| Activity owner             | Work on and complete an activity. Members of this role can transfer owned work items to an administrator or a potential owner. |
| Activity administrator     | Repair activities that are stopped due to unexpected errors, and force terminate long-running activities.                      |
| Process starter            | View the properties of the associated process instance, and its input and output messages.                                     |



| Role                  | Authorized actions                                                                                                                                                                                                                                                                               |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Process reader        | View the properties of the associated process instance and its input and output messages. Process readers can also view the properties, and input and output messages for any activities that are contained in the process instance, but they cannot see any information about its subprocesses. |
| Process administrator | Members of this role can administer process instances and intervene in a process that has started; create, delete, and transfer work items. Members of this role also have activity administrator authorization.                                                                                 |

Do not delete the user ID of the process starter from your user registry if the process instance still 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 roles for actions on process instances:*

Access to the LocalBusinessFlowManager or the BusinessFlowManager interface does not guarantee that the caller can perform all of the actions on a process. The caller must be logged on to the client application with a role that is authorized to perform the action.

The following table shows the actions on a process instance that a specific role can take.

| Action                    | Caller's principal role |         |               |
|---------------------------|-------------------------|---------|---------------|
|                           | Reader                  | Starter | Administrator |
| createMessage             | x                       | x       | x             |
| createWorkItem            |                         |         | x             |
| delete                    |                         |         | x             |
| deleteWorkItem            |                         |         | x             |
| forceTerminate            |                         |         | x             |
| getActiveHandlers         | x                       | x       | x             |
| getAllActivities          | x                       |         | x             |
| getAllWorkItems           | x                       |         | x             |
| getClientUISettings       | x                       |         | x             |
| getCustomProperties       | x                       | x       | x             |
| getCustomProperty         | x                       | x       | x             |
| getCustomPropertyNames    | x                       | x       | x             |
| getFaultMessage           | x                       | x       | x             |
| getInputClientUISettings  | x                       |         | x             |
| getInputMessage           | x                       | x       | x             |
| getOutputClientUISettings | x                       |         | x             |
| getOutputMessage          | x                       | x       | x             |
| getProcessInstance        | x                       | x       | x             |
| getVariable               | x                       | x       | x             |

| Action               | Caller's principal role |         |               |
|----------------------|-------------------------|---------|---------------|
|                      | Reader                  | Starter | Administrator |
| getWaitingActivities | x                       | x       | x             |
| getWorkItems         | x                       |         | x             |
| resume               |                         |         | x             |
| restart              |                         |         | x             |
| setCustomProperty    |                         | x       | x             |
| setVariable          |                         |         | x             |
| suspend              |                         |         | x             |
| transferWorkItem     |                         |         | x             |

*Required roles for actions on business-process activities:*

Access to the LocalBusinessFlowManager or the BusinessFlowManager interface does not guarantee that the caller can perform all of the actions on an activity. The caller must be logged on to the client application with a role that is authorized to perform the action.

The following table shows the actions on an activity instance that a specific role can take.

| Action                 | Caller's principal role |        |                 |       |               |
|------------------------|-------------------------|--------|-----------------|-------|---------------|
|                        | Reader                  | Editor | Potential owner | Owner | Administrator |
| cancelClaim            |                         |        |                 | x     | x             |
| claim                  |                         |        | x               |       | x             |
| complete               |                         |        |                 | x     | x             |
| createMessage          | x                       | x      | x               | x     | x             |
| createWorkItem         |                         |        |                 |       | x             |
| deleteWorkItem         |                         |        |                 |       | x             |
| forceComplete          |                         |        |                 |       | x             |
| forceRetry             |                         |        |                 |       | x             |
| getActivityInstance    | x                       | x      | x               | x     | x             |
| getAllWorkItems        | x                       | x      | x               | x     | x             |
| getClientUISettings    | x                       | x      | x               | x     | x             |
| getCustomProperties    | x                       | x      | x               | x     | x             |
| getCustomProperty      | x                       | x      | x               | x     | x             |
| getCustomPropertyNames | x                       | x      | x               | x     | x             |
| getFaultMessage        | x                       | x      | x               | x     | x             |
| getFaultNames          | x                       | x      | x               | x     | x             |
| getInputMessage        | x                       | x      | x               | x     | x             |
| getOutputMessage       | x                       | x      | x               | x     | x             |
| getVariable            | x                       | x      | x               | x     | x             |
| getWorkItems           | x                       | x      | x               | x     | x             |
| setCustomProperty      |                         | x      |                 | x     | x             |

| Action           | Caller's principal role |        |                 |                                                 |               |
|------------------|-------------------------|--------|-----------------|-------------------------------------------------|---------------|
|                  | Reader                  | Editor | Potential owner | Owner                                           | Administrator |
| setFaultMessage  |                         | x      |                 | x                                               | x             |
| setOutputMessage |                         | x      |                 | x                                               | x             |
| setVariable      |                         |        |                 |                                                 | x             |
| transferWorkItem |                         |        |                 | x<br>To potential owners or administrators only | x             |

### Managing the life cycle of a business process:

A process instance comes into existence when a Business Process Choreographer API method that can start a process is invoked. The navigation of the process instance continues until all of its activities are in an end state. Various actions can be taken on the process instance to manage its life cycle.

Examples are provided that show how you might develop applications for the following typical life-cycle actions on processes.

#### *Starting business processes:*

The way in which a business process is started depends on whether the process is a microflow or a long-running process. The service that starts the process is also important to the way in which a process is started; the process can have either a unique starting service or several starting services.

Examples are provided that show how you might develop applications for typical scenarios for starting microflows and long-running processes.

#### *Running a microflow that contains a unique starting service:*

A microflow can be started by a receive activity or a pick activity. The starting service is unique if the microflow starts with a receive activity or when the pick activity has only one onMessage definition.

If the microflow implements a request-response operation, that is, the process contains a reply, you can use the call method to run the process passing the process template name as a parameter in the call.

If the microflow is a one-way operation, use the sendMessage method to run the process. This method is not covered in this example.

1. **Optional:** List the process templates to find the name of the 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 =
PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_MICROFLOW",
"PROCESS_TEMPLATE.NAME",
new Integer(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.getInputMessageType());
DataObject myMessage = null;
if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the strings in the message, for example, a customer name
 myMessage.setString("CustomerName", "Smith");
}

//run the process
ClientObjectWrapper output = process.call(template.getName(), input);
DataObject myOutput = null;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myOutput = (DataObject)output.getObject();
 int order = myOutput.getInt("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 microflow that contains a non-unique starting service:*

A microflow can be started by a receive activity or a pick activity. The starting service is not unique if the microflow starts with a pick activity that has multiple *onMessage* definitions.

If the microflow implements a request-response operation, that is, the process contains a reply, you can use the call method to run the process passing the ID of the starting service in the call.

If the microflow is a one-way operation, use the *sendMessage* method to run the process. This method is not covered in this example.

1. **Optional:** List the process templates to find the name of the 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 =
 PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_MICROFLOW",
 "PROCESS_TEMPLATE.NAME",
 new Integer(50),
 null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as microflows.

2. Determine the starting service to be called.

This example uses the first template that is found.

```
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.getInputMessageType());

DataObject myMessage = null;
if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the strings in the message, for example, a customer name
 myMessage.setString("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
DataObject myOutput = null;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myOutput = (DataObject)output.getObject();
 int order = myOutput.getInt("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.

*Starting a long-running process that contains a unique starting service:*

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 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 =
 PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_LONG_RUNNING",
 "PROCESS_TEMPLATE.NAME",
 new Integer(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.getInputMessageType());

DataObject myMessage = null;

```

```

if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the strings in the message, for example, a customer name
 myMessage.setString("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. This person 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 a long-running process that contains a non-unique starting service:*

A long-running 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 or pick 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 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 =
 PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_LONG_RUNNING",
"PROCESS_TEMPLATE.NAME",
new Integer(50),
null);

```

The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as long-running 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.getInputMessageType());
DataObject myMessage = null;
if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the strings in the message, for example, a customer name
 myMessage.setString("CustomerName", "Smith");
}

```

```

}
//start the process
PIID piid = process.sendMessage(activity.getServiceTemplateID(),
 activity.getActivityTemplateID(),
 input);

```

This action creates an instance 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.

#### *Suspending and resuming a business process:*

You can suspend long-running, top-level process instance while it is running and resume it again to complete it.

The caller must be an administrator of the process instance or a business process administrator. To suspend a process instance, it must be in the running or failing state.

You might want to suspend a process instance, for example, so that you can configure access to a back-end system that is used later in the process. When the prerequisites for the process are met, you can resume the process instance. You might also want to suspend a process to fix a problem that is causing the process instance to fail, and then resume it again when the problem is fixed.

1. Get the running process, `CustomerOrder`, that you want to suspend.

```

ProcessInstanceData processInstance =
 process.getProcessInstance("CustomerOrder");

```

2. Suspend the process instance.

```

PIID piid = processInstance.getID();
process.suspend(piid);

```

This action suspends the specified top-level process instance. The process instance is put into the suspended state. Subprocesses with the `autonomy` attribute set to `child` are also suspended if they are in the running, failing, terminating, or compensating state. Inline tasks that are associated with this process instance are also suspended; standalone tasks associated with this process instance are not suspended.

In this state, activities that are started can still be finished but no new activities are activated, for example, a staff activity in the claimed state can be completed.

3. Resume the process instance.

```

process.resume(piid);

```

This action puts the process instance and its subprocesses into the states they had before they were suspended.

#### *Restarting a business process:*

You can restart a process instance that is in the finished, terminated, failed, or compensated state.

The caller must be an administrator of the process instance or a business process administrator.

Restarting a process instance is similar to starting a process instance for the first time. However, when a process instance is restarted, the process instance ID is known and the input message for the instance is available.

If the process has more than one receive activity or pick activity (also known as a receive choice activity) that can create the process instance, all of the messages that belong to these activities are used to restart the process instance. If any of these activities implement a request-response operation, the response is sent again when the associated reply activity is navigated.

1. Get the process that you want to restart.

```
ProcessInstanceData processInstance =
 process.getProcessInstance("CustomerOrder");
```

2. Restart the process instance.

```
PIID piid = processInstance.getID();
process.restart(piid);
```

This action restarts the specified process instance.

*Terminating a process instance:*

Sometimes, it is necessary for someone with process administrator authorization to terminate a top-level process instance that is known to be in an unrecoverable state. Because a process instance terminates immediately, without waiting for any outstanding subprocesses or activities, you should terminate a process instance only in exceptional situations.

1. Retrieve the process instance that is to be terminated.

```
ProcessInstanceData processInstance =
 process.getProcessInstance("CustomerOrder");
```

2. Terminate the process instance.

If you terminate a process instance, you can terminate the process instance with or without compensation.

To terminate the process instance with compensation:

```
PIID piid = processInstance.getID();
process.forceTerminate(piid, CompensationBehaviour.INVOKE_COMPENSATION);
```

To terminate the process instance without compensation:

```
PIID piid = processInstance.getID();
process.forceTerminate(piid);
```

If you terminate the process instance with compensation, the compensation handler defined for the process template is called. If the process template does not have a compensation handler defined, the default compensation handler is called. If you terminated the process instance without compensation, the process instance is terminated immediately without waiting for activities, participating tasks, or inline originating tasks to end normally.

Applications that are started by the process and standalone tasks that are related to the process are not terminated by the force terminate request. If these applications are to be terminated, you must add statements to your process application that explicitly terminate the applications started by the process.

*Deleting process instances:*

Completed process instances are automatically deleted from the Business Process Choreographer database if the corresponding property is set for the process



template in the process model. You might want to keep process instances in your database, for example, to query data from process instances that are not written to the audit log. However, stored process instance data does not only impact disk space and performance but also prevents process instances that use the same correlation set values from being created. Therefore, you should regularly delete process instance data from the database.

To delete a process instance, you need process administrator rights and the process instance must be a top-level process instance.

The following example shows how to delete all of the 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 process instances that are finished.

2. Delete the process instances that are finished.

```
while (result.next())
{
 PIID piid = (PIID) result.getOID(1);
 process.delete(piid);
}
```

This action deletes the selected process instance and its inline tasks from the database.

### Processing staff activities:

Staff activities in business processes are assigned to various people in your organization through work items. When a process is started, work items are created for the potential owners.

A potential owner 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 aaid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aaid);
 DataObject activityInput = null ;
 if (input.getObject() != null && input.getObject() instanceof DataObject)
 {
```

```

 activityInput = (DataObject)input.getObject();
 // read the values
 ...
 }
}

```

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 for these actions. 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.getOutputMessageType());
DataObject myMessage = null ;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myMessage = (DataObject)output.getObject();
 //set the parts in your message, for example, an order number
 myMessage.setInt("OrderNo", 4711);
}

//complete the activity
process.complete(aiid, output);

```

This action sets an output message that contains 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
DataObject myMessage = null ;
if (myFault.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)myFault.getObject();
 //set the parts in the message, for example, a customer name
 myMessage.setInt("error",1304);
}

process.complete(aiid, (String)faultNames.get(0), myFault);

```

This action sets the activity in either the failed or the stopped state. If the **continueOnError** 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 **continueOnError** parameter is set to false and the fault is not caught on the surrounding scope, the activity is put into the stopped state. In this state the activity can be repaired using force terminate or force retry.

### Processing a single person workflow:

Some workflows are performed by only one person, for example, ordering books from an online bookstore. This type of workflow has no parallel paths. The `completeAndClaimSuccessor` API supports the processing of this type of workflow.

In an online bookstore, the purchaser completes a sequence of actions to order a book. This sequence of actions can be implemented as a series of staff activities (participating tasks). If the purchaser decides to order several books, this is equivalent to claiming the next staff activity. This type of workflow is also known as *page flow* because user interface definitions are associated with the activities to control the flow of the dialogs in the user interface.

The `completeAndClaimSuccessor` API completes a staff activity and claims the next one in the same process instance for the logged-on person. It returns information about the next claimed activity, including the input message to be worked on. Because the next activity is made available within the same transaction of the activity that completed, the transactional boundaries must be set in the process model to participate.

1. Claim the first activity in the sequence of activities.

```
//
//Query the list of activities that can be claimed by the logged-on user
//
QueryResultSet result =
 process.query("ACTIVITY.AIID",
 "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
 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);

...
//
//Claim the first activity
//
if (result.size() > 0)
{
 result.first();
 AIID aaid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aaid);
 DataObject activityInput = null ;
 if (input.getObject() != null && input.getObject() instanceof DataObject)
 {
 activityInput = (DataObject)input.getObject();
 // read the values
 ...
 }
}
```

When the activity is claimed, the input message of the activity is returned.

2. When work on the activity is finished, complete the activity, and claim the next activity.

To complete the activity, an output message is passed. When you create the output message, you must specify the message type name so that the message definition is contained.

```
ActivityInstanceData activity = process.getActivityInstance(aaid);
ClientObjectWrapper output =
 process.createMessage(aaid, activity.getOutputMessageType());
DataObject myMessage = null ;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myMessage = (DataObject)output.getObject();
 //set the parts in your message, for example, an order number
```

```

 myMessage.setInt("OrderNo", 4711);
}

//complete the activity and claim the next one
CompleteAndClaimSuccessorResult successor =
 process.completeAndClaimSuccessor(aiid, output);

```

This action sets an output message that contains the item number and claims the next activity in the sequence. If `AutoClaim` is set for successor activities, a random activity is returned as the next activity. If there are no more successor activities that can be assigned to this user, `Null` is returned. If `AutoClaim` is set for successor activities and if there are multiple paths that can be followed, all of the successor activities are claimed and a random activity is returned as the next activity.

If the process contains parallel paths that can be followed and these paths contain staff activities for which the logged-on user is a potential owner of more than one of these activities, a random activity is claimed automatically and returned as the next activity.

### 3. Work on the next activity.

```

String name = successor.getActivityName();

ClientObjectWrapper nextInput = successor.getInputMessage();
if (nextInput.getObject() !=
 null && nextInput.getObject() instanceof DataObject)
{
 activityInput = (DataObject)input.getObject();
 // read the values
 ...
}

```

### 4. Continue with step 2 to complete the activity.

#### **Sending a message to a waiting activity:**

You can use inbound message activities (receive activities, `onMessage` in pick activities, `onEvent` in event handlers) to synchronize a running process with events from the "outside world". For example, the receipt of an e-mail from a customer in response to a request for information might be such an event.

You can use originating tasks to send the message to the activity.

#### 1. List the activity service templates that are waiting for a message from the logged-on user in a process instance with a specific process instance ID.

```

ActivityServiceTemplateData[] services = process.getWaitingActivities(piid);

```

#### 2. Send a message to the first waiting service.

It is assumed that the first service is the one that you want serve. The caller must be a potential starter of the activity that receives the message, or an administrator of the process instance.

```

VTID vtid = services[0].getServiceTemplateID();
ATID atid = services[0].getActivityTemplateID();
String inputType = services[0].getInputMessageType();

// create a message for the service to be called
ClientObjectWrapper message =
 process.createMessage(vtid,atid,inputMessageType);
DataObject myMessage = null;
if (message.getObject() != null && message.getObject() instanceof DataObject)
{
 myMessage = (DataObject)message.getObject();
 //set the strings in the message, for example, chocolate is to be ordered
}

```

```

 myMessage.setString("Order", "chocolate");
 }

 // send the message to the waiting activity
 process.sendMessage(vtid, atid, message);
}

```

This action 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 the process instance that is identified by the correlation values in the message. If the process instance ID is specified, the process instance that is found using the correlation values is checked to ensure that it has the specified process instance ID.

### Handling events:

An entire business process and each of its scopes can be associated with event handlers that are invoked if the associated event occurs. Event handlers are similar to receive or pick activities in that a process can provide Web service operations using event handlers.

You can invoke an event handler any number of times as long as the corresponding scope is running. In addition, multiple instances of an event handler can be activated concurrently.

The following code snippet shows how to get the active event handlers for a given process instance and how to send an input message.

1. Determine the data of the process instance ID and list the active event handlers for the process.

```

ProcessInstanceData processInstance =
 process.getProcessInstance("CustomerOrder2711");
EventHandlerTemplateData[] events = process.getActiveEventHandlers(
 processInstance.getID());

```

2. Send the input message.

This example uses the first event handler that is found.

```

EventHandlerTemplateData event = null;
if (events.length > 0)
{
 event = events[0];

 // create a message for the service to be called
 ClientObjectWrapper input = process.createMessage(
 event.getID(), event.getInputMessageType());

 if (input.getObject() != null && input.getObject() instanceof DataObject)
 {
 DataObject inputMessage = (DataObject)input.getObject();
 // set content of the message, for example, a customer name, order number
 inputMessage.setString("CustomerName", "Smith");
 inputMessage.setString("OrderNo", "2711");

 // send the message
 process.sendMessage(event.getProcessTemplateName(),
 event.getPortTypeNamespace(),
 event.getPortTypeName(),
 event.getOperationName(),

 input);
 }
}

```

This action sends the specified message to the active event handler for the process.

### Analyzing the results of a process:

A process can expose Web services operations that are modeled as Web Services Description Language (WSDL) one-way or request-response operations. If a long-running process exposes a one-way operation, the results of the process, such as the values of process variables, must be retrieved from the database.

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);
 DataObject myOutput = null;
 if (output.getObject() != null && output.getObject() instanceof DataObject)
 {
 myOutput = (DataObject)output.getObject();
 int order = myOutput.getInt("OrderNo");
 }
}
```

### Repairing activities:

A long-running process can contain activities that are also long running. These activities might encounter uncaught errors and go into the stopped state. An activity in the running state might also appear to be not responding. In both of these cases, a process administrator can act on the activity in a number of ways so that the navigation of the process can continue.

The Business Process Choreographer API provides the `forceRetry` and `forceComplete` methods for repairing activities. Examples are provided that show how you might add repair actions for activities to your applications.

#### *Forcing the completion of an activity:*

Activities in long-running processes can sometimes encounter faults. If these faults are not caught by a fault handler 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 the completion of the activity.

You can also force the completion of activities in the running state if, for example, an activity is not responding.

Additional requirements exist for certain types of activities.

### Staff activities

You can pass parameters in the force-complete call, such as the message that should have been sent or the fault that should have been raised.

### Script activities

You cannot pass parameters in the force-complete call. However, you must set the variables that need to be repaired.

### Invoke activities

You can also force the completion of invoke activities that call an asynchronous service that is not a subprocess if these activities are in the running state. You might want to do this, for example, if the asynchronous service is called and it does not respond.

1. List the stopped activities in the stopped state.

```
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, for example, a stopped staff activity.

In this example, an output message is passed.

```
if (result.size() > 0)
{
 result.first();
 AIID aaid = (AIID) result.getOID(1);
 ActivityInstanceData activity = process.getActivityInstance(aaid);
 ClientObjectWrapper output =
 process.createMessage(aaid, activity.getOutputMessageType());
 DataObject myMessage = null;
 if (output.getObject() != null && output.getObject() instanceof DataObject)
 {
 myMessage = (DataObject)output.getObject();
 //set the parts in your message, for example, an order number
 myMessage.setInt("OrderNo", 4711);
 }

 boolean continueOnError = true;
 process.forceComplete(aaid, output, continueOnError);
}
```

This action completes the activity. If an error occurs, the **continueOnError** parameter determines the action to be taken if a fault is provided with the forceComplete request.

In the example, **continueOnError** is true. This value means that if a fault is provided, 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 it eventually reaches the failed state.

#### *Retrying the execution of a stopped activity:*

If an activity in a long-running process encounters 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. With the exception of script activities, 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, for example, a stopped staff activity.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ActivityInstanceData activity = process.getActivityInstance(aiid);
 ClientObjectWrapper input =
 process.createMessage(aiid, activity.getOutputMessageType());
 DataObject myMessage = null;
 if (input.getObject() != null && input.getObject() instanceof DataObject)
 {
 myMessage = (DataObject)input.getObject();
 //set the strings in your message, for example, chocolate is to be ordered
 myMessage.setString("OrderNo", "chocolate");
 }

 boolean continueOnError = true;
 process.forceRetry(aiid, input, continueOnError);
}
```

This action retries the activity. If an error occurs, the **continueOnError** parameter determines the action to be taken if an error occurs during processing of the forceRetry request.

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 it eventually reaches the failed state.

### **BusinessFlowManagerService interface:**

The BusinessFlowManagerService interface exposes business-process functions that can be called by a client application.

The methods that can be called by the BusinessFlowManagerService interface depend on the state of the process or the activity and the authorization of the person that uses the application containing the method. The main methods for manipulating business process objects are listed here. For more information about these methods and the other methods that are available in the BusinessFlowManagerService interface, see the Javadoc in the com.ibm.bpe.api package.

### **Process templates**

A process template is a versioned, deployed, and installed process model that contains the specification of a business process. It can be instantiated and started by issuing appropriate requests, for example, sendMessage(). The execution of the process instance is driven automatically by the server.



Table 21. API methods for process templates

| Method               | Description                                                  |
|----------------------|--------------------------------------------------------------|
| getProcessTemplate   | Retrieves the specified process template.                    |
| queryProcessTemplate | Retrieves process templates that are stored in the database. |

## Process instances

The following API methods start process instances.

Table 22. API methods for starting process instances

| Method                     | Description                                                                                                                                                                                                                                       |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| call                       | Creates and runs a microflow.                                                                                                                                                                                                                     |
| callWithReplyContext       | Creates and runs a microflow with a unique starting service or a long-running process with a unique starting service from the specified process template. The call waits asynchronously for the result.                                           |
| callWithUISettings         | Creates and runs a microflow and returns the output message and the client user interface (UI) settings.                                                                                                                                          |
| initiate                   | Creates a process instance and initiates processing of the process instance. Use this method for long-running processes. You can also use this method for microflows that you want to fire and forget.                                            |
| sendMessage                | Sends the specified message to the specified activity service and process instance. If a process instance with the same correlation set values does not exist, it is created. The process can have either unique or non-unique starting services. |
| getStartActivities         | Returns information about the activities that can start a process instance from the specified process template.                                                                                                                                   |
| getActivityServiceTemplate | Retrieves the specified activity service template.                                                                                                                                                                                                |

Table 23. API methods for controlling the life cycle of process instances

| Method  | Description                                                                                                   |
|---------|---------------------------------------------------------------------------------------------------------------|
| suspend | Suspends the execution of a long-running, top-level process instance that is in the running or failing state. |
| resume  | Resumes the execution of a long-running, top-level process instance that is in the suspended state.           |
| restart | Restarts a long-running, top-level process instance that is in the finished, failed, or terminated state.     |

Table 23. API methods for controlling the life cycle of process instances (continued)

| Method         | Description                                                                                                                                 |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| forceTerminate | Terminates the specified top-level process instance, its subprocesses with child autonomy, and its running, claimed, or waiting activities. |
| delete         | Deletes the specified top-level process instance and its subprocesses with child autonomy.                                                  |
| query          | Retrieves the properties from the database that match the search criteria.                                                                  |

## Activities

For invoke activities, you can specify in the process model that these activities continue in error situations. If the `continueOnError` flag is set to `false` and an unhandled error occurs, the activity is put into the stopped state. A process administrator can then repair the activity. The `continueOnError` flag and the associated repair functions can, for example, be used in a long-running process where an invoke activity fails occasionally, but the effort required to model compensation and fault handling is too high.

The following methods are available for working with and repairing activities.

Table 24. API methods for controlling the life cycle of activity instances

| Method                    | Description                                                                                               |
|---------------------------|-----------------------------------------------------------------------------------------------------------|
| claim                     | Claims a ready activity instance for a user to work on the activity.                                      |
| cancelClaim               | Cancels the claim of the activity instance.                                                               |
| complete                  | Completes the activity instance.                                                                          |
| completeAndClaimSuccessor | Completes a staff activity and claims the next one in the same process instance for the logged-on person. |
| forceComplete             | Forces the completion of an activity instance that is in the running or stopped state.                    |
| forceRetry                | Forces the repetition of an activity instance that is in the running or stopped state.                    |
| query                     | Retrieves the properties from the database that match the search criteria.                                |

## Variables and custom properties

The interface provides a `get` and a `set` method to retrieve and set values for variables. You can also associate named properties with, and retrieve named properties from, process and activity instances. Custom property names and values must be of the `java.lang.String` type.

Table 25. API methods for variables and custom properties

| Method      | Description                       |
|-------------|-----------------------------------|
| getVariable | Retrieves the specified variable. |
| setVariable | Sets the specified variable.      |

Table 25. API methods for variables and custom properties (continued)

| Method                 | Description                                                                                  |
|------------------------|----------------------------------------------------------------------------------------------|
| getCustomProperty      | Retrieves the named custom property of the specified activity or process instance.           |
| getCustomProperties    | Retrieves the custom properties of the specified activity or process instance.               |
| getCustomPropertyNames | Retrieves the names of the custom properties for the specified activity or process instance. |
| setCustomProperty      | Stores custom-specific values for the specified activity or process instance.                |

## Developing applications for human tasks

A task is the means by which components invoke humans as services or by which humans invoke services. Examples of typical applications for human tasks are provided.

For more information on the Business Process Choreographer API, see the Javadoc in the `com.ibm.task.api` package.

### Authorization roles for human tasks:

Actions that you can take on human tasks depend on your authorization role. This role can be a J2EE role or an instance-based role.

A role is a set of employees who share the same level of authority. Java 2 Platform, Enterprise Edition (J2EE) roles are set up when the human task container is configured. Instance-based roles are assigned to human tasks and escalations when the task is modeled. Role-based authorization requires that global security is enabled in WebSphere Application Server.

### J2EE roles

The following J2EE roles are supported:

- **J2EE TaskSystemAdministrator.** Users assigned to this role have all privileges. This role is also referred to as the system administrator for human tasks.
- **J2EE TaskSystemMonitor.** Users assigned to this role can view the properties of all of the task objects. This role is also referred to as the system monitor for human tasks.

You can use the administrative console to change the assignment of users and groups to these roles.

**Setting up Roles using RACF security:** These RACF permissions apply when the following security fields are specified:

- **com.ibm.security.SAF.authorization= true**  

```

DEFINE EJBROLE TaskSystemAdministrator UACC(NONE)
PERMIT TaskSystemAdministrator CLASS(EJBROLE) ID(userid) ACCESS(READ)
DEFINE EJBROLE TaskSystemMonitor UACC(NONE)
PERMIT TaskSystemMonitor CLASS(EJBROLE) ID(userid) ACCESS(READ)

```
- **com.ibm.security.SAF.delegation= true**  

```

DEFINE EJBROLE JMSAPIUser UACC(NONE) APPLDATA('userid')

```

You can use Security Authorization Facility (SAF)-based authorization (for example, using the RACF EJBROLE profile) to control access by a client to Java 2 Platform, Enterprise Edition (J2EE) roles in EJB and Web applications, including the WebSphere Application Server administrative console application. For more information, see System Authorization Facility for role-based authorization in the WebSphere Application Server for z/OS information center.

### Instance-based roles

A task instance or an escalation instance is not assigned directly to a staff member in the task model, instead it is assigned to one of the available roles. Any staff member that is assigned to an instance-based role can perform the actions for that role. The association of users to instance-based roles is determined at runtime using staff resolution.

The following instance-based roles are supported:

- For tasks: potential instance creator, originator, potential starter, starter, potential owner, owner, reader, editor, administrator
- For escalations: escalation receiver

These roles are authorized to perform the following actions:

| Role                       | Authorized actions                                                                                                                                                                                                                                                                             |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Potential instance creator | Members of this role can create an instance of the task. If no potential instance creator is defined for the task template or the application components, then all users are considered to be a member of this role.                                                                           |
| Originator                 | Members of this role have administrative rights until the task starts. When the task starts, the originator has the authority of a reader and can perform some administrative actions, such as suspending and resuming tasks, and transferring work items.                                     |
| Potential starter          | Members of this role can start an existing task instance. If a potential starter is not specified, the originator becomes the potential starter. For inline tasks without a potential starter, the default is everybody.                                                                       |
| Starter                    | Members of this role have the authority of a reader and can perform some administrative actions, such as transferring work items.                                                                                                                                                              |
| Potential owner            | Members of this role can claim a task. If no potential owner is defined for the task template or the application components, then all users are considered to be a member of this role. If staff resolution fails for this role, then the administrators are assigned as the potential owners. |
| Owner                      | Work on and complete a task.                                                                                                                                                                                                                                                                   |
| Reader                     | View the properties of all of the task objects, but cannot work on them.                                                                                                                                                                                                                       |
| Editor                     | Members of this role can work with the content of a task, but cannot claim or complete it                                                                                                                                                                                                      |
| Administrator              | Members of this role can administer tasks, task templates, and escalations.                                                                                                                                                                                                                    |
| Escalation receiver        | Members of this role have the authority of a reader for the escalation and the escalated task.                                                                                                                                                                                                 |

*Required roles for actions on tasks:*

Access to the LocalHumanTaskManager or the HumanTaskManager interface does not guarantee that the caller can perform all of the actions on a task; the caller must also be authorized to perform the action. The following table shows the actions that a specific role can take.

| Action                                 | Caller's principal role |           |                |                |                  |                |        |        |              |
|----------------------------------------|-------------------------|-----------|----------------|----------------|------------------|----------------|--------|--------|--------------|
|                                        | Owner                   | Pot owner | Starter        | Pot starter    | Origin           | Admin          | Editor | Reader | Esc receiver |
| callTask                               |                         |           |                | X <sup>1</sup> | X <sup>1</sup>   | X <sup>1</sup> |        |        |              |
| cancelClaim                            | X                       |           |                |                |                  | X              |        |        |              |
| claim                                  |                         | X         |                |                |                  | X              |        |        |              |
| complete                               | X                       |           |                |                |                  | X              |        |        |              |
| completeWithFollowOn Task <sup>3</sup> | X                       |           |                |                |                  | X              |        |        |              |
| createFaultMessage                     | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| createInputMessage                     | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| createOutputMessage                    | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| createWorkItem                         |                         |           |                |                | X <sup>1,2</sup> | X              |        |        |              |
| delete                                 |                         |           |                |                | X                | X              |        |        |              |
| deleteWorkItem                         |                         |           |                |                | X <sup>1,2</sup> | X              |        |        |              |
| getCustomProperty                      | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getDocumentation                       | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getFaultMessage                        | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getFaultNames                          | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getInputMessage                        | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getOutputMessage                       | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getRoleInfo                            | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getTask                                | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| getUISettings                          | X                       | X         | X              | X              | X <sup>1</sup>   | X              | X      | X      | X            |
| resume                                 | X                       |           |                |                | X <sup>1</sup>   | X              |        |        |              |
| setCustomProperty                      | X                       |           | X              |                | X <sup>1</sup>   | X              | X      |        |              |
| setFaultMessage                        | X                       |           |                |                |                  | X              | X      |        |              |
| setOutputMessage                       | X                       |           |                |                |                  | X              | X      |        |              |
| startTask                              |                         |           |                | X              | X <sup>1</sup>   | X              |        |        |              |
| startTaskAsSubtask <sup>4</sup>        | X                       |           |                |                |                  | X              |        |        |              |
| suspend                                | X                       |           |                |                | X <sup>1</sup>   | X              |        |        |              |
| suspendWithCancelClaim                 | X                       |           |                |                |                  | X              |        |        |              |
| terminate                              | X                       |           | X <sup>1</sup> |                | X <sup>1</sup>   | X              |        |        |              |
| transferWorkItem                       | X                       |           | X              |                | X <sup>6</sup>   | X              |        |        |              |
| update                                 | X                       |           | X              |                | X <sup>1</sup>   | X              | X      |        |              |
| updateInactiveTask                     |                         |           |                |                | X <sup>5</sup>   |                |        |        |              |

| Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Caller's principal role |           |         |             |        |       |        |        |              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|-----------|---------|-------------|--------|-------|--------|--------|--------------|
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Owner                   | Pot owner | Starter | Pot starter | Origin | Admin | Editor | Reader | Esc receiver |
| <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. For stand-alone tasks, ad-hoc tasks, and tasks derived from task templates only.</li> <li>2. For potential owner, potential starter, editor, reader and escalation receiver work items only.</li> <li>3. The caller must also have at least task reader rights to the follow-on task.</li> <li>4. The caller must also have at least task reader rights to the subtask.</li> <li>5. For stand-alone tasks and ad-hoc tasks only.</li> <li>6. For potential owner, potential starter, originator, editor, reader and escalation receiver work items only.</li> </ol> <p><b>Abbreviations:</b></p> <p><b>Admin</b> Administrator</p> <p><b>Esc receiver</b><br/>Escalation receiver</p> <p><b>Origin</b> Originator</p> <p><b>Pot owner</b><br/>Potential owner</p> <p><b>Pot starter</b><br/>Potential starter</p> |                         |           |         |             |        |       |        |        |              |

### Starting an originating task that invokes a synchronous interface:

Originating tasks that invoke a synchronous interface include inline originating tasks in a microflow, stand-alone originating tasks in a microflow, and originating tasks that start an SCA (Service Component Architecture) component that is implemented, for example, by a simple Java class.

This scenario creates an instance of a task template and passes some customer data. The task remains in the running state until the two-way operation returns. The result of the task, `OrderNo`, is returned to the caller.

1. **Optional:** List the task templates to find the name of the originating task you want to run.

This step is optional if you already know the name of the task.

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates(
 "TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
 "TASK_TEMPL.NAME",
 new Integer(50),
 null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted originating templates.

2. Create an input message of the appropriate type.

```
TaskTemplate template = taskTemplates[0];

// create a message for the selected task
ClientObjectWrapper input = task.createInputMessage(template.getID());
DataObject myMessage = null ;
if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the parts in the message, for example, a customer name
 myMessage.setString("CustomerName", "Smith");
}
```

3. Create the task and run the task synchronously.

For a task to run synchronously, it must be a two-way operation. The example uses the `createAndCallTask` method to create and run the task.

```
ClientObjectWrapper output = task.createAndCallTask(template.getName(),
 template.getNamespace(),
 input);
```

4. Analyze the result of the task.

```
DataObject myOutput = null;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myOutput = (DataObject)output.getObject();
 int order = myOutput.getInt("OrderNo");
}
```

### Starting an originating task that invokes an asynchronous interface:

Originating tasks that invoke an asynchronous interface include inline originating tasks in a microflow, stand-alone originating tasks in a microflow, and originating tasks that start an SCA (Service Component Architecture) component that is implemented, for example, by a simple Java class.

This scenario creates an instance of a task template and passes some customer data.

1. **Optional:** List the task templates to find the name of the originating task you want to run.

This step is optional if you already know the name of the task.

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates
("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
 "TASK_TEMPL.NAME",
 new Integer(50),
 null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted originating templates.

2. Create an input message of the appropriate type.

```
TaskTemplate template = taskTemplates[0];

// create a message for the selected task
ClientObjectWrapper input = task.createInputMessage(template.getID());
DataObject myMessage = null ;
if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the parts in the message, for example, a customer name
 myMessage.setString("CustomerName", "Smith");
}
```

3. Create the task and run it asynchronously.

The example uses the `createAndStartTask` method to create and run the task.

```
task.createAndStartTask(template.getName(),
 template.getNamespace(),
 input,
 null);
```

### Creating and starting a task instance:

This scenario shows how to create an instance of a task template that defines a human task and start the task instance.

1. **Optional:** List the task templates to find the name of the originating task you want to run.

This step is optional if you already know the name of the task.

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates
("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_HUMAN",
 "TASK_TEMPL.NAME",
 new Integer(50),
 null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted human task templates.

2. Create an input message of the appropriate type.

```
TaskTemplate template = taskTemplates[0];

// create a message for the selected task
ClientObjectWrapper input = task.createInputMessage(template.getID());
DataObject myMessage = null ;
if (input.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)input.getObject();
 //set the parts in the message, for example, a customer name
 myMessage.setString("CustomerName", "Smith");
}
```

3. Create and start the human task; a reply handler is not specified in this example.

The example uses the createAndStartTask method to create and start the task.

```
TKIID tkiid = task.createAndStartTask(template.getName(),
 template.getNamespace(),
 input,
 null);
```

Work items are created for the people concerned with the task instance. For example, a potential owner can claim the new task instance.

4. Claim the task instance.

```
ClientObjectWrapper input2 = task.claim(tkiid);
DataObject taskInput = null ;
if (input2.getObject() != null && input2.getObject() instanceof DataObject)
{
 taskInput = (DataObject)input2.getObject();
 // read the values
 ...
}
```

When the task instance is claimed, the input message of the task is returned.

### Processing participating or purely human tasks:

Participating or purely human tasks are assigned to various people in your organization through work items. Participating tasks and their associated work items are created, for example, when a process navigates to a staff activity. One of the potential owners claims the task associated with the work item. This person is responsible for providing the relevant information and completing the task.

1. List the tasks belonging to a logged-on person that are ready to be worked on.

```
QueryResultSet result =
 task.query("TASK.TKIID",
 "TASK.STATE = TASK.STATE.STATE_READY AND
 (TASK.KIND = TASK.KIND.KIND_PARTICIPATING OR
 TASK.KIND = TASK.KIND.KIND_HUMAN)AND
 WORK_ITEM.REASON =
 WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
 null, null, null);
```



This action returns a query result set that contains the tasks that can be worked on by the logged-on person.

2. Claim the task to be worked on.

```
if (result.size() > 0)
{
 result.first();
 TKIID tkiid = (TKIID) result.getOID(1);
 ClientObjectWrapper input = task.claim(tkiid);
 DataObject taskInput = null ;
 if (input.getObject() != null && input.getObject() instanceof DataObject)
 {
 taskInput = (DataObject)input.getObject();
 // read the values
 ...
 }
}
```

When the task is claimed, the input message of the task is returned.

3. When work on the task is finished, complete the task.

The task can be completed either successfully or with a fault message. If the task is successful, an output message is passed. If the task is unsuccessful, a fault message is passed. You must create the appropriate messages for these actions.

- a. To complete the task successfully, create an output message.

```
ClientObjectWrapper output =
 task.createOutputMessage(tkiid);
DataObject myMessage = null ;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myMessage = (DataObject)output.getObject();
 //set the parts in your message, for example, an order number
 myMessage.setInt("OrderNo", 4711);
}

//complete the task
task.complete(tkiid, output);
```

This action sets an output message that contains the order number. The task is put into the finished state.

- b. To complete the task when a fault occurs, create a fault message.

```
//retrieve the faults modeled for the task
List faultNames = task.getFaultNames(tkiid);

//create a message of the appropriate type
ClientObjectWrapper myFault =
 task.createFaultMessage(tkiid, (String)faultNames.get(0));

// set the parts in your fault message, for example, an error number
DataObject myMessage = null ;
if (myFault.getObject() != null && input.getObject() instanceof DataObject)
{
 myMessage = (DataObject)myFault.getObject();
 //set the parts in the message, for example, a customer name
 myMessage.setInt("error",1304);
}

task.complete(tkiid, (String)faultNames.get(0), myFault);
```

This action sets a fault message that contains the error code. The task is put into the failed state.

### **Suspending and resuming a task instance:**

You can suspend human task instances or participating task instances and resume them again to complete them.

The task instance can be in the ready or claimed state. It can be escalated. The caller must be the owner, originator, or administrator of the task instance.

You can suspend a task instance while it is running. You might want to do this, for example, so that you can gather information that is needed to complete the task. When the information is available, you can resume the task instance.

1. Get a list of tasks that are claimed by the logged-on user.

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
 "TASK.STATE = TASK.STATE.STATE_CLAIMED",
 null, null, null);
```

This action returns a query result set that contains a list of the tasks that are claimed by the logged-on user.

2. Suspend the task instance.

```
if (result.size() > 0)
{
 result.first();
 TKIID tkiid = (TKIID) result.getOID(1);
 task.suspend(tkiid);
}
```

This action suspends the specified task instance. The task instance is put into the suspended state.

3. Resume the process instance.

```
task.resume(tkiid);
```

This action puts the task instance into the state it had before it was suspended.

### Analyzing the results of a task:

A participating or purely human task runs asynchronously. If a reply handler is specified when the task starts, the output message is automatically returned when the task completes. If a reply handler is not specified, the message must be retrieved explicitly.

The results of the task are stored in the database only if the task template from which the task instance was derived does not specify automatic deletion of the derived task instances.

Analyze the results of the task.

The example shows how to check the order number of a successfully completed task.

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
 "TASK.NAME = 'CustomerOrder' AND
 TASK.STATE = TASK.STATE.STATE_FINISHED",
 null, null, null);

if (result.size() > 0)
{
 result.first();
 TKIID tkiid = (TKIID) result.getOID(1);
 ClientObjectWrapper output = task.getOutputMessage(tkiid);
 DataObject myOutput = null;
 if (output.getObject() != null && output.getObject() instanceof DataObject)
 {
```

```

 myOutput = (DataObject)output.getObject();
 int order = myOutput.getInt("OrderNo");
 }
}

```

### Terminating a task instance:

Sometimes it is necessary for someone with administrator rights to terminate a task instance that is known to be in an unrecoverable state. Because the task instance is terminated immediately, you should terminate a task instance only in exceptional situations.

1. Retrieve the task instance to be terminated.

```
Task taskInstance = task.getTask(tkiid);
```

2. Terminate the task instance.

```
TKIID tkiid = taskInstance.getID();
task.terminate(tkiid);
```

The task instance is terminated immediately without waiting for any outstanding tasks.

### Deleting task instances:

Task instances are only automatically deleted when they complete if this is specified in the associated task template from which the instances are derived. This example shows how to delete all of the task instances that are finished and are not automatically deleted.

1. List the task instances that are finished.

```
QueryResultSet result =
 task.query("DISTINCT TASK.TKIID",
 "TASK.STATE = TASK.STATE.STATE_FINISHED",
 null, null, null);
```

This action returns a query result set that lists task instances that are finished.

2. Delete the task instances that are finished.

```
while (result.next())
{
 TKIID tkiid = (TKIID) result.getOID(1);
 task.delete(tkiid);
}
```

### Releasing a claimed task:

When a potential owner claims a task, this person is responsible for completing the task. However, sometimes the claimed task must be released so that another potential owner can claim it.

Sometimes it is necessary for someone with administrator rights to release a claimed task. This situation might occur, for example, when a task must be completed but the owner of the task is absent. The owner of the task can also release a claimed task.

1. List the claimed tasks owned by a specific person, for example, Smith.

```
QueryResultSet result =
 task.query("DISTINCT TASK.TKIID",
 "TASK.STATE = TASK.STATE.STATE_CLAIMED AND
 TASK.OWNER = 'Smith'",
 null, null, null);
```

This action returns a query result set that lists the tasks claimed by the specified person, Smith.

2. Release the claimed task.

```
if (result.size() > 0)
{
 result.first();
 TKIID tkiid = (TKIID) result.getOID(1);
 task.cancelClaim(tkiid, true);
}
```

This action returns the task to the ready state so that it can be claimed by one of the other potential owners. Any output or fault data that is set by the original owner is kept.

### Managing work items:

During the lifetime of an activity instance or a task instance, the set of people associated with the object can change, for example, because a person is on vacation, 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.

A work item represents the assignment of an object to a user or group of users for a particular reason. The object is typically a staff activity instance, a process instance, or a human task. The reasons are derived from the role that the user has for an activity or task. An activity or task can have multiple work items because a user can have different roles in association with the activity or task, and a work item is created for each of these roles.

The actions that can be taken to manage work items depend on the role that the user has, for example, an administrator can create, delete and transfer work items, but the task owner can transfer work items only.

- Create a work item.

```
// query the task instance for which an additional
// administrator is to be specified
QueryResultSet result = task.query("TASK.TKIID",
 "TASK.NAME='CustomerOrder'",
 null, null, null);

if (result.size() > 0)
{
 result.first();
 // create the work item
 task.createWorkItem((TKIID)(result.getOID(1)),
 WorkItem.REASON_ADMINISTRATOR,"Smith");
}
```

This action creates a work item for the user Smith who has the administrator role.

- Delete a work item.

```
// query the task instance for which a work item is to be deleted
QueryResultSet result = task.query("TASK.TKIID",
 "TASK.NAME='CustomerOrder'",
 null, null, null);

if (result.size() > 0)
{
 result.first();
 // delete the work item
 task.deleteWorkItem((TKIID)(result.getOID(1)),
 WorkItem.REASON_READER,"Smith");
}
```

This action deletes the work item for the user Smith who has the reader role.

- Transfer a work item.

```
// query the task that is to be rescheduled
QueryResultSet result =
 task.query("DISTINCT TASK.TKIID",
 "TASK.NAME='CustomerOrder' AND
 TASK.STATE=TASK.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 task
 task.transferWorkItem((TKIID)(result.getOID(1)),
 WorkItem.REASON_POTENTIAL_OWNER,"Miller","Smith");
}
```

This action transfers the work item to the user Smith so that he can work on it.

### Creating task templates and task instances at runtime:

You usually use a modeling tool, such as WebSphere Integration Developer to build task templates. You then install the task templates in WebSphere Process Server and create instances from these templates, for example, using Business Process Choreographer Explorer. However, you can also create human or participating task instances or templates at runtime. You might want to do this, for example, when the task definition is not available when the application is deployed, the tasks that are part of a workflow are not yet known, or you need a task to cover some ad-hoc collaboration between a group of people.

1. **Optional:** If your interfaces contain types that are not simple Java types, create or identify an application that contains the data types that are used by the runtime task or template.

The runtime task or task template runs in the context of the application and gets access to the data types. Ensure that your application also contains a task or process definition so that the application is loaded by Business Process Choreographer. These tasks or processes can be dummy tasks or processes.

2. Create a task model.

The model refers to the data types in the application identified in step 1.

3. Validate the task model.

4. Create the task template or the task instance.

Use the `HumanTaskManagerService` interface to complete this action. If your interfaces contain types other than simple Java types, specify the name of the application that contains the data type definitions when you create your task instance or template.

#### *Creating runtime tasks that use simple Java types:*

This example creates a runtime task that uses only simple Java types in its interface, for example, a `String` object.

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

1. Access the `ClientTaskFactory` and create a resource set to contain the definitions of the new task model.

```

ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
2. Create the WSDL definition and add the descriptions of your operations.
// create the WSDL interface
Definition definition = factory.createWSDLDefinition
 (resourceSet, new QName("http://www.ibm.com/task/test/", "test"));

// create a port type
PortType portType = factory.createPortType(definition, "doItPT");

// create an operation; the input and output messages are of type String:
// a fault message is not specified
Operation operation = factory.createOperation
 (definition, portType, "doIt",
 new QName("http://www.w3.org/2001/XMLSchema", "string"),
 new QName("http://www.w3.org/2001/XMLSchema", "string"),
 null);

```

3. Create the EMF model of your new human task.

If you are creating a task instance, a valid-from date (UTCDate) is not required.

```

TTask humanTask = factory.createTTask(resourceSet,
 TTaskKinds.HTASK_LITERAL,
 "TestTask",
 new UTCDate("2005-01-01T00:00:00"),
 "http://www.ibm.com/task/test/",
 portType,
 operation);

```

This step initializes the properties of the task model with default values.

4. Modify the properties of your human task model.

```

// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance(TBoolean, YES_LITERAL);

// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();

// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
 taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);

```

5. Create the task model that contains all the resource definitions.

```

TaskModel taskModel = ClientTaskFactory.createTaskModel(resourceSet);

```

6. Validate the task model and correct any validation problems that are found.

```

ValidationProblem[] validationProblems = taskModel.validate();

```

7. Create the runtime task instance or template.

Use the HumanTaskManagerService interface to create the task instance or the task template. Because the application uses simple Java types only, you do not need to specify an application name.

- The following snippet creates a task instance:  

```
task.createTask(taskModel, null, "HTM");
```
- The following snippet creates a task template:  

```
task.createTaskTemplate(taskModel, null);
```

If a runtime task instance is created, it can now be started. If a runtime task template is created, you can now create task instances from the template.

*Creating runtime tasks that use complex types:*

This example creates a runtime task that uses complex types in its interface. The complex types are already defined, that is, the local file system on the client has XSD files that contain the description of the complex types.

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

1. Access the ClientTaskFactory and create a resource set to contain the definitions of the new task model.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Add the XSD definitions of your complex types to the resource set so that they are available when you define your operations.

The files are located relative to the location where the code is executed.

```
factory.loadXSDSchema(resourceSet, "InputBO.xsd");
factory.loadXSDSchema(resourceSet, "OutputBO.xsd");
```

3. Create the WSDL definition and add the descriptions of your operations.

```
// create the WSDL interface
Definition definition = factory.createWSDLDefinition
 (resourceSet, new QName("http://www.ibm.com/task/test/", "test"));
```

```
// create a port type
PortType portType = factory.createPortType(definition, "doItPT");
```

```
// create an operation; the input message is an InputBO and
// the output message an OutputBO;
// a fault message is not specified
Operation operation = factory.createOperation
 (definition, portType, "doIt",
 new QName("http://Input", "InputBO"),
 new QName("http://Output", "OutputBO"),
 null);
```

4. Create the EMF model of your new human task.

If you are creating a task instance, a valid-from date (UTCDate) is not required.

```
TTask humanTask = factory.createTTask(resourceSet,
 TTaskKinds.HTASK_LITERAL,
 "TestTask",
 new UTCDate("2005-01-01T00:00:00"),
 "http://www.ibm.com/task/test/",
 portType,
 operation);
```

This step initializes the properties of the task model with default values.

5. Modify the properties of your human task model.

```
// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance(TBoolean, YES_LITERAL);
```

```
// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();
```

```
// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");
```

```
// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
```

```

 taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

```

```

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);

```

6. Create the task model that contains all the resource definitions.

```

TaskModel taskModel = ClientTaskFactory.createTaskModel(resourceSet);

```

7. Validate the task model and correct any validation problems that are found.

```

ValidationProblem[] validationProblems = taskModel.validate();

```

8. Create the runtime task instance or template.

Use the `HumanTaskManagerService` interface to create the task instance or the task template. You must provide an application name that contains the data type definitions so that they can be accessed. The application must also contain a dummy task or process so that the application is loaded by Business Process Choreographer.

- The following snippet creates a task instance:

```

task.createTask(taskModel, "B0application", "HTM");

```

- The following snippet creates a task template:

```

task.createTaskTemplate(taskModel, "B0application");

```

If a runtime task instance is created, it can now be started. If a runtime task template is created, you can now create task instances from the template.

*Creating runtime tasks that use an existing interface:*

This example creates a runtime task that uses an interface that is already defined, that is, the local file system on the client has a file that contains the description of the interface.

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

1. Access the `ClientTaskFactory` and create a resource set to contain the definitions of the new task model.

```

ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();

```

2. Access the WSDL definition and the descriptions of your operations.

The interface description is located relative to the location where the code is executed.

```

Definition definition = factory.loadWSDLDefinition(
 resourceSet, "interface.wsdl");
PortType portType = definition.getPortType(
 new QName(definition.getTargetNamespace(), "doItPT"));
Operation operation = portType.getOperation("doIt", null, null);

```

3. Create the EMF model of your new human task.

If you are creating a task instance, a valid-from date (`UTCDate`) is not required.

```

TTask humanTask = factory.createTTask(resourceSet,
 TTaskKinds.HTASK_LITERAL,
 "TestTask",
 new UTCDate("2005-01-01T00:00:00"),
 "http://www.ibm.com/task/test/",
 portType,
 operation);

```

This step initializes the properties of the task model with default values.



4. Modify the properties of your human task model.

```
// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance(TBoolean, YES_LITERAL);

// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();

// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
 taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Create the task model that contains all the resource definitions.

```
TaskModel taskModel = ClientTaskFactory.createTaskModel(resourceSet);
```

6. Validate the task model and correct any validation problems that are found.

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. Create the runtime task instance or template.

Use the `HumanTaskManagerService` interface to create the task instance or the task template. You must provide an application name that contains the data type definitions so that they can be accessed. The application must also contain a dummy task or process so that the application is loaded by Business Process Choreographer.

- The following snippet creates a task instance:

```
task.createTask(taskModel, "B0application", "HTM");
```

- The following snippet creates a task template:

```
task.createTaskTemplate(taskModel, "B0application");
```

If a runtime task instance is created, it can now be started. If a runtime task template is created, you can now create task instances from the template.

*Creating runtime tasks that use an interface from the calling application:*

This example creates a runtime task that uses an interface that is part of the calling application. For example, the runtime task is created in a Java snippet of a business process and uses an interface from the process application.

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

1. Access the `ClientTaskFactory` and create a resource set to contain the definitions of the new task model.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
```

```
// specify the context class loader so that following resources are found
ResourceSet resourceSet = factory.createResourceSet
 (Thread.currentThread().getContextClassLoader());
```

2. Access the WSDL definition and the descriptions of your operations.

Specify the path within the containing package JAR file.

```

Definition definition = factory.loadWSDLDefinition(resourceSet,
 "com/ibm/workflow/metaflow/interface.wsdl");
PortType portType = definition.getPortType(
 new QName(definition.getTargetNamespace(), "doItPT"));
Operation operation = portType.getOperation("doIt", null, null);

```

3. Create the EMF model of your new human task.

If you are creating a task instance, a valid-from date (UTCDate) is not required.

```

TTask humanTask = factory.createTTask(resourceSet,
 TTaskKinds.HTASK_LITERAL,
 "TestTask",
 new UTCDate("2005-01-01T00:00:00"),
 "http://www.ibm.com/task/test/",
 portType,
 operation);

```

This step initializes the properties of the task model with default values.

4. Modify the properties of your human task model.

```

// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance(TBoolean, YES_LITERAL);

// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();

// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
 taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);

```

5. Create the task model that contains all the resource definitions.

```
TaskModel taskModel = ClientTaskFactory.createTaskModel(resourceSet);
```

6. Validate the task model and correct any validation problems that are found.

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. Create the runtime task instance or template.

Use the HumanTaskManagerService interface to create the task instance or the task template. You must provide an application name that contains the data type definitions so that they can be accessed.

- The following snippet creates a task instance:  

```
task.createTask(taskModel, "WorkflowApplication", "HTM");
```
- The following snippet creates a task template:  

```
task.createTaskTemplate(taskModel, "WorkflowApplication");
```

If a runtime task instance is created, it can now be started. If a runtime task template is created, you can now create task instances from the template.

### Creating plug-ins to customize human task functionality:

Business Process Choreographer provides an event handling infrastructure for events that occur during the processing of human tasks. Plug-in points are also provided so that you can adapt the functionality to your needs. You can use the service provider interfaces (SPIs) to create customized plug-ins for handling events and the processing of staff queries.

You can create plug-ins for human task API events and escalation notification events. You can also create a plug-in that processes the results that are returned from staff resolution. For example, at peak periods you might want to add users to the result list to help balance the workload.

You can register your plug-ins on different levels, for all tasks on a global level, for the tasks in an application component, for all of the tasks associated with a task template, or for a single task instance.

#### *Creating API event handlers:*

An API event occurs when an API method manipulates a human task. Use the API event handler plug-in service provider interface (SPI) to create plug-ins to handle the task events sent by the API or the internal events that have equivalent API events.

Complete the following steps to create an API event handler.

1. Write a class that implements the `APIEventHandlerPlugin2` interface or extends the `APIEventHandler` implementation class. This class can invoke the methods of other classes.
  - If you use the `APIEventHandlerPlugin2` interface, you must implement all of the methods of the `APIEventHandlerPlugin2` interface and the `APIEventHandlerPlugin` interface.
  - If you extend the SPI implementation class, overwrite the methods that you need.

This class runs in the context of a Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB) application. Ensure that this class and its helper classes follow the EJB specification.

**Tip:** If you want to call the `HumanTaskManagerService` interface from this class, do not call a method that updates the task that produced the event. This action results in a database deadlock.

2. Assemble the plug-in class and its helper classes into a JAR file.

If the helper classes are used by several J2EE applications, you can package these classes in a separate JAR file that you register as a shared library.
3. Create a service provider configuration file for the plug-in in the `META-INF/services/` directory of your JAR file.

The configuration file provides the mechanism for identifying and loading the plug-in. This file conforms to the Java 2 service provider interface specification.

- a. Create a file with the name `com.ibm.task.spi.plugin_in_nameAPIEventHandlerPlugin`, where `plugin_in_name` is the name of the plug-in.

For example, if your plug-in is called `Customer` and it implements the `com.ibm.task.spi.APIEventHandlerPlugin` interface, the name of the configuration file is `com.ibm.task.spi.CustomerAPIEventHandlerPlugin`.

- b. In the first line of the file that is neither a comment line nor a blank line, specify the fully qualified name of the plug-in class that you created in step 1.

For example, if your plug-in class is called `MyAPIEventHandler` and it is in the `com.customer.plugins` package, then the first line of the configuration file must contain the following entry:

```
com.customer.plugins.MyAPIEventHandler.
```

You have an installable JAR file that contains a plug-in that handles API events and a service provider configuration file that can be used to load the plug-in.

**Tip:** You only have one `eventHandlerName` property available to register event handlers. If you want to use an API event handler and a notification event handler both plug-in implementations must have the same name, for example, `Customer` as the event handler name for the SPI implementations.

If you want to package both plug-in implementations into the same JAR file, create two files in the `META-INF/services/` directory, for example, `com.ibm.task.spi.CustomerNotificationEventHandlerPlugin` and `com.ibm.task.spi.CustomerAPIEventHandlerPlugin2`. You can also package both SPI implementations in different JAR files.

You can also implement both interfaces with one class. If you choose this method, you also need two entries in the `META-INF/services/` directory. Because class loading problems can occur if you implement both interfaces in a single class, do not package the class in two different JAR files.

You now need to install and register the plug-in so that it is available to the human task container at runtime. You can register API event handlers with a task instance, a task template, or an application component.

*API event handlers:*

API events occur when a human task is modified or it changes state. To handle these API events, the event handler is invoked directly before the task is modified (pre-event method) and just before the API call returns (post-event method).

If the pre-event method throws an `ApplicationVetoException` exception, the API action is not performed, the exception is returned to the API caller, and the transaction associated with the event is rolled back. If the pre-event method was triggered by an internal event and an `ApplicationVetoException` exception is thrown, the internal event, such as an automatic claim, is not performed but an exception is not returned to the client application. In this case, an information message is written to the `SystemOut.log` file. If the API method throws an exception during processing, the exception is caught and passed to the post-event method. The exception is passed again to the caller after the post-event method returns.

The following rules apply to pre-event methods:

- Pre-event methods receive the parameters of the associated API method or internal event.
- Pre-event methods can throw an `ApplicationVetoException` exception to prevent processing from continuing.

The following rules apply to post-event methods:

- Post-event methods receive the parameters that were supplied to the API call, and the return value. If an exception is thrown by the API method implementation, the post-event method also receives the exception.
- Post-event methods cannot modify return values.
- Post-event methods cannot throw exceptions; runtime exceptions are logged but they are ignored.

To implement API event handlers, you can use either the `APIEventHandlerPlugin2` interface, which extends the `APIEventHandlerPlugin` interface, or extend the default `com.ibm.task.spi.APIEventHandler` SPI implementation class. If your event handler inherits from the default implementation class, it always implements the most recent version of the SPI. If you upgrade to a newer version of Business Process Choreographer, fewer changes are necessary if you want to exploit new SPI methods.

If you have both a notification event handler and an API event handler, both of these handlers must have the same name because you can register only one event handler name.

#### *Creating notification event handlers:*

Notification events are produced when human tasks are escalated. Business Process Choreographer provides functionality for handling escalations, such as creating escalation work items or sending e-mails. You can create notification event handlers to customize the way in which escalations are handled.

To implement notification event handlers, you can use either the `NotificationEventHandlerPlugin` interface, or you can extend the default `com.ibm.task.spi.NotificationEventHandler` service provider interface (SPI) implementation class.

Complete the following steps to create a notification event handler.

1. Write a class that implements the `NotificationEventHandlerPlugin` interface or extends the `NotificationEventHandler` implementation class. This class can invoke the methods of other classes.

If you use the `NotificationEventHandlerPlugin` interface, you must implement all of the interface methods. If you extend the SPI implementation class, overwrite the methods that you need.

This class runs in the context of a Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB) application. Ensure that this class and its helper classes follow the EJB specification.

The plug-in is invoked with the authority of the `EscalationUser` role. This role is defined when the human task container is configured.

**Tip:** If you want to call the `HumanTaskManagerService` interface from this class, do not call a method that updates the task or the escalation that produced the event. This action results in a database deadlock.

2. Assemble the plug-in class and its helper classes into a JAR file.

If the helper classes are used by several J2EE applications, you can package these classes in a separate JAR file that you register as a shared library.

3. Create a service provider configuration file for the plug-in in the `META-INF/services/` directory of your JAR file.

The configuration file provides the mechanism for identifying and loading the plug-in. This file conforms to the Java 2 service provider interface specification.

- a. Create a file with the name `com.ibm.task.spi.plugin_nameNotificationEventHandlerPlugin`, where `plugin_name` is the name of the plug-in.

For example, if your plug-in is called `HelpDeskRequest` (event handler name) and it implements the `com.ibm.task.spi.NotificationEventHandlerPlugin` interface, the name of the configuration file is `com.ibm.task.spi.HelpDeskRequestNotificationEventHandlerPlugin`.

- b. In the first line of the file that is neither a comment line nor a blank line, specify the fully qualified name of the plug-in class that you created in step 1.

For example, if your plug-in class is called `MyEventHandler` and it is in the `com.customer.plugins` package, then the first line of the configuration file must contain the following entry: `com.customer.plugins.MyEventHandler`.

You have an installable JAR file that contains a plug-in that handles notification events and a service provider configuration file that can be used to load the plug-in. You can register API event handlers with a task instance, a task template, or an application component.

**Tip:** You only have one `eventHandlerName` property available to register event handlers. If you want to use an API event handler and a notification event handler both plug-in implementations must have the same name, for example, `Customer` as the event handler name for the SPI implementations.

If you want to package both plug-in implementations into the same JAR file, create two files in the `META-INF/services/` directory, for example, `com.ibm.task.spi.CustomerNotificationEventHandlerPlugin` and `com.ibm.task.spi.CustomerAPIEventHandlerPlugin2`. You can also package both SPI implementations in different JAR files.

You can also implement both interfaces with one class. If you choose this method, you also need two entries in the `META-INF/services/` directory. Because class loading problems can occur if you implement both interfaces in a single class, do not package the class in two different JAR files.

You now need to install and register the plug-in so that it is available to the human task container at runtime. You can register notification event handlers with a task instance, a task template, or an application component.

*Creating plug-ins to post-process staff query results:*

Staff resolution returns a list of the users that are assigned to a specific role, for example, potential owner of a task. You can create a plug-in to change the results of staff queries returned by staff resolution. For example, to improve workload balancing, you might have a plug-in that removes users from the query result who already have a high workload.

You can have only one post-processing plug-in; this means that the plug-in must handle the staff results from all tasks. Your plug-in can add or remove users, or change user or group information. It can also change the result type, for example, from a list of users to a group, or to everybody.

Because the plug-in runs after staff resolution completes, any rules that you have to preserve confidentiality or security have already been applied. The plug-in receives information about users that have been removed during staff resolution (in the `HTM_REMOVED_USERS` map key). You must ensure that your plug-in uses this context information to preserve any confidentiality or security rules you might have.

To implement post-processing of staff query results, you use the `StaffQueryResultPostProcessingPlugin` interface. The interface has methods for modifying the query results for tasks, escalations, task templates, and application components.

Complete the following steps to create a plug-in to post-process staff query results.

1. Write a class that implements the `StaffQueryResultPostProcessingPlugin` interface.

You must implement all of the interface methods. This class can invoke methods of other classes.

This class runs in the context of a Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB) application. Ensure that this class and its helper classes follow the EJB specification.

**Tip:** If you want to call the `HumanTaskManagerService` interface from this class, do not call a method that updates the task that produced the event. This action results in a database deadlock.

The following example shows how you might change the editor role of a task called `SpecialTask`.

```
public StaffQueryResult processStaffQueryResult
 (StaffQueryResult originalStaffQueryResult,
 Task task,
 int role,
 Map context)
{
 StaffQueryResult newStaffQueryResult = originalStaffQueryResult;
 StaffQueryResultFactory staffResultFactory =
 StaffQueryResultFactory.newInstance();
 if (role == com.ibm.task.api.WorkItem.REASON_EDITOR &&
 task.getName() != null &&
 task.getName().equals("SpecialTask"))
 {
 UserData user = staffResultFactory.newUserData
 ("SuperEditor",
 new Locale("en-US"),
 "SuperEditor@company.com");
 ArrayList userList = new ArrayList();
 userList.add(user);

 newStaffQueryResult = staffResultFactory.newStaffQueryResult(userList);
 }
 return(newStaffQueryResult);
}
```

2. Assemble the plug-in class and its helper classes into a JAR file.

If the helper classes are used by several J2EE applications, you can package these classes in a separate JAR file that you register as a shared library.

3. Create a service provider configuration file for the plug-in in the `META-INF/services/` directory of your JAR file.

The configuration file provides the mechanism for identifying and loading the plug-in. This file conforms to the Java 2 service provider interface specification.

- a. Create a file with the name `com.ibm.task.spi.plug-in_nameStaffQueryResultPostProcessingPlugin`, where *plug-in\_name* is the name of the plug-in.

For example, if your plug-in is called `MyHandler` and it implements the `com.ibm.task.spi.StaffQueryResultPostProcessingPlugin` interface, the name of the configuration file is `com.ibm.task.spi.MyHandlerStaffQueryResultPostProcessingPlugin`.

- b. In the first line of the file that is neither a comment line nor a blank line, specify the fully qualified name of the plug-in class that you created in step 1.

For example, if your plug-in class is called `StaffPostProcessor` and it is in the `com.customer.plugins` package, then the first line of the configuration



file must contain the following entry:

`com.customer.plugins.StaffPostProcessor`. You have an installable JAR file that contains a plug-in that handles notification events and a service provider configuration file that can be used to load the plug-in.

4. Install the plug-in.

You can have only one post-processing plug-in for staff query results. You must install the plug-in as a shared library.

5. Register the plug-in.

- a. In the administrative console, go to the Custom Properties page of the human task container (**Application servers** → *server\_name* → **Human task container** → **Custom properties**).
- b. Add a custom property with the name **Staff.PostProcessingPlugin**, and a value of the name that you gave to your plug-in, `MyHandler` in this example.

*Installing plug-ins:*

To use a plug-in, you must install the plug-in so that it can be accessed by the task container.

The way in which you install the plug-in depends on whether the plug-in is to be used by only one Java 2 Enterprise Edition (J2EE) application, or several applications.

Complete one of the following steps to install a plug-in.

- Install a plug-in for use by a single J2EE application.

Add your plug-in JAR file to the application EAR file. In the deployment descriptor editor in WebSphere Integration Developer, install the JAR file for your plug-in as a project utility JAR file for the J2EE application of the main enterprise JavaBeans (EJB) module.

- Install a plug-in for use by several J2EE applications.

Put the JAR file in a WebSphere Application Server shared library and associate the library with the applications that need access to the plug-in. To make the JAR file available in a network deployment environment, distribute the JAR file on each server manually, and then install the shared library once for each cell.

You can now register the plug-in.

*Registering plug-ins:*

You can register your plug-ins on different levels in the task container artifact hierarchy. For example, for all tasks on a global level, for the tasks of an application component, for all of the tasks associated with a task template, or for a single task instance.

When you register multiple plug-ins, scoping is supported. This means that a plug-in that is registered on a lower level of the task container artifact hierarchy, such as a task instance, is used instead of the plug-in that is registered on a higher level, such as a task template or application component. Scoping is supported for all of the hierarchy levels. The task container uses the plug-in that is registered on the lowest level of the hierarchy.

You can register a plug-in in one of the following ways.

- Register the plug-in in the task model.



In the task editor in WebSphere Integration Developer in the Details page of the properties area for the task, specify the name of the event handler in the **Event handler name** field.

- Register the plug-in for ad-hoc tasks or task templates that you create at runtime.  
Use the `setEventHandlerName` method of the `TTask` class to register the name of the event handler.
- Change the registered event handler for a task instance at runtime.  
Use the `update(Task task)` method to use a different event handler for a task instance at runtime. The caller must have task administrator authority to update this property.
- Register the plug-in on a global level.  
In the administration console on the Custom properties page for the human task container, define a custom property for the plug-in. The value of the custom property is the plug-in name.

### HumanTaskManagerService interface:

The `HumanTaskManagerService` interface exposes task-related functions that can be called by a local or a remote client.

The methods that can be called depend on the state of the task and the authorization of the person that uses the application containing the method. The main methods for manipulating task objects are listed here. For more information about these methods and the other methods that are available in the `HumanTaskManagerService` interface, see the Javadoc in the `com.ibm.task.api` package.

### Task templates

The following methods are available to work with task templates.

Table 26. API methods for task templates

| Method                          | Description                                                                                                               |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <code>getTaskTemplate</code>    | Retrieves the specified task template.                                                                                    |
| <code>createAndCallTask</code>  | Creates and runs a task instance from the specified task template and waits synchronously for the result.                 |
| <code>createAndStartTask</code> | Creates and starts a task instance from the specified task template.                                                      |
| <code>createTask</code>         | Creates a task instance from the specified task template.                                                                 |
| <code>createInputMessage</code> | Creates an input message for the specified task template. For example, create a message that can be used to start a task. |
| <code>queryTaskTemplates</code> | Retrieves task templates that are stored in the database.                                                                 |

## Task instances

The following methods are available to work with task instances.

*Table 27. API methods for task instances*

| Method           | Description                                                                                                                     |
|------------------|---------------------------------------------------------------------------------------------------------------------------------|
| getTask          | Retrieves a task instance; the task instance can be in any state.                                                               |
| callTask         | Starts an originating task synchronously.                                                                                       |
| startTask        | Starts a task that has already been created.                                                                                    |
| suspend          | Suspends the human or participating task.                                                                                       |
| resume           | Resumes the human or participating task.                                                                                        |
| terminate        | Terminates the specified task instance. If an originating task is terminated, this action has no impact on the invoked service. |
| delete           | Deletes the specified task instance.                                                                                            |
| claim            | Claims the task for processing.                                                                                                 |
| update           | Updates the task instance.                                                                                                      |
| complete         | Completes the task instance.                                                                                                    |
| cancelClaim      | Releases a claimed task instance so that it can be worked on by another potential owner.                                        |
| createWorkItem   | Creates a work item for the task instance.                                                                                      |
| transferWorkItem | Transfers the work item to a specified owner.                                                                                   |
| deleteWorkItem   | Deletes the work item.                                                                                                          |

## Escalations

The following methods are available to work with escalations.

*Table 28. API methods for working with escalations*

| Method        | Description                                  |
|---------------|----------------------------------------------|
| getEscalation | Retrieves the specified escalation instance. |

## Custom properties

Tasks, task templates, and escalations can all have custom properties. The interface provides a get and a set method to retrieve and set values for custom properties. You can also associate named properties with, and retrieve named properties from task instances. Custom property names and values must be of the `java.lang.String` type. The following methods are valid for tasks, task templates, and escalations.

*Table 29. API methods for variables and custom properties*

| Method              | Description                                                         |
|---------------------|---------------------------------------------------------------------|
| getCustomProperty   | Retrieves the named custom property of the specified task instance. |
| getCustomProperties | Retrieves the custom properties of the specified task instance.     |

Table 29. API methods for variables and custom properties (continued)

| Method                 | Description                                                         |
|------------------------|---------------------------------------------------------------------|
| getCustomPropertyNames | Retrieves the names of the custom properties for the task instance. |
| setCustomProperty      | Stores custom-specific values for the specified task instance.      |

Allowed actions for tasks:

The actions that can be carried out on a task depend on whether the task is a participating task, a purely human task, an originating task, or an administrative task.

You cannot use all of the actions provided by the LocalHumanTaskManager or the HumanTaskManager interface for all kinds of tasks. The following table shows the actions that you can carry out on each kind of task.

| Action                                | Kind of task       |                |                  |                     |
|---------------------------------------|--------------------|----------------|------------------|---------------------|
|                                       | Participating task | Human task     | Originating task | Administrative task |
| callTask                              |                    |                | X <sup>1</sup>   |                     |
| cancelClaim                           | X                  | X <sup>1</sup> |                  |                     |
| claim                                 | X                  | X <sup>1</sup> |                  |                     |
| complete                              | X                  | X <sup>1</sup> |                  | X                   |
| completeWithFollowOnTask <sup>4</sup> | X                  | X <sup>1</sup> |                  |                     |
| completeWithFollowOnTask <sup>5</sup> |                    | X <sup>3</sup> | X <sup>3</sup>   |                     |
| createFaultMessage                    | X                  | X              | X                | X                   |
| createInputMessage                    | X                  | X              | X                | X                   |
| createOutputMessage                   | X                  | X              | X                | X                   |
| createWorkItem                        | X                  | X <sup>1</sup> | X                | X                   |
| delete                                | X <sup>1</sup>     | X <sup>1</sup> | X                | X <sup>1</sup>      |
| deleteWorkItem                        | X                  | X <sup>1</sup> | X                | X                   |
| getCustomProperty                     | X                  | X <sup>1</sup> | X                | X                   |
| getDocumentation                      | X                  | X <sup>1</sup> | X                | X                   |
| getFaultNames                         | X                  | X <sup>1</sup> |                  |                     |
| getFaultMessage                       | X                  | X <sup>1</sup> | X                |                     |
| getInputMessage                       | X                  | X <sup>1</sup> | X                |                     |
| getOutputMessage                      | X                  | X <sup>1</sup> | X                |                     |
| getRoleInfo                           | X                  | X <sup>1</sup> | X                | X                   |
| getTask                               | X                  | X <sup>1</sup> | X                | X                   |
| getUISettings                         | X                  | X <sup>1</sup> | X                | X                   |
| resume                                | X                  | X <sup>1</sup> |                  |                     |
| setCustomProperty                     | X                  | X <sup>1</sup> | X                | X                   |
| setFaultMessage                       | X                  | X <sup>1</sup> |                  |                     |
| setOutputMessage                      | X                  | X <sup>1</sup> |                  |                     |
| startTask                             | X <sup>1</sup>     | X <sup>1</sup> | X                | X                   |

| Action                          | Kind of task       |                |                  |                     |
|---------------------------------|--------------------|----------------|------------------|---------------------|
|                                 | Participating task | Human task     | Originating task | Administrative task |
| startTaskAsSubtask <sup>6</sup> | X                  | X <sup>1</sup> |                  |                     |
| startTaskAsSubtask <sup>7</sup> |                    | X <sup>3</sup> | X <sup>3</sup>   |                     |
| suspend                         | X                  | X <sup>1</sup> |                  |                     |
| suspendWithCancelClaim          | X                  | X <sup>1</sup> |                  |                     |
| terminate                       | X <sup>1</sup>     | X <sup>1</sup> | X <sup>1</sup>   |                     |
| transferWorkItem                | X                  | X <sup>1</sup> | X                | X                   |
| updateInactiveTask              | X <sup>2</sup>     | X <sup>3</sup> | X <sup>2</sup>   | X <sup>2</sup>      |
| updateTask                      | X                  | X <sup>1</sup> | X                | X                   |

**Notes:**

1. For stand-alone tasks, ad-hoc tasks, and task templates only
2. For stand-alone tasks, inline tasks in business processes, and ad-hoc tasks only
3. For stand-alone tasks and ad-hoc tasks only
4. The tasks kinds that can have follow-on tasks
5. The task kinds that can be used as follow-on tasks
6. The tasks kinds that can have subtasks
7. The task kinds that can be used as subtasks

## Handling exceptions and faults

A BPEL process might encounter a fault at different points in the process.

Business Process Execution Language (BPEL) faults originate from:

- Web service invocations (Web Services Description Language (WSDL) faults)
- Throw activities
- BPEL standard faults that are recognized by Business Process Choreographer

Mechanisms exist to handle these faults. Use one of the following mechanisms to handle faults that are generated by a process instance:

- Pass control to the corresponding fault handlers
- Compensate previous work in the process
- Stop the process and let someone repair the situation (force-retry, force-complete)

A BPEL process can also return faults to a caller of an operation provided by the process. You can model the fault in the process as a reply activity with a fault name and fault data. These faults are returned to the API caller as checked exceptions.

If a BPEL process does not handle a BPEL fault or if an API exception occurs, a runtime exception is returned to the API caller. An example for an API exception is 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:

If a method in the BusinessFlowManagerService interface or the HumanTaskManagerService 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` or `TaskException`. It is a *best practice* to catch generic exceptions with a final `catch(ProcessException)` or `catch(TaskException)` statement. This statement helps to ensure the upward compatibility of your application program because it takes account of all of the other exceptions that can occur.

#### Checking which fault is set for an activity:

1. List the task 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 activities.

2. Read the name of the fault.

```
if (result.size() > 0)
{
 result.first();
 AIID aaid = (AIID) result.getOID(1);
 ClientObjectWrapper faultMessage = process.getFaultMessage(aaid);
 DataObject fault = null ;
 if (faultMessage.getObject() != null && faultMessage.getObject()
 instanceof DataObject)
 {
 fault = (DataObject) faultMessage.getObject();
 Type type = fault.getType();
 String name = type.getName();
 String uri = type.getURI();
 }
}
```

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:

If an activity causes a fault to occur, the fault type determines the actions that you can take to repair the activity.

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.

```
if (result.size() > 0)
{
 result.first();
 AIID aaid = (AIID) result.getOID(1);
 ActivityInstanceData activity = process.getActivityInstance(aaid);

 ProcessException excp = activity.getUnhandledException();
```

```

 if (excp instanceof ApplicationFaultException)
 {
 ApplicationFaultException fault = (ApplicationFaultException)excp;
 String faultName = fault.getFaultName();
 }
}

```

## Developing Web service API client applications

You can develop client applications that access business process applications and human task applications through Web services APIs.

Client applications can be developed in any Web service client environment, including Java Web services and Microsoft .NET.

### Introduction: Web services

Web services are Web-based enterprise applications that use open, XML-based standards and transport protocols to exchange data with client applications. Web services allow the use of a language- and environment-neutral programming model.

Web services use the following core technologies:

- XML (Extensible Markup Language). XML solves the problem of data independence. You use it to describe data, and also to map that data into and out of any application or programming language
- WSDL (Web Services Description Language). You use this XML-based language to create a description of an underlying application. It is this description that turns an application into a Web service, by acting as the interface between the underlying application and other Web-enabled applications.
- SOAP (Simple Object Access Protocol). SOAP is the core communications protocol for the Web, and most Web services use this protocol to talk to each other.

### Web service components and sequence of control

A number of client-side and server-side components participate in the sequence of control that represents a Web service request and response.

A typical sequence of control is as follows.

1. On the client side:
  - a. A client application (provided by the user) issues a request for a Web service.
  - b. A proxy client (also provided by the user, but which can be automatically generated using client-side utilities) wraps the service request in a SOAP request envelope.
  - c. The client-side development infrastructure forwards the request to a URL defined as the Web service's endpoint.
2. The network transmits the request to the Web service endpoint using HTTP or HTTPS.
3. On the server side:
  - a. The generic Web services API receives and decodes the request.
  - b. The request is either handled directly by the generic Business Flow Manager or Human Task Manager component, or forwarded to the specified business process or human task.
  - c. The returned data is wrapped in a SOAP response envelope.

4. The network transmits the response to the client-side environment using HTTP or HTTPS.
5. Back on the client side:
  - a. The client-side development infrastructure unwraps the SOAP response envelope.
  - b. The proxy client extracts the data from the SOAP response and passes it to the client application.
  - c. The client application processes the returned data as necessary.

### **Overview of the Web services APIs**

Web services APIs allow you to develop client applications that use Web services to access business processes and human tasks running in the Business Process Choreographer environment.

The Business Process Choreographer Web services API provides two separate Web service interfaces (WSDL port types):

- The Business Flow Manager API. Allows client applications to interact with microflows and long-running processes, for example:
  - Create process templates and process instances
  - Claim existing processes
  - Query a process by its ID

Refer to “Developing applications for business processes” on page 196 for a complete list of possible actions.

- The Human Task Manager API. Allows client applications to:
  - Create and start tasks
  - Claim existing tasks
  - Complete tasks
  - Query a task by its ID
  - Query a collection of tasks.

Refer to “Developing applications for human tasks” on page 216 for a complete list of possible actions.

Client applications can use either or both of the Web service interfaces.

### **Example**

The following is a possible outline for a client application that accesses the Human Task Manager Web services API to process a participating human task:

1. The client application issues a `query(...)` Web service call to the WebSphere Process Server requesting a list of participating tasks to be worked on by a user.
2. The list of participating tasks is returned in a SOAP/HTTP response envelope.
3. The client application then issues a `claim(...)` Web service call to claim one of the participating tasks.
4. The WebSphere Process Server returns the task’s input message.
5. The client application issues a `complete(...)` Web service call to complete the task with an output or fault message.

## Requirements for business processes and human tasks

Business processes and human tasks developed with the WebSphere Integration Developer to run on the Business Process Choreographer must conform to specific rules to be accessible through the Web services APIs.

The requirements are:

1. The interfaces of business processes and human tasks must be defined using the "document/literal wrapped" style defined in the Java API for XML-based RPC (JAX-RPC 1.1) specification. This is the default style for all business processes and human tasks developed with the WID.
2. Fault messages exposed by business processes and human tasks for Web service operations must comprise a single WSDL message part defined with an XML Schema element. For example:

```
<wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
```

### Related information

Java API for XML based RPC (JAX-RPC) downloads page

Which style of WSDL should I use?

## Developing client applications

The client application development process consists of a number of steps.

1. Decide which Web services API your client application needs to use: the Business Flow Manager API, Human Task Manager API, or both.
2. Export the necessary files from the WebSphere Process Server environment. Alternatively, you can copy the files from the WebSphere Process Server client CD.
3. In your chosen client application development environment, generate a *proxy client* using the exported artifacts.
4. **Optional:** Generate *helper classes*. Helper classes are required if your client application interacts directly with concrete processes or tasks on the WebSphere server. They are not, however, necessary if your client application is only going to perform generic tasks such as issuing queries.
5. Develop the code for your client application.
6. Add any necessary security mechanisms to your client application.

## Copying artifacts

A number of artifacts must be copied from the WebSphere environment to help in the creation of client applications.

There are two ways to obtain these artifacts:

- Publish and export them from the WebSphere Process Server environment.
- Copy files from the WebSphere Process Server client CD.

### Publishing and exporting artifacts from the server environment:

Before you can develop client applications to access the Web services APIs, you must publish and export a number of artifacts from the WebSphere server environment.

The artifacts to be exported are:

- Web Service Definition Language (WSDL) files describing the port types and operations that make up the Web services APIs.



- XML Schema Definition (XSD) files containing definitions of data types referenced by services and methods in the WSDL files.
- Additional WSDL and XSD files describing business objects. Business objects describe concrete business processes or human tasks running on the WebSphere server. These additional files are only required if your client application needs to interact directly with the concrete business processes or human tasks through the Web services APIs. They are not necessary if your client application is only going to perform generic tasks, such as issuing queries.

After these artifacts are published, you need to copy them to your client programming environment, where they are used to generate a proxy client and helper classes.

*Specifying the Web service endpoint address:*

The Web service endpoint address is the URL that a client application must specify to access the Web services APIs. The endpoint address is written into the WSDL file that you export to generate a proxy client for your client application.

The Web service endpoint address to use depends on your WebSphere server configuration:

- Scenario 1. A single WebSphere server. The WebSphere endpoint address to specify is the host name and port number of the server, for example **host1:9080**.
- Scenario 2. A WebSphere cluster composed of several servers. The WebSphere endpoint address to specify is the host name and port of the server that is hosting the Web services APIs, for example, **host2:9081**.
- Scenario 3. A Web server is used as a front end. The WebSphere endpoint address to specify is the host name and port of the Web server, for example: **host:80**.

By default, the Web service endpoint address takes the form *protocol://host:port/context\_root/fixed\_path*. Where:

- *protocol*. The communications protocol to be used between the client application and the WebSphere server. The default protocol is HTTP. You can instead choose to use the more secure HTTPS (HTTP over SSL) protocol. It is recommended to use HTTPS.
- *host:port*. The host name and port number used to access the machine that is hosting the Web services APIs. These values vary depending on the WebSphere server configuration; for example, whether your client application is to access the application directly or through a Web server front end.
- *context\_root*. You are free to choose any value for the context root. The value you choose must, however, be unique within each WebSphere cell. The default value uses a "node\_server/cluster" suffix that eliminates the risk of naming conflicts.
- *fixed\_path* is either `/sca/com/ibm/bpe/api/BFMWS` (for the Business Flow Manager API) or `/sca/com/ibm/task/api/HTMWS` (for the Human Task Manager API) and cannot be modified.

The Web service endpoint address is initially specified when configuring the business process container or human task container:

1. Log on to the administrative console with a user ID with administrator rights.
2. Choose **Applications** → **SCA modules**.

**Note:** You can also select **Applications** → **Enterprise applications** to display a list of all available enterprise applications.

3. Select **BPEContainer** (for the business process container) or **TaskContainer** (for the human task container) from the list of SCA modules or applications.
4. Choose **Provide HTTP endpoint URL information** from the list of **Additional properties**.
5. Select one of the default prefixes from the list, or enter a custom prefix. Use a prefix from the default prefix list if your client applications are to connect directly to the application server hosting the Web services API. Otherwise, specify a custom prefix.
6. Click **Apply** to copy the selected prefix to the SCA module.
7. Click **OK**. The URL information is saved to your workspace.

You can view the current value in the administrative console (for example, for the business process container: **Enterprise Applications** → **BPEContainer** → **View Deployment Descriptor**).

In the exported WSDL file, the `location` attribute of the `soap:address` element contains the specified Web services endpoint address. For example:

```
<wsdl:service name="BFMWSservice">
 <wsdl:port name="BFMWSport" binding="this:BFMWSBinding">
 <soap:address location=
 "https://myserver:9080/WebServicesAPIs/sca/com/ibm/bpe/api/BFMWS"/>
 </wsdl:port>
</wsdl:service>
```

#### **Related concepts**

“Adding security (Java Web services)” on page 256

You must secure Web service communications by implementing security mechanisms in your client application.

#### **Related tasks**

“Adding security (.NET)” on page 262

You can secure Web service communications by integrating security mechanisms into your client application.

#### *Publishing WSDL files:*

A Web Service Definition Language (WSDL) file contains a detailed description of all the operations available with a Web services API. Separate WSDL files are available for the Business Flow Manager and Human Task Manager Web services APIs. You must first publish these WSDL files then copy them from the WebSphere environment to your development environment, where they are used to generate a proxy client.

Before publishing the WSDL files, be sure to specify the correct Web services endpoint address. This is the URL that your client application uses to access the Web services APIs.

You only need to publish WSDL files once.

**Note:** If you have the WebSphere Process Server client CD, you can copy the files directly from there to your client programming environment instead.

#### *Publishing the business process WSDL:*

Use the administrative console to publish the WSDL file.

1. Log on to the administrative console with a user ID with administrator rights.

## 2. Select **Applications** → **SCA modules**

**Note:** You can also select **Applications** → **Enterprise applications** to display a list of all available enterprise applications.

3. Choose the **BPEContainer** application from the list of SCA modules or applications.
4. Select **Publish WSDL files** from the list of **Additional properties**
5. Click on the zip file in the list.
6. On the File Download window that appears, click **Save**.
7. Browse to a local folder and click **Save**.

The exported zip file is named BPEContainer\_WSDLFiles.zip. The zip file contains a WSDL file that describes the Web services, and any XSD files referenced from within the WSDL file.

### *Publishing the human task WSDL:*

Use the administrative console to publish the WSDL file.

1. Log on to the administrative console with a user ID with administrator rights.
2. Select **Applications** → **SCA modules**

**Note:** You can also select **Applications** → **Enterprise applications** to display a list of all available enterprise applications.

3. Choose the **TaskContainer** application from the list of SCA modules or applications.
4. Select **Publish WSDL files** from the list of **Additional properties**
5. Click on the zip file in the list.
6. On the File Download window that appears, click **Save**.
7. Browse to a local folder and click **Save**.

The exported zip file is named TaskContainer\_WSDLFiles.zip. The zip file contains a WSDL file that describes the Web services, and any XSD files referenced from within the WSDL file.

### *Exporting business objects:*

Business processes and human tasks have well-defined interfaces that allow them to be accessed externally as Web services. If these interfaces reference business objects, you need to export the interface definitions and business objects to your client programming environment.

This procedure must be repeated for each business object that your client application needs to interact with.

In WebSphere Process Server, business objects define the format of request, response and fault messages that interact with business processes or human tasks. These messages can also contain definitions of complex data types.

For example, to create and start a human task, the following items of information must be passed to the task interface:

- The task template name
- The task template namespace

- An input message, containing formatted business data
- A response wrapper for returning the response message
- A fault message for returning faults and exceptions

These items are encapsulated within a single business object. All operations of the Web service interface are modeled as a "document/literal wrapped" operation. Input and output parameters for these operations are encapsulated in wrapper documents. Other business objects define the corresponding response and fault message formats.

In order to create and start the business process or human task through a Web service, these wrapper objects must be made available to the client application on the client side.

This is achieved by exporting the business objects from the WebSphere environment as Web Service Definition Language (WSDL) and XML Schema Definition (XSD) files, importing the data type definitions into your client programming environment, then converting them to helper classes for use by the client application.

1. Launch the WebSphere Integration Developer (WID) Workspace if it is not already running.
2. Select the Library module containing the business objects to be exported. A Library module is a compressed file that contains the necessary business objects.
3. Export the Library module.
4. Copy the exported files to your client application development environment.

### Example

Assume a business process exposes the following Web service operation:

```
<wsdl:operation name="updateCustomer">
 <wsdl:input message="tns:updateCustomerRequestMsg"
 name="updateCustomerRequest"/>
 <wsdl:output message="tns:updateCustomerResponseMsg"
 name="updateCustomerResponse"/>
 <wsdl:fault message="tns:updateCustomerFaultMsg"
 name="updateCustomerFault"/>
</wsdl:operation>
```

with the WSDL messages defined as:

```
<wsdl:message name="updateCustomerRequestMsg">
 <wsdl:part element="types:updateCustomer"
 name="updateCustomerParameters"/>
</wsdl:message>
<wsdl:message name="updateCustomerResponseMsg">
 <wsdl:part element="types:updateCustomerResponse"
 name="updateCustomerResult"/>
</wsdl:message>
<wsdl:message name="updateCustomerFaultMsg">
 <wsdl:part element="types:updateCustomerFault"
 name="updateCustomerFault"/>
</wsdl:message>
```

The *concrete* customer-defined elements "types:updateCustomer", "types:updateCustomerResponse" and "types:updateCustomerFault" must be passed to and received from the Web services APIs using <xsd:any> parameters in all *generic* operations (**call**, **sendMessage**, and so on) performed by the client

application. These customer-defined elements are created, serialized and deserialized on the client application side using helper classes that are generated using the exported XSD files.

#### **Related tasks**

“Creating helper classes for BPEL processes (.NET)” on page 259  
Certain Web services API operations require client applications to use “document/literal” style wrapped elements. Client applications require helper classes to help them generate the necessary wrapper elements.

#### **Using files on the client CD:**

As an alternative to exporting artifacts from the WebSphere server environment, you can copy the files necessary for generating a client application from the WebSphere Process Server client CD.

In this case, you must manually modify the default Web services endpoint address of the Business Flow Manager API or Human Task Manager API.

If the client application is to access both APIs, you must edit the default endpoint address for both APIs.

#### *Copying files from the client CD:*

The files necessary to access the Web services APIs are available on the WebSphere Process Server client CD.

1. Access the client CD and browse to the ProcessChoreographer\client directory.
2. Copy the necessary files to your client application development environment.

For the Business Flow Manager API, copy:

#### **BFMWS.wsdl**

Describes the Web services available in the Business Flow Manager Web services API. This file contains the endpoint address.

#### **BFMIF.wsdl**

Describes the parameters and data type of each Web service in the Business Flow Manager Web services API.

#### **BFMIF.xsd**

Describes data types used in the Business Flow Manager Web services API.

#### **BPCGEN.xsd**

Contains data types that are common between the Business Flow Manager and Human Task Manager Web services APIs.

For the Human Task Manager API, copy:

#### **HTMWS.wsdl**

Describes the Web services available in the Human Task Manager Web services API. This file contains the endpoint address.

#### **HTMIF.wsdl**

Describes the parameters and data type of each Web service in the Human Task Manager Web services API.

#### **HTMIF.xsd**

Describes data types used in the Human Task Manager Web services API.

### **BPCGEN.xsd**

Contains data types that are common between the Business Flow Manager and Human Task Manager Web services APIs.

**Note:** The BPCGen.xsd file is common to both APIs.

After you copy the files, you must manually change the Web services API endpoint address the BFMWS.wsdl or HTMWWS.wsdl files to that of the WebSphere application server that is hosting the Web services APIs.

*Manually changing the Web service endpoint address:*

If you copy files from the client CD, you must change the default Web service endpoint address specified in WSDL files to that of the server that is hosting the Web services APIs.

You can use the administrative console to set the Web service endpoint address before exporting the WSDL files. If, however, you copy the WSDL files from the WebSphere Process Server client CD, you must modify the default Web service endpoint address manually.

The Web service endpoint address to use depends on your WebSphere server configuration:

- Scenario 1. There is a single WebSphere server. The WebSphere endpoint address to specify is the host name and port number of the server, for example **host1:9080**.
- Scenario 2. A WebSphere cluster composed of several servers. The WebSphere endpoint address to specify is the host name and port of the server that is hosting the Web services APIs, for example, **host2:9081**.
- Scenario 3. A Web server is used as a front end. The WebSphere endpoint address to specify is the host name and port of the Web server, for example: **host:80**.

*Changing the Business Flow Manager API endpoint:*

If you copy the Business Flow Manager API files from the WebSphere Process Server client CD, you must manually edit the default endpoint address.

1. Navigate to the directory containing the files copied from the client CD.
2. Open the BFMWS.wsdl file in a text editor or XML editor.
3. Locate the soap:address element (towards the bottom of the file).
4. Modify the value of the location attribute with the HTTP URL of the server on which the Web service API is running. To do this:
  - a. Optionally, replace http with https to use the more secure HTTPS protocol.
  - b. Replace localhost with the host name or IP address of the Web services APIs server endpoint address.
  - c. Replace 9080 with the port number of the application server.
  - d. Replace BPEContainer\_N1\_server1 with the "context root" of the application running the Web services API. The default context root is composed of:
    - BPEContainer. The application name.
    - N1. The node name.
    - server1. The server name.
  - e. Do not modify the fixed portion of the URL (/sca/com/ibm/bpe/api/BFMWS) .

For example, if the application is running on the server **s1.n1.ibm.com** and the server is accepting SOAP/HTTP requests at port **9080**, modify the `<soap:address>` element as follows:

```
<soap:address location="http://s1.n1.ibm.com:9080/
 BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
```

#### **Related concepts**

“Adding security (Java Web services)” on page 256

You must secure Web service communications by implementing security mechanisms in your client application.

#### **Related tasks**

“Adding security (.NET)” on page 262

You can secure Web service communications by integrating security mechanisms into your client application.

#### *Changing the Human Task Manager API endpoint:*

If you copy the Human Task Manager API files from the WebSphere Process Server client CD, you must manually edit the default endpoint address.

1. Navigate to the directory containing the files copied from the client CD.
2. Open the HTMWS.wsdl file in a text editor or XML editor.
3. Locate the `soap:address` element (towards the bottom of the file).
4. Modify the value of the `location` attribute with the correct endpoint address.  
To do this:
  - a. Optionally, replace `http` with `https` to use the more secure HTTPS protocol.
  - b. Replace `localhost` with the host name or IP address of the Web services API server’s endpoint address.
  - c. Replace `9080` with the port number of the application server.
  - d. Replace `HTMContainer_N1_server1` with the “context root” of the application running the Web services API. The default context root is composed of:
    - `HTMContainer`. The application name.
    - `N1`. The node name.
    - `server1`. The server name.
  - e. Do not modify the fixed portion of the URL (`/sca/com/ibm/task/api/HTMWS`).

For example, if the application is running on the server **s1.n1.ibm.com** and the server is accepting SOAP/HTTPS requests at port **9081**, modify the `<soap:address>` element as follows:

```
<soap:address location="https://s1.n1.ibm.com:9081/
 HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

#### **Related concepts**

“Adding security (Java Web services)” on page 256

You must secure Web service communications by implementing security mechanisms in your client application.

#### **Related tasks**

“Adding security (.NET)” on page 262

You can secure Web service communications by integrating security mechanisms into your client application.

### **Developing client applications in the Java Web services environment**

You can use any Java-based development environment compatible with Java Web services to develop client applications for the Web services APIs.



## Generating a proxy client (Java Web services):

Java Web service client applications use a *proxy client* to interact with the Web services APIs.

A proxy client for Java Web services contains a number of Java Bean classes that the client application calls to perform Web service requests. The proxy client handles the assembly of service parameters into SOAP messages, sends SOAP messages to the Web service over HTTP, receives responses from the Web service, and passes any returned data to the client application.

Basically, therefore, a proxy client allows a client application to call a Web service as if it were a local function.

**Note:** You only need to generate a proxy client once. All client applications accessing the same Web services API can then use the same proxy client.

In the IBM Web services environment, there are two ways to generate a proxy client:

- Using Rational® Application Developer or WebSphere Integration Developer integrated development environments.
- Using the WSDL2Java command-line tool.

Other Java Web services development environments usually include either the WSDL2Java tool or proprietary client application generation facilities.

*Using Rational Application Developer to generate a proxy client:*

The Rational Application Developer integrated development environment allows you to generate a proxy client for your client application.

Before generating a proxy client, you must have previously exported the WSDL files that describe the business process or human task Web services interfaces from the WebSphere environment (or the WebSphere Process Server client CD) and copied them to your client programming environment.

1. Add the appropriate WSDL file to your project:
  - BFMWS.WSDL for business processes
  - HTMWS.WSDL for human tasks
2. Modify the Web Service wizard properties:
  - a. In Rational Application Developer, choose **Preferences** → **Web services** → **Code generation** → **IBM WebSphere runtime**.
  - b. Select the **Generate Java from WSDL using the no wrapped style** option.
3. Select the BFMWS.WSDL or HTMWS.WSDL file.
4. Choose the client side container type: **client** (client container), **ejb** (EJB container), **java** (Java container), or **none**.
5. Right-click and choose **Web services** → **Generate client**.
6. Follow the on-screen instructions.

A proxy client, made up of a number of proxy, locator and helper Java classes, is generated and added to your project. The deployment descriptor is also updated.

*Using WSDL2Java to generate a proxy client:*



WSDL2Java is a command-line tool that generates a proxy client. A proxy client make it easier to program client applications.

Before generating a proxy client, you must have previously exported the WSDL files that describe the business process or human task Web services APIs from the WebSphere environment (or the WebSphere Process Server client CD) and copied them to your client programming environment.

1. Use the WSDL2Java tool to generate a proxy client: Type:

```
wsdl2java options WSDLfilepath
```

Where:

- *options* include:

**-noWrappedOperations (-w)**

Disables the detection of wrapped operations. Java beans for request and response messages are generated.

**Note:** This is not the default value.

**-role (-r)**

Specify the value **client** to generate files and binding files for client-side development.

**-container (-c)**

The client-side container to use. Valid arguments include:

**client** A client container

**ejb** An Enterprise JavaBeans (EJB) container.

**none** No container

**web** A Web container

**-output (-o)**

The folder in which to store the generated files.

For a complete list of WSDL2Java parameters, use the **-help** command line switch, or refer to the online help for the WSDL2Java tool in the WID/RAD.

- *WSDLfilepath* is the path and filename of the WSDL file that you exported from WebSphere environment or copied from the client CD.

The following example generates a proxy client for the Human Task Activities Web services API:

```
call wsdl2java.bat -r client -c client -noWrappedOperations
-output c:\ws\proxyClient c:\ws\bin\HTMWS.wsdl
```

2. Include the generated class files in your project.

### Creating helper classes for BPEL processes (Java Web services):

Business objects referenced in concrete API requests (for example, sendMessage, or call) require client applications to use "document/literal wrapped" style elements. Client applications require helper classes to help them generate the necessary wrapper elements.

To create helper classes, you must have exported the WSDL file of the Web services API from the WebSphere Process Server environment.

The call() and sendMessage() operations of the Web services APIs allow interaction with BPEL processes on the WebSphere Process Server. The input message of the call() operation expects the document/literal wrapper of the process input message to be provided.

There are a number of possible techniques for generating helper classes for a BPEL process or human task, including:

1. Use the SoapElement object.

In the Rational Application Developer environment available in WebSphere Integration Developer, the Web service engine supports JAX-RPC 1.1. In JAX-RPC 1.1, the SoapElement object extends a Document Object Model (DOM) element, so it is possible to use the DOM API to create, read, load, and save SOAP messages.

For example, assume the WSDL file contains the following input message for a workflow process or human task:

```
<xsd:element name="operation1">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="input1" nillable="true" type="xsd:string"/>
 </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

To create the corresponding SOAP message using the DOM API:

```
SOAPFactory soapfactoryinstance = SOAPFactory.newInstance();
SOAPElement soapmessage = soapfactoryinstance.createElement
 ("operation1", namespaceprefix, interfaceURI);
SOAPElement inputelement = soapfactoryinstance.createElement("input1");
inputelement.addTextNode(message value);
soapmessage.addChildElement(outputelement);
```

The following example shows how to create input parameters for the sendMessage operation:

```
SendMessage inWsend = new SendMessage();
inWsend.setProcessTemplateName(processTemplateName);
inWsend.setPortType(porttype);
inWsend.setOperation(operationname);
inWsend.set_any(soapmessage);
```

2. Use the WebSphere Custom Data Binding feature.

This technique is described in the following developerWorks articles:

- How to choose a custom mapping technology for Web services
- Developing Web Services with EMF SDOs for complex XML schema

Interoperability With Patterns and Strategies for Document-Based Web Services

Web Services support for Schema/WSDL(s) containing optional JAX-RPC 1.0/1.1 XML Schema Types

### Creating a client application (Java Web services):

A client application sends requests to and receives responses from the Web services APIs. By using a proxy client to manage communications and helper classes to format complex data types, a client application can invoke Web service methods as if they were local functions.

Before starting to create a client application, generate the proxy client and any necessary helper classes.

You can develop client applications using any Web services-compatible development tool, for example IBM Rational Application Developer (RAD). You can build any type of Web services application to call the Web services APIs.

1. Create a new client application project.
2. Generate the proxy client and add the Java helper classes to your project.
3. Code your client application.
4. Build the project.
5. Run the client application.

The following example shows how to use the Business Flow Manager Web service API.

```
// create the service locator and the proxy
 BFMWSServiceLocator locator = new BFMWSServiceLocator();
 BFMIF proxy = locator.getBFMWSPort();

// prepare the input data for the operation
 GetProcessTemplate iw = new GetProcessTemplate();
 iw.setIdentifier(your_process_template_name);

// invoke the operation
 GetProcessTemplateResponse ow = proxy.getProcessTemplate(iw);

// process output of the operation
 ProcessTemplateType ptd = ow.getProcessTemplate();
 System.out.println("getName= " + ptd.getName());
 System.out.println("getPtid= " + ptd.getPtid());
```

#### **Related tasks**

“Generating a proxy client (Java Web services)” on page 253

Java Web service client applications use a *proxy client* to interact with the Web services APIs.

“Creating helper classes for BPEL processes (Java Web services)” on page 254  
Business objects referenced in concrete API requests (for example, `sendMessage`, or `call`) require client applications to use “document/literal wrapped” style elements. Client applications require helper classes to help them generate the necessary wrapper elements.

#### **Adding security (Java Web services):**

You must secure Web service communications by implementing security mechanisms in your client application.

WebSphere Application Server currently supports the following security mechanisms for the Web services APIs:

- The user name token
- Lightweight Third Party Authentication (LTPA)

#### **Related concepts**

“Authorization roles for human tasks” on page 157

Actions that you can take on human tasks depend on your authorization role. This role can be a J2EE role or an instance-based role.

“Authorization roles for business processes” on page 149

Actions that you can take on business processes depend on your authorization role. This role can be a J2EE role or an instance-based role.

#### **Related information**

Securing applications during assembly and deployment

### *Implementing the user name token:*

The user name token security mechanism provides user name and password credentials.

With the user name token security mechanism, you can choose to implement various *callback handlers*. Depending on your choice:

- You are prompted to supply a user name and password each time you run the client application.
- The user name and password are written into the deployment descriptor.

In either case, the supplied user name and password must match those of an authorized role in the corresponding business process container or human task container.

The user name and password are encapsulated in the request message envelope, and so appear "in clear" in the SOAP message header. It is therefore strongly recommended that you configure the client application to use the HTTPS (HTTP over SSL) communications protocol. All communications are then encrypted. You can select the HTTPS communications protocol when you specify the Web service API's endpoint URL address.

To define a user name token:

1. In the Rational Application Developer environment available in WebSphere Integration Developer, choose **WS Binding** → **Security Request Generator Binding Configuration** → **Token Generator**.
2. On the Token Generator dialog, choose **Username** as the **Token type**.
3. In the **Call back handler** field, type either **com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler** (which prompts for the user name and password when you run the client application) or **com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler**.
4. If you choose **NonPromptCallbackHandler**, you must specify a valid user name and password in the deployment descriptor.

#### **Related tasks**

"Specifying the Web service endpoint address" on page 246

The Web service endpoint address is the URL that a client application must specify to access the Web services APIs. The endpoint address is written into the WSDL file that you export to generate a proxy client for your client application.

#### **Related information**

IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6

### *Implementing the LTPA security mechanism:*

The Lightweight Third Party Authentication (LTPA) security mechanism can be used when the client application is running within a previously established security context.

The LTPA security mechanism is only available if your client application is running in a secure environment in which a security context has already been established. For example, if your client application is running in an Enterprise JavaBeans (EJB) container, then the EJB client must log in before being able to invoke the client application. A security context is then established. If the EJB client application then

invokes a Web service, the LTPA callback handler retrieves the LTPA token from the security context and adds it to the SOAP request message. On the server side, the LTPA token is handled by the LTPA mechanism.

To implement the LTPA security mechanism:

1. In the Rational Application Developer environment available in WebSphere Integration Developer, choose **WS Binding** → **Security Request Generator Binding Configuration** → **Token Generator**.
2. On the Token Generator dialog, choose **LTPAToken** as the **Token type**.
3. In the **Call back handler** field, type **com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler**.

At runtime, the **LTPATokenCallbackHandler** retrieves the LTPA token from the existing security context and adds it to the SOAP request message.

### **Adding transaction support (Java Web services):**

Java Web service client applications can be configured to allow server-side request processing to participate in the client's transaction, by passing a client application context as part of the service request. This atomic transaction support is defined in the Web Services-Atomic Transaction (WS-AT) specification.

WebSphere Application Server runs each Web services API request as a separate atomic transaction. Client applications can be configured to use transaction support in one of the following ways:

- Participate in the transaction. Server-side request processing is performed within the client application transaction context. Then, if the server encounters a problem while the Web services API request is running and rolls back, the client application's request is also rolled back.
- Not use transaction support. WebSphere Application Server still creates a new transaction in which to run the request, but server-side request processing is not performed with the client application transaction context.

#### **Related information**

Web Services Atomic Transaction support in WebSphere Application Server

## **Developing client applications in the .NET environment**

Microsoft .NET offers a powerful development environment in which to connect applications through Web services.

### **Generating a proxy client (.NET):**

.NET client applications use a *proxy client* to interact with the Web service APIs. A proxy client shields client applications from the complexity of the Web service messaging protocol.

To create a proxy client, you must first export a number of WSDL files from the WebSphere environment and copy them to your client programming environment.

**Note:** If you have the WebSphere Process Server client CD, you can copy the files from there instead.

A proxy client comprises a set of C# bean classes. Each class contains all the methods and objects exposed by a single Web service. The service methods handle

the assembly of parameters into complete SOAP messages, send SOAP messages to the Web service over HTTP, receives responses from the Web service, and handle any returned data.

**Note:** You only need to generate a proxy client once. All client applications accessing the Web services APIs can then use the same proxy client.

1. Use the WSDL command to generate a proxy client: Type:

```
wSDL options WSDLfilepath
```

Where:

- *options* include:

**/language**

Allows you to specify the language used to create the proxy class. The default is C#. You can also specify **VB** (Visual Basic), **JS** (JScript), or **VJS** (Visual J#) as the language argument.

**/output**

The name of the output file, with the appropriate suffix. For example, proxy.cs

**/protocol**

The protocol implemented in the proxy class. **SOAP** is the default setting.

For a complete list of **WSDL.exe** parameters, use the */?* command line switch, or refer to the online help for the WSDL tool in Visual Studio.

- *WSDLfilepath* is the path and filename of the WSDL file that you exported from the WebSphere environment or copied from the client CD.

The following example generates a proxy client for the Human Task Manager Web services API:

```
wSDL /language:cs /output:proxycient.cs c:\ws\bin\HTMWS.wSDL
```

2. Compile the proxy client as a Dynamic Link Library (DLL) file.

### **Creating helper classes for BPEL processes (.NET):**

Certain Web services API operations require client applications to use "document/literal" style wrapped elements. Client applications require helper classes to help them generate the necessary wrapper elements.

To create helper classes, you must have exported the WSDL file of the Web services API from the WebSphere Process Server environment.

The call() and sendMessage() operations of the Web services APIs cause BPEL processes to be launched within WebSphere Process Server. The input message of the call() operation expects the document/literal wrapper of the BPEL process input message to be provided. To generate the necessary beans and classes for the BPEL process, copy the <wSDL:types> element into a new XSD file, then use the **xsd.exe** tool to generate helper classes.

1. If you have not already done so, export the WSDL file of the BPEL process interface from WebSphere Integration Developer.
2. Open the WSDL file in a text editor or XML editor.
3. Copy the contents of all child elements of the <wSDL:types> element and paste it into a new, skeleton, XSD file.
4. Run the **xsd.exe** tool on the XSD file:

```
call xsd.exe file.xsd /classes /o
```

Where:

**file.xsd**

The XML Schema Definition file to convert.

**/classes (/c)**

Generate helper classes that correspond to the contents of the specified XSD file or files.

**/output (/o)**

Specify the output directory for generated files. If this directory is omitted, the default is the current directory.

For example:

**call xsd.exe ProcessCustomer.xsd /classes /output:c:\temp**

5. Add the class file that is generated to your client application. If you are using Visual Studio, for example, you can do this using the **Project** → **Add Existing Item** menu option.

If the ProcessCustomer.wsdl file contains the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:bons1="http://com/ibm/bpe/unittest/sca"
 xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 name="ProcessCustomer"
 targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
 <wsdl:types>
 <xsd:schema targetNamespace="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:bons1="http://com/ibm/bpe/unittest/sca"
 xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
 schemaLocation="xsd-includes/http.com.ibm.bpe.unittest.sca.xsd"/>
 <xsd:element name="doit">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="input1" nillable="true" type="bons1:Customer"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:element name="doitResponse">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="output1" nillable="true" type="bons1:Customer"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 </xsd:schema>
 </wsdl:types>
 <wsdl:message name="doitRequestMsg">
 <wsdl:part element="tns:doit" name="doitParameters"/>
 </wsdl:message>
 <wsdl:message name="doitResponseMsg">
 <wsdl:part element="tns:doitResponse" name="doitResult"/>
 </wsdl:message>
 <wsdl:portType name="ProcessCustomer">
 <wsdl:operation name="doit">
 <wsdl:input message="tns:doitRequestMsg" name="doitRequest"/>
 <wsdl:output message="tns:doitResponseMsg" name="doitResponse"/>
 </wsdl:operation>
 </wsdl:portType>
</wsdl:definitions>
```

The resulting XSD file contains:



```

<xsd:schema xmlns:bons1="http://com/ibm/bpe/unittest/sca"
 xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
 <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
 schemaLocation="Customer.xsd"/>
 <xsd:element name="doit">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="input1" type="bons1:Customer" nillable="true"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:element name="doitResponse">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="output1" type="bons1:Customer" nillable="true"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
</xsd:schema>

```

### Related information

Microsoft documentation for the XML Schema Definition Tool (XSD.EXE)

### Creating a client application (.NET):

A client application sends requests to and receives responses from the Web services APIs. By using a proxy client to manage communications and helper classes to format complex data types, a client application can invoke Web service methods as if they were local functions.

Before starting to create a client application, generate the proxy client and any necessary helper classes.

You can develop .NET client applications using any .NET-compatible development tool, for example, Visual Studio .NET. You can build any type of .NET application to call the generic Web service APIs.

1. Create a new client application project. For example, create a **WinFX Windows Application** in Visual Studio.
2. In the project options, add a reference to the Dynamic Link Library (DLL) file of the proxy client. Add all of the helper classes that contain business object definitions to your project. In Visual Studio, for example, you can do this using the **Project** → **Add existing item** option.
3. Create a proxy client object. For example:

```

HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService service =
 new HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService();

```

4. Declare any business object data types used in messages to be sent to or received from the Web service. For example:

```

HTMClient.HTMReference.TKIID id = new HTMClient.HTMReference.TKIID();

ClipBG bg = new ClipBG();
Clip clip = new Clip();

```

5. Call specific Web service functions and specify any required parameters. For example, to create and start a human task:

```

HTMClient.HTMReference.createAndStartTask task =
 new HTMClient.HTMReference.createAndStartTask();
HTMClient.HTMReference.StartTask sTask = new HTMClient.HTMReference.StartTask();

sTask.taskName = "SimpleTask";

```



```
sTask.taskNamespace = "http://myProcess/com/acme/task";
sTask.inputMessage = bg;
task.inputTask = sTask;
```

```
id = service.createAndStartTask(task).outputTask;
```

6. Remote processes and tasks are identified with persistent IDs (`id` in the example in the previous step). For example, to claim a previously created human task:

```
HTMClient.HTMReference.claimTask claim = new HTMClient.HTMReference.claimTask();
claim.inputTask = id;
```

#### Related tasks

“Generating a proxy client (.NET)” on page 258

.NET client applications use a *proxy client* to interact with the Web service APIs. A proxy client shields client applications from the complexity of the Web service messaging protocol.

“Creating helper classes for BPEL processes (.NET)” on page 259

Certain Web services API operations require client applications to use “document/literal” style wrapped elements. Client applications require helper classes to help them generate the necessary wrapper elements.

#### Adding security (.NET):

You can secure Web service communications by integrating security mechanisms into your client application.

These security mechanisms can include user name token (user name and password), or custom binary and XML-based security tokens.

1. Download and install the Web Services Enhancements (WSE) 2.0 SP3 for Microsoft .NET. This is available from:

<http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en>

2. Modify the generated proxy client code as follows.

Change:

```
public class Export1_MyMicroflowHttpService : System.Web.Services.Protocols.SoapHttpClientProtocol {
 To:
 public class Export1_MyMicroflowHttpService : Microsoft.Web.Services2.WebServicesClientProtocol {
```

**Note:** These modifications are lost if you regenerate the proxy client by running the WSDL.exe tool.

3. Modify the client application code by adding the following lines at the top of the file:

```
using System.Web.Services.Protocols;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Security.Tokens;
...
```

4. Add code to implement the desired security mechanism. For example, the following code adds user name and password protection:

```
string user = "U1";
string pwd = "password";
UsernameToken token =
 new UsernameToken(user, pwd, PasswordOption.SendPlainText);

me._proxy.RequestSoapContext.Security.Tokens.Clear();
me._proxy.RequestSoapContext.Security.Tokens.Add(token);
```

## Querying business-process and task-related objects

You can use the Web services APIs to query business-process and task-related objects in the Business Process Choreographer database to retrieve specific properties of these objects.

The Business Process Choreographer database stores template (model) and instance (runtime) data for managing business processes and tasks.

Through the Web services APIs, client applications can issue queries to retrieve information from the database about business processes and tasks.

Client applications can issue a one-off query to retrieve a specific property of an object. Queries that you use often can be saved. These stored queries can then be retrieved and used by your client application.

### Queries on business-process and task-related objects:

Use the query interface of the Web services APIs to obtain information about business processes and tasks.

Client applications use an SQL-like syntax to query the database.

### Example for Java Web services

```
string processTemplateName = "ProcessCustomerLR";
query query1 = new query();
query1.selectClause = "DISTINCT PROCESS_INSTANCE.STARTED, PROCESS_INSTANCE.PIID";
query1.whereClause =
 "PROCESS_INSTANCE.TEMPLATE_NAME = '" + processTemplateName + "'";
query1.orderByClause = "PROCESS_INSTANCE.STARTED";
query1.threshold = null;
query1.timeZone = "UTC"; query1.skipTuples = null;
queryResponse queryResponse1 = proxy.query(query1);
```

Information retrieved from the database is returned through the Web services APIs as a *query result set*.

For example:

```
QueryResultSetType queryResultSet = queryResponse1.queryResultSet;
if (queryResultSet != null) {
 Console.WriteLine("--> QueryResultSetType");
 Console.WriteLine(" . size= " + queryResultSet.size);
 Console.WriteLine(" . numberColumns= " + queryResultSet.numberColumns);
 string indent = " . ";

 // -- the query column info
 QueryColumnInfoType[] queryColumnInfo = queryResultSet.QueryColumnInfo;
 if (queryColumnInfo.Length > 0) {
 Console.WriteLine();
 Console.WriteLine("= . QueryColumnInfoType size= " + queryColumnInfo.Length);
 Console.WriteLine(" | tableName ");
 for (int i = 0; i < queryColumnInfo.Length ; i++) {
 Console.WriteLine(" | " + queryColumnInfo[i].tableName.PadLeft(20));
 }
 Console.WriteLine();
 Console.WriteLine(" | columnName ");
 for (int i = 0; i < queryColumnInfo.Length ; i++) {
 Console.WriteLine(" | " + queryColumnInfo[i].columnName.PadLeft(20));
 }
 Console.WriteLine();
 Console.WriteLine(" | data type ");
 for (int i = 0; i < queryColumnInfo.Length ; i++) {
```

```

 QueryColumnInfoType tt = queryColumnInfo[i].type;
 Console.WriteLine(" | " + tt.ToString());
 }
 Console.WriteLine();
}
else {
 Console.WriteLine("--> queryColumnInfo= <null>");
}

// - the query result values
string[][] result = queryResultSet.result;
if (result !=null) {
 Console.WriteLine();
 Console.WriteLine("= . result size= " + result.Length);
 for (int i = 0; i < result.Length; i++) {
 Console.Write(indent + i);
 string[] row = result[i];
 for (int j = 0; j < row.Length; j++) {
 Console.Write(" | " + row[j]);
 }
 Console.WriteLine();
 }
}
else {
 Console.WriteLine("--> result= <null>");
}
}
else {
 Console.WriteLine("--> QueryResultSetType= <null>");
}
}

```

The query function returns objects according to the caller's authorization. The query result set only contains the properties of those objects that the caller is authorized to see.

Predefined database views are provided for you to query the object properties. 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 task templates, the query function has the following syntax:

```

TaskTemplate[] queryTaskTemplates
 (java.lang.String whereClause,
 java.lang.String orderByClause,
 java.lang.Integer threshold,
 java.util.TimeZone timezone);

```

For the other business-process and task-related 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 skipTuples
 java.lang.Integer threshold,
 java.util.TimeZone timezone);

```

The query interface also contains a queryAll method. You can use this method to retrieve all of the relevant data about an object, for example, for monitoring purposes. The caller of the queryAll method must have one of the following Java 2 Platform, Enterprise Edition (J2EE) roles: BPESystemAdministrator,

BPESystemMonitor, TaskSystemAdministrator, or TaskSystemMonitor. Authorization checking using the corresponding work item of the object is not applied.

### Example for .NET

```
ProcessTemplateType[] templates = null;

try {
 queryProcessTemplates iW = new queryProcessTemplates();
 iW.whereClause = "PROCESS_TEMPLATE.STATE=PROCESS_TEMPLATE.STATE.STATE_STARTED";
 iW.orderByClause = null;
 iW.threshold = null;
 iW.timeZone = null;

 Console.WriteLine("--> queryProcessTemplates ... ");
 Console.WriteLine("--> query: WHERE " + iW.whereClause + " ORDER BY " +
 iW.orderByClause + " THRESHOLD " + iW.threshold + " TIMEZONE " + iW.timeZone);

 templates = proxy.queryProcessTemplates(iW);

 if (templates.Length < 1) {
 Console.WriteLine("--> No templates found :-(");
 }
 else {
 for (int i = 0; i < templates.Length ; i++) {
 Console.WriteLine("--> found template with ptid: " + templates[i].ptid);
 Console.WriteLine(" and name: " + templates[i].name);
 /* ... other properties of ProcessTemplateType ... */
 }
 }
}
catch (Exception e) {
 Console.WriteLine("exception= " + e);
}
```

#### *Query parameters:*

Each query must specify a number of SQL-like clauses and parameters.

A query is made up of:

- Select clause
- Where clause
- Order-by clause
- Skip-tuples parameter
- Threshold parameter
- Time-zone parameter

#### **Predefined views for queries on business-process and human-task objects:**

Predefined database views are provided for business-process and human-task objects.

Use these views when you query reference data for these objects. When you use these views, you do not need to explicitly add join predicates for view columns, these constructs are added automatically for you. You can use the query function of the Web services APIs to query this data.

#### **Managing stored queries:**

Stored queries provide a way to save queries that are run often. The stored query can be either a query that is available to all users (public query), or a query that belongs to a specific user (private query).

A stored query is a query that is stored in the database and identified by a name. A private and a public stored query can have the same name; private stored queries from different owners can also have the same name.

You can have stored queries for business process objects, task objects, or a combination of these two object types.

#### Managing public stored queries

Public stored queries are created by the system administrator. These queries are available to all users.

#### Managing private stored queries for other users

Private queries can be created by any user. These queries are available only to the owner of the query and the system administrator.

#### Working with your private stored queries

If you are not a system administrator, you can create, run, and delete your own private stored queries. You can also use the public stored queries that the system administrator created.

## Developing Web applications for business processes and human tasks, using JSF components

Business Process Choreographer Explorer provides several JavaServer Faces (JSF) components. You can extend and integrate these components to add business-process and human-task functionality to Web applications.

You can use WebSphere Integration Developer to build your Web application.

1. Create a dynamic project and change the Web Project Features properties of the Web project to include the JSF base components.

For more information on creating a Web project, go to the information center for WebSphere Integration Developer.

2. Add the prerequisite Business Process Choreographer Explorer Java archive (JAR files).

Add the following files to the WEB-INF/lib directory of your project:

- bpcclientcore.jar
- bfmclientmodel.jar
- htmclientmodel.jar
- bpcjsfcomponents.jar

These files are in the *install\_root*/ProcessChoreographer/client directory.

3. Add the EJB references that you need to the Web application deployment descriptor, web.xml file.

```
<ejb-ref id="EjbRef_1">
 <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
 <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
<ejb-ref id="EjbRef_2">
 <ejb-ref-name>ejb/HumanTaskManagerEJB</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.task.api.HumanTaskManagerHome</home>
 <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
```

```

<ejb-local-ref id="EjbLocalRef_1">
 <ejb-ref-name>ejb/LocalBusinessProcessHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
 <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
<ejb-local-ref id="EjbLocalRef_2">
 <ejb-ref-name>ejb/LocalHumanTaskManagerEJB</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
 <local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>

```

4. Add the Business Process Choreographer Explorer JSF components to the JSF application.
  - a. Add the tag libraries that you need for your applications to the JavaServer Pages (JSP) files. Typically, you need the JSF and HTML tag libraries, and the tag library required by the JSF components.
    - `<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>`
    - `<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>`
    - `<%@ taglib uri="http://com.ibm.bpe.jsf/taglib" prefix="bpe" %>`
  - b. Add an `<f:view>` tag to the body of the JSP page, and an `<h:form>` tag to the `<f:view>` tag.
  - c. Add the JSF components to the JSP files.  
Depending on your application, add the List component, the Details component, the CommandBar component, or the Message component to the JSP files. You can add multiple instances of each component.
  - d. Configure the managed beans in the JSF configuration file.  
By default, the configuration file is the `faces-config.xml` file. This file is in the `WEB-INF` directory of the Web application. Depending on the component that you add to your JSP file, you also need to add the references to the query and other wrapper objects to the JSF configuration file.
  - e. Implement the custom code that you need to support the JSF components.
5. Deploy the application.  
Map the EJB references to the Java Naming and Directory Interface (JNDI) names or manually add the references to the `ibm-web-bnd.xml` file.  
The following table lists the reference bindings and their default mappings.

Table 30. Mapping of the reference bindings to JNDI names

Reference binding	JNDI name	Comments
ejb/BusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	Remote session bean
ejb/LocalBusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	Local session bean
ejb/HumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	Remote session bean
ejb/LocalHumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	Local session bean

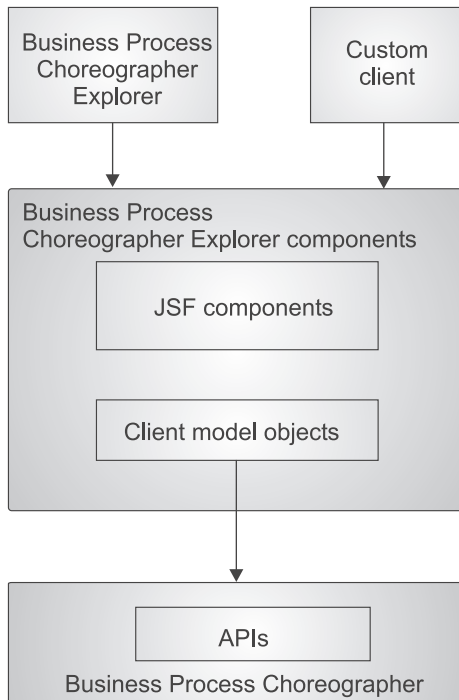
Your deployed Web application contains the functionality provided by the Business Process Choreographer Explorer components.

### Business Process Choreographer Explorer components

The Business Process Choreographer Explorer components are a set of configurable, reusable elements that are based on the JavaServer Faces (JSF)

technology. You can imbed these elements in Web applications. The Web applications can then access installed business process and human task applications.

The components consist of a set of JSF components and a set of client model objects. The relationship of the components to Business Process Choreographer, Business Process Choreographer Explorer, and other custom clients is shown in the following figure.



## JSF components

The Business Process Choreographer Explorer components include the following JSF components. You imbed these JSF components in your JavaServer Pages (JSP) files when you build Web applications for working with business processes and human tasks.

- List component  
The List component displays a list of application objects in a table, for example, tasks, activities, process instances, process templates, work items, or escalations. This component has an associated list handler.
- Details component  
The Details component displays the properties of tasks, work items, activities, process instances, and process templates. This component has an associated details handler.
- CommandBar component  
The CommandBar component displays a bar with buttons. These buttons represent commands that operate on either the object in a details view or the selected objects in a list. These objects are provided by a list handler or a details handler.
- Message component

The Message component displays a message that can contain either a Service Data Object (SDO) or a simple type.

## Client model objects

The client model objects are used with the JSF components. The objects implement some of the interfaces of the underlying Business Process Choreographer API and wrap the original object. The client model objects provide national language support for labels and converters for some properties.

### List handling in the List component:

Every instance of the List component is associated with an instance of the `com.ibm.bpe.jsf.handler.BPCListHandler` class.

This list handler tracks the selected items in the associated list and it provides a notification mechanism. The list handler is bound to the List component through the **model** attribute of the `bpe:list` tag.

The notification mechanism of the list handler is implemented using the `com.ibm.bpe.jsf.handler.ItemListener` interface. Business Process Choreographer Explorer uses this notification mechanism to associate the list entries with the details pages for the different kinds of items. The trigger for a notification event is typically one of the properties of the items that is displayed in the current list.

To exploit the notification mechanism, set the value of the **action** attribute of the `bpe:column` tag for the property to the JavaServer Faces (JSF) navigation target your application is to continue with when the notification event is triggered. The List component renders the entry in the column as a JSF command link. If the link is triggered, the object that represents the entry in the list is determined, and it is passed to all of the registered item listeners. You can register implementations of this interface in the configuration file of your JSF application.

The `BPCListHandler` class also provides a `refreshList` method. You can use this method in JSF method bindings to implement a user interface control for running the query again.

### Query implementations

You can use the list handler to display all kinds of objects and their properties. The content of the list that is displayed depends on the list of objects that is returned by the implementation of the `com.ibm.bpc.clientcore.Query` interface that is configured for the list handler. You can set the query either programmatically using the `setQuery` method of the `BPCListHandler` class, or you can configure it in the JSF configuration files of the application.

You can run queries not only against the Business Process Choreographer APIs, but also against any other source of information that is accessible from your application, for example, a content management system or a database. The only requirement is that the result of the query is returned as a `java.util.List` of objects by the `execute` method.

The type of the objects returned must guarantee that the appropriate getter methods are available for all of the properties that are displayed in the columns of the list for which the query is defined. To ensure that the type of the object that is returned fits the list definitions, you can set the value of the `type` property on the



BPCListHandler instance that is defined in the faces configuration file to the fully qualified class name of the returned objects. You can return this name in the `getType` call of the query implementation. At run time, the list handler checks that the object types conform to the definitions.

To map error messages to specific entries in a list, the objects returned by the query must implement a method with the signature `public Object getID()`.

### Error handling

You can take advantage of the error handling functions provided by the BPCListHandler class in the following error situations.

- Errors that occur when queries are run or commands are executed

If an error occurs during the execution of a query, the BPCListHandler class distinguishes between errors that were caused by insufficient access rights and other exceptions. To catch errors due to insufficient access rights, the **rootCause** parameter of the ClientException that is thrown by the `execute` method of the query must be a `com.ibm.bpe.api.EngineNotAuthorizedException` or a `com.ibm.task.api.NotAuthorizedException` exception. The List component displays the error message instead of the result of the query.

If the error is not caused by insufficient access rights, the BPCListHandler class passes the exception object to the implementation of the `com.ibm.bpc.clientcore.util.ErrorBean` interface that is defined by the `BPCError` key in your JSF application configuration file. When the exception is set, the error navigation target is called.

- Errors that occur when working with items that are displayed in a list

The BPCListHandler class implements the `com.ibm.bpe.jsf.handler.ErrorHandler` interface. You can provide information about these errors with the `map` parameter of type `java.util.Map` in the `setErrors` method. This map contains identifiers as keys and the exceptions as values. The identifiers must be the values returned by the `getID` method of the object that caused the error. If the map is set and any of the IDs match any of the items displayed in the list, the list handler automatically adds a column containing the error message to the list.

To avoid outdated error messages in the list, reset the errors map. In the following situations, the map is reset automatically:

- The `refreshList` method BPCListHandler class is called.
- A new query is set on the BPCListHandler class.
- The `CommandBar` component is used to trigger actions on items of the list. The `CommandBar` component uses this mechanism as one of the methods for error handling.

### CommandBar component:

Use the `CommandBar` component to add action buttons to your application. The component creates the buttons for the actions in the user interface and handles the events that are created when a button is clicked.

These buttons trigger functions that act on the objects that are returned by a `com.ibm.bpe.jsf.handler.ItemProvider` interface, such as the BPCListHandler class, or the `BPCDetailsHandler` class. The `CommandBar` component uses the item provider that is defined by the value of the **model** attribute in the `bpe:commandbar` tag.

## How commands are processed

When a button in the command-bar section of the application's user interface is clicked, the associated event is handled by the CommandBar component in the following way.

1. The CommandBar component identifies the implementation of the `com.ibm.bpc.clientcore.Command` interface that is specified for the button that generated the event.
2. If the model associated with the CommandBar component implements the `com.ibm.bpe.jsf.handler.ErrorHandler` interface, the `clearErrorMap` method is invoked to remove error messages from previous events.
3. The `getSelectedItems` method of the `ItemProvider` interface is called. The list of items that is returned is passed to the `execute` method of the command, and the command is invoked.
4. The CommandBar component determines the JavaServer Faces (JSF) navigation target. If an **action** attribute is not specified in the `bpe:commandbar` tag, the return value of the `execute` method specifies the navigation target. If the **action** attribute is set to a JSF method binding, the string returned by the method is interpreted as the navigation target. The **action** attribute can also specify an explicit navigation target.

## Error handling

An action method that is specified by the **action** attribute in the `bpe:commandbar` tag is invoked if one of the following conditions are met:

- An exception is not thrown
- If an exception is thrown, it is an `ErrorsInCommandException` exception

You can implement error handling in the CommandBar component in several ways:

- You can decide not to use any of the features of the CommandBar component. If, for example, you want to display the errors on a page that is specific to the selected command, the implementation of the command can catch the exceptions that occur and propagate them to a page bean that is used for the error page. You can make the page bean available to the command implementation by using the **context** attribute of the `bpe:commandbar` tag. After the exception is set on the page bean, the command returns the string of the JSF navigation rule that is defined for the error page.
- If you want to display an error message below the command-bar section in the user interface, create an exception class that implements the `com.ibm.bpc.clientcore.exception.CommandBarMessage` marker interface. This interface provides a message catalog of error messages.
- If the command operates on a list of items, you might want to track the success of the command for each of the items in the list. To track the errors, map each exception to the item for which the operation failed. The CommandBar component can pass a map, which contains the identifiers as keys and the exceptions as values, to the model object that is defined for the CommandBar component.

For this mechanism to work, the model object must implement the `com.ibm.bpe.jsf.handler.ErrorHandler` interface and the command must throw a `com.ibm.bpc.clientcore.exception.ErrorsInCommandException` exception. The CommandBar component then passes the map contained in the exception to the error handler. The action method is triggered although an error occurred, and

the current view is refreshed. The Business Process Choreographer Explorer application makes use of this method to display exceptions in lists.

- If you throw a `ClientException` exception that does not implement the `CommandBarMessage` interface and the exception is not an `ErrorsInCommandException`, the `CommandBar` component propagates the exception to the `BPCError` error bean that is defined in the configuration file of your application. The error processing continues with the error page.

### Utilities provided by the Business Process Choreographer Explorer JSF components:

The JavaServer Faces (JSF) components provide utilities for user-specific time zone information and error handling.

#### User-specific time zone information

The `BPCListHandler` class uses the `com.ibm.bpc.clientcore.util.User` interface to get information about the time zone and locale of each user. The `List` component expects the implementation of the interface to be configured with `user` as the managed-bean name in your JavaServer Faces (JSF) configuration file. If this entry is missing from the configuration file, the time zone in which WebSphere Process Server is running is returned.

The `com.ibm.bpc.clientcore.util.User` interface is defined as follows:

```
public interface User {

 /**
 * The locale used by the client of the user.
 * @return Locale.
 */
 public Locale getLocale();
 /**
 * The time zone used by the client of the user.
 * @return TimeZone.
 */
 public TimeZone getTimeZone();

 /**
 * The name of the user.
 * @return name of the user.
 */
 public String getName();
}
```

#### ErrorBean interface for error handling

The JSF components exploit a predefined managed bean, `BPCError`, for error handling. This bean implements the `com.ibm.bpc.clientcore.util.ErrorBean` interface. In error situations that trigger the error page, the exception is set on the error bean. The error page is displayed in the following situations:

- If an error occurs during the execution of a query that is defined for a list handler, and the error is generated as a `ClientException` error by the `execute` method of a command
- If a `ClientException` error is generated by the `execute` method of a command and this error is not an `ErrorsInCommandException` error nor does it implement the `CommandBarMessage` interface
- If an error message is displayed in the component, and you follow the hyperlink for the message

A default implementation of the `com.ibm.bpc.clientcore.util.ErrorBeanImpl` interface is available.

The interface is defined as follows:

```
public interface ErrorBean {

 public void setException(Exception ex);

 /*
 * This setter method call allows a locale and
 * the exception to be passed. This allows the
 * getExceptionMessage methods to return localized Strings
 */
 public void setException(Exception ex, Locale locale);

 public Exception getException();
 public String getStack();
 public String getNestedExceptionMessage();
 public String getNestedExceptionStack();
 public String getRootExceptionMessage();
 public String getRootExceptionStack();

 /*
 * This method returns the exception message
 * concatenated recursively with the messages of all
 * the nested exceptions.
 */
 public String getAllExceptionMessages();

 /*
 * This method is returns the exception stack
 * concatenated recursively with the stacks of all
 * the nested exceptions.
 */
 public String getAllExceptionStacks();
}
```

## Adding the List component to a JSF application

Use the Business Process Choreographer Explorer List component to display a list of client model objects, for example, business process instances or task instances.

1. Add the List component to the JavaServer Pages (JSP) file.

Add the `bpe:list` tag to the `h:form` tag. The `bpe:list` tag must include a model attribute. Add `bpe:column` tags to the `bpe:list` tag to add the properties of the objects that are to appear in each of the rows in the list.

The following example shows how to add a List component to display task instances.

```
<h:form>

 <bpe:list model="#{TaskPool}">
 <bpe:column name="name" action="taskInstanceDetails" />
 <bpe:column name="state" />
 <bpe:column name="kind" />
 <bpe:column name="owner" />
 <bpe:column name="originator" />
 </bpe:list>

</h:form>
```

The model attribute refers to a managed bean, `TaskPool`. The managed bean provides the list of Java objects over which the list iterates and then displays in individual rows.

2. Configure the managed bean referred to in the `bpe:list` tag.

For the List component, this managed bean must be an instance of the `com.ibm.bpe.jsf.handler.BPCListHandler` class.

The following example shows how to add the TaskPool managed bean to the configuration file.

```
<managed-bean>
<managed-bean-name>TaskPool</managed-bean-name>
<managed-bean-class>com.ibm.bpe.jsf.handler.BPCListHandler</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>

 <managed-property>
 <property-name>query</property-name>
 <value>#{TaskPoolQuery}</value>
 </managed-property>

 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>
</managed-bean>
<managed-bean>
<managed-bean-name>htmConnection</managed-bean-name>
<managed-bean-class>com.ibm.task.clientmodel.HTMConnection</managed-bean-class>
<managed-bean-scope>application</managed-bean-scope>

 <managed-property>
 <property-name>jndiName</property-name>
 <value>java:comp/env/ejb/LocalHumanTaskManagerEJB</value>
 </managed-property>
</managed-bean>
```

The example shows that TaskPool has two configurable properties: `query` and `type`. The value of the `query` property refers to another managed bean, `TaskPoolQuery`. The value of the `type` property specifies the bean class, the properties of which are shown in the columns of the displayed list. The associated query instance can also have a property type. If a property type is specified, it must be the same as the type specified for the list handler.

To provide a connection to Human Task Manager, the TaskPool managed bean is implemented using the `htmConnection` managed bean.

3. Add the custom code for the managed bean that is referred to by the list handler.

The following example shows how to add custom code for the TaskPool managed bean.

```
public class MyTaskQuery implements Query {

 public List execute throws ClientException {

 // Examine the faces-config file for a managed bean "htmConnection".
 //
 FacesContext ctx = FacesContext.getCurrentInstance();
 Application app = ctx.getApplication();
 ValueBinding htmVb = app.createValueBinding("#{htmConnection}");
 HTMConnection htmConnection = (HTMConnection) htmVb.getValue(ctx);
 HumanTaskManagerService taskService =
 htmConnection.getHumanTaskManagerService();

 // Then call the actual query method on the Human Task Manager service.
 //
 QueryResultSet queryResult = taskService.query(
 "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
 + "TASK.STARTED, TASK.ACTIVATED, TASK.DUE, TASK.EXPIRES, TASK.PRIORITY" ,
```

```

 "TASK.KIND IN(101,102,105) AND TASK.STATE IN(2)
 AND WORK_ITEM.REASON IN (1)",
 null,
 null,
 null);
 List applicationObjects = transformToTaskList (queryResult);
 return applicationObjects ;
}

```

```

private List transformToTaskList(QueryResultSet result) {

ArrayList array = null;
int entries = result.size();
array = new ArrayList(entries);

// Transforms each row in the QueryResultSet to a task instance beans.
for (int i = 0; i < entries; i++) {
 result.next();
 array.add(new TaskInstanceBean(result, connection));
}
return array ;
}
}

```

The `TaskPoolQuery` bean queries the properties of the Java objects. This bean must implement the `com.ibm.bpc.clientcore.Query` interface. When the list handler refreshes its contents, it calls the `execute` method of the query. The call returns a list of Java objects. The `getType` method must return the class name of the returned Java objects.

Your JSF application now contains a `JavaServer` page that displays the properties of the requested list of objects, for example, the state, kind, owner, and originator of the task instances that are available to you.

### List component: Tag definitions:

The Business Process Choreographer Explorer List component displays a list of objects in a table, for example, tasks, activities, process instances, process templates, work items, and escalations.

The List component consists of the JSF component tags: `bpe:list` and `bpe:column`. The `bpe:column` tag is a subelement of the `bpe:list` tag.

### Component class

`com.ibm.bpe.jsf.component.ListComponent`

### Example syntax

```

<bpe:list model="#{ProcessTemplateList}">
 rows="20"
 styleClass="list"
 headerStyleClass="listHeader"
 rowClasses="normal">

 <bpe:column name="name" action="processTemplateDetails"/>
 <bpe:column name="validFromTime"/>
 <bpe:column name="executionMode" label="Execution mode"/>
 <bpe:column name="state" converterID="my.state.converter"/>
 <bpe:column name="autoDelete"/>
 <bpe:column name="description"/>

</bpe:list>

```

## Tag attributes

The body of the `bpe:list` tag can contain only `bpe:column` tags. When the table is rendered, the List component iterates over the list of application objects and provides the specific object for each column.

Table 31. *bpe:list* attributes

Attribute	Required	Description
model	yes	A value binding for a managed bean of the <code>com.ibm.bpe.jsf.handler.BPCListHandler</code> class.
styleClass	no	The cascading style sheet (CSS) style for rendering the overall table containing titles, rows, and paging buttons.
headerStyleClass	no	The CSS style class for rendering the table header.
cellStyleClass	no	The CSS style class for rendering individual table cells.
buttonStyleClass	no	The CSS style class for rendering the buttons in the footer area.
rowClasses	no	The CSS style class for rendering the rows in the table.
rows	no	The number of rows that are shown on a page. If the number of items exceeds the number of rows, paging buttons are displayed at the end of the table. Value expressions are not supported for this attribute.
checkbox	no	Determines whether the check box for selecting multiple items is rendered. The attribute has a value of either <code>true</code> or <code>false</code> .

Table 32. *bpe:column* attributes

Attribute	Required	Description
name	yes	The name of the object property that is shown in this column. This name must correspond to a named property as defined in the corresponding client model class.
action	no	If this attribute is specified as an outcome string, it defines an outcome used by the JavaServer Faces (JSF) navigation handler to determine the next page.  If this attribute is specified as a method binding ( <code>#{...}</code> ), the method to be called has the signature <code>String method()</code> and its return value is used by the JSF navigation handler to determine the next page.

Table 32. `bpe:column` attributes (continued)

Attribute	Required	Description
label	no	The label displayed in the header of the column or the cell of the table header row. If this attribute is not set, a default label is provided by the client model class.
converterID	no	The ID used to register the converter in the JSF configuration file. If a converter ID is not specified, the implementation of the objects displayed in the list can contain a definition of a converter for the current property. The List component uses this converter.

## Adding the Details component to a JSF application

Use the Business Process Choreographer Explorer Details component to display the properties of tasks, work items, activities, process instances, and process templates.

1. Add the Details component to the JavaServer Pages (JSP) file.

Add the `bpe:details` tag to the `<h:form>` tag. The `bpe:details` tag must contain a model attribute. You can add properties to the Details component with the `bpe:property` tag. If the Details component does not contain any properties, all of the properties of the object are displayed.

The following example shows how to add a Details component to display some of the properties for a task instance.

```
<h:form>

 <bpe:details model="#{TaskInstanceDetails}">
 <bpe:property name="displayName" />
 <bpe:property name="owner" />
 <bpe:property name="kind" />
 <bpe:property name="state" />
 <bpe:property name="escalated" />
 <bpe:property name="suspended" />
 <bpe:property name="originator" />
 <bpe:property name="activationTime" />
 <bpe:property name="expirationTime" />
 </bpe:details>

</h:form>
```

The model attribute refers to a managed bean, `TaskInstanceDetails`. The bean provides the properties of the Java object.

2. Configure the managed bean referred to in the `bpe:details` tag.

For the Details component, this managed bean must be an instance of the `com.ibm.bpe.jsf.handler.BPCDetailsHandler` class. This handler class wraps a Java object and exposes its public properties to the details component.

The following example shows how to add the `TaskInstanceDetails` managed bean to the configuration file.

```
<managed-bean>
 <managed-bean-name>TaskInstanceDetails</managed-bean-name>
 <managed-bean-class>com.ibm.bpe.jsf.handler.BPCDetailsHandler</managed-bean-class>
 <managed-bean-scope>session</managed-bean-scope>
 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>
</managed-bean>
```



The example shows that the TaskInstanceDetails bean has a configurable type property. The value of the type property specifies the bean class (com.ibm.task.clientmodel.bean.TaskInstanceBean), the properties of which are shown in the rows of the displayed details.

Your JSF application now contains a JavaServer page that displays the details of the specified object, for example, the details of a task instance.

### Details component: Tag definitions:

The Business Process Choreographer Explorer Details component displays the properties of tasks, work items, activities, process instances, and process templates.

The Details component consists of the JSF component tags: bpe:details and bpe:property. The bpe:property tag is a subelement of the bpe:details tag.

### Component class

com.ibm.bpe.jsf.component.DetailsComponent

### Example syntax

```
<bpe:details model="#{MyActivityDetails}">
 <bpe:property name="name"/>
 <bpe:property name="owner"/>
 <bpe:property name="activated"/>
</bpe:details>
<bpe:details model="#{MyActivityDetails}" style="style" styleClass="cssStyle">
 style="style"
 styleClass="cssStyle"
</bpe:details>
```

### Tag attributes

Use bpe:property tags to specify both the subset of attributes that are shown and the order in which these attributes are shown. If the details tag does not contain any attribute tags, it renders all of the available attributes of the model object.

Table 33. bpe:details attributes

Attribute	Required	Description
model	yes	A value binding for a managed bean of the com.ibm.bpe.jsf.handler.BPCDetailsHandler class.
styleClass	no	The cascading style sheet style (CSS) class for rendering the HTML element.
columnClasses	no	A list of CSS styles, separated by commas, for rendering columns.
rowClasses	no	A list of CSS styles, separated by commas, for rendering rows.

Table 34. bpe:property attributes

Attribute	Required	Description
name	yes	The name of the property to be displayed. This name must correspond to a named property as defined in the corresponding client model class.

Table 34. *bpe:property attributes (continued)*

Attribute	Required	Description
label	no	The label for the property. If this attribute is not set, a default label is provided by the client model class.
converterID	no	The ID used to register the converter in the JavaServer Faces (JSF) configuration file.

## Adding the CommandBar component to a JSF application

Use the Business Process Choreographer Explorer CommandBar component to display a bar with buttons. These buttons represent commands that operate on the details view of an object or the selected objects in a list.

When the user clicks a button in the user interface, the corresponding command is run on the selected objects. You can add and extend the CommandBar component in your JSF application.

1. Add the CommandBar component to the JavaServer Pages (JSP) file.

Add the `bpe:commandbar` tag to the `<h:form>` tag. The `bpe:commandbar` tag must contain a model attribute.

The following example shows how to add a CommandBar component that provides refresh and claim commands for a task instance list.

```
<h:form>

 <bpe:commandbar model="#{TaskInstanceList}">
 <bpe:command commandID="Refresh" >
 action="#{TaskInstanceList.refreshList}"
 label="Refresh"/>

 <bpe:command commandID="MyClaimCommand" >
 label="Claim" >
 commandClass="<customcode>"/>
 </bpe:commandbar>

</h:form>
```

The **model** attribute refers to a managed bean. This bean must implement the `ItemProvider` interface and provide the selected Java objects. The CommandBar component is usually used with either the List component or the Details component in the same JSP file. Generally, the model that is specified in the tag is the same as the model that is specified in the List component or Details component on the same page. So for the List component, for example, the command acts on the selected items in the list.

In this example, the **model** attribute refers to the `TaskInstanceList` managed bean. This bean provides the selected objects in the task instance list. The bean must implement the `ItemProvider` interface. This interface is implemented by the `BPCListHandler` class and the `BPCDetailsHandler` class.

2. **Optional:** Configure the managed bean that is referred to in the `bpe:commandbar` tag.

If the CommandBar **model** attribute refers to a managed bean that is already configured, for example, for a list or details handler, no further configuration is required. If you change the configuration of either of these handlers or you use a different managed bean, add a managed bean that implements the `ItemProvider` interface to the JSF configuration file.

3. Add the code that implements the custom commands to the JSF application.

The following code snippet shows how to write a command class that extends the command bar. This command class (MyClaimCommand) is referred to by the `bpe:command` tag in the JSP file.

The command checks the preconditions and any other prerequisites, for example, the correct number of selected items. It then retrieves a reference to the human task API, `HumanTaskManagerService`. The command iterates over the selected objects and tries to process them. The task is claimed through the `HumanTaskManagerService` API by an ID. If an exception does not occur, the state is updated for the corresponding `TaskInstanceBean` object. This action avoids retrieving the value of the object from the server again.

```
public class MyClaimCommand implements Command {

 public String execute(List selectedObjects) throws ClientException {
 if(selectedObjects != null && selectedObjects.size() > 0) {
 try {
 // Determine HumanTaskManagerService from an HTMConnection bean.
 // Configure the bean in the faces-config.xml for easy access
 // in the JSF application.
 FacesContext ctx = FacesContext.getCurrentInstance();
 ValueBinding vb =
 ctx.getApplication().createValueBinding("{htmConnection}");
 HTMConnection htmConnection = (HTMConnection) htmVB.getValue(ctx);
 HumanTaskManagerService htm =
 htmConnection.getHumanTaskManagerService();

 Iterator iter = selectedObjects.iterator() ;
 while(iter.hasNext()) {
 try {
 TaskInstanceBean task = (TaskInstanceBean) iter.next() ;
 TKIID tiid = task.getID() ;

 htm.claim(tiid) ;
 task.setState(new Integer(TaskInstanceBean.STATE_CLAIMED)) ;

 }
 catch(Exception e) {
 ; // Error while iterating or claiming task instance.
 // Ignore for better understanding of the sample.
 }
 }
 }
 catch(Exception e) {
 ; // Configuration or communication error.
 // Ignore for better understanding of the sample
 }
 }
 return null;
 }

 // Default implementations
 public boolean isMultiSelectEnabled() { return false; }
 public boolean[] isApplicable(List itemsOnList) {return null; }
 public void setContext(Object targetModel) {}; // Not used here
}
```

The command is processed in the following way:

- a. A command is invoked when a user clicks the corresponding button in the command bar. The `CommandBar` component retrieves the selected items from the item provider that is specified in the **model** attribute and passes the list of selected objects to the `execute` method of the `commandClass` instance.
- b. The **commandClass** attribute refers to a custom command implementation that implements the `Command` interface. This means that the command

must implement the public String execute(List selectedObjects) throws ClientException method. The command returns a result that is used to determine the next navigation rule for the JSF application.

- c. After the command completes, the CommandBar component evaluates the **action** attribute. The **action** attribute can be a static string or a method binding to a JSF action method with the public String Method() signature. Use the **action** attribute to override the outcome of a command class or to explicitly specify an outcome for the navigation rules. The **action** attribute is not processed if the command generates an exception other than an ErrorsInCommandException exception.

Your JSF application now contains a JavaServer page that implements a customized command bar.

### CommandBar component: Tag definitions:

The Business Process Choreographer Explorer CommandBar component displays a bar with buttons. These buttons operate on the object in a details view or the selected objects in a list.

The CommandBar component consists of the JSF component tags: bpe:commandbar and bpe:command. The bpe:command tag is a subelement of the bpe:commandbar tag.

### Component class

com.ibm.bpe.jsf.component.CommandBarComponent

### Example syntax

```
<bpe:commandbar model="#{TaskInstanceList}">
 <bpe:command
 commandID="Work on"
 label="Work on..."
 commandClass="com.ibm.bpc.explorer.command.WorkOnTaskCommand"
 context="#{TaskInstanceDetailsBean}" />
 <bpe:command
 commandID="Cancel"
 label="Cancel"
 commandClass="com.ibm.task.clientmodel.command.CancelClaimTaskCommand"
 context="#{TaskInstanceList}" />
</bpe:commandbar>
```

### Tag attributes

Table 35. bpe:commandbar attributes

Attribute	Required	Description
model	yes	A value binding expression to a managed bean that implements the ItemProvider interface. This managed bean is usually the com.ibm.bpe.jsf.handler.BPCListHandler class or the com.ibm.bpe.jsf.handler.BPCDetailsHandler class that is used by the List component or Details component in the same JavaServer Pages (JSP) file as the CommandBar component.

Table 35. *bpe:commandbar* attributes (continued)

Attribute	Required	Description
styleClass	no	The cascading style sheet (CSS) style for rendering the bar.
buttonStyleClass	no	The CSS style for rendering the buttons in the command bar.

Table 36. *bpe:command* attributes

Attribute	Required	Description
commandID	yes	The ID of the command.
commandClass	yes	The command class that is triggered.
action	no	<p>A JavaServer Faces (JSF) action method that has the signature: <code>String method()</code>. The value that is returned by the <b>action</b> method, or that is directly specified as a literal overrides the target returned by the <code>execute</code> method of the command. The <b>action</b> attribute is not processed if the command generates an exception other than an <code>ErrorsInCommandException</code> exception.</p> <p>If this attribute is specified as an outcome string, it defines a result that is used by the JSF navigation handler to determine the navigation rule and the next page to display.</p> <p>If this attribute is specified as a method binding (<code>#{...}</code>), the method to be called has the signature <code>String method()</code>. Its return value is used by the JSF navigation handler to determine the navigation rule and the next page to display.</p>
label	yes	The label of the button that is rendered in the command bar.
styleClass	no	The CSS style for rendering the button. This style overrides the button style defined for the command bar.
context	no	A value binding expression, which refers to a managed bean. Use this attribute if the command needs to initialize the target page or bean.

## Adding the Message component to a JSF application

Use the Business Process Choreographer Explorer Message component to render data objects and primitive types in a JavaServer Faces (JSF) application.

If the message type is a primitive type, a label and an input field are rendered. If the message type is a data object, the component traverses the object and renders the elements within the object.

1. Add the Message component to the JavaServer Pages (JSP) file.

Add the `bpe:form` tag to the `<h:form>` tag. The `bpe:form` tag must include a `model` attribute.

The following example shows how to add a Message component.

```

<h:form>
 <h:outputText value="Input Message" />
 <bpe:form model="#{MyHandler.inputMessage}" readOnly="true" />

 <h:outputText value="Output Message" />
 <bpe:form model="#{MyHandler.outputMessage}" />

</h:form>

```

The **model** attribute of the Message component refers to a `com.ibm.bpc.clientcore.MessageWrapper` object. This wrapper object wraps either a Service Data Object (SDO) object or a Java primitive type, for example, `int` or `boolean`. In the example, the message is provided by a property of the `MyHandler` managed bean.

2. Configure the managed bean referred to in the `bpe:form` tag.

The following example shows how to add the `MyHandler` managed bean to the configuration file.

```

<managed-bean>
<managed-bean-name>MyHandler</managed-bean-name>
<managed-bean-class>com.ibm.bpc.sample.jsf.MyHandler</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>

 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>

</managed-bean>

```

3. Add the custom code to the JSF application.

The following example shows how to implement input and output messages.

```

public class MyHandler implements ItemListener {

 private TaskInstanceBean taskBean;
 private MessageWrapper inputMessage, outputMessage

 /* Listener method, e.g. when a task instance was selected in a list handler.
 * Ensure that the handler is registered in the faces-config.xml or manually.
 */
 public void itemChanged(Object item) {
 if(item instanceof TaskInstanceBean) {
 taskBean = (TaskInstanceBean) item ;
 }
 }

 /* Get the input message wrapper
 */
 public MessageWrapper getInputMessage() {
 try{
 inputMessage = taskBean.getInputMessageWrapper() ;
 }
 catch(Exception e) {
 ; //...ignore errors for simplicity
 }
 return inputMessage;
 }

 /* Get the output message wrapper
 */
 public MessageWrapper getOutputMessage() {
 // Retrieve the message from the bean. If there is no message, create
 // one if the task has been claimed by the user. Ensure that only
 // potential owners or owners can manipulate the output message.
 try{

```

```

 outputMessage = taskBean.getOutputMessageWrapper();
 if(outputMessage == null
 && taskBean.getState() == TaskInstanceBean.STATE_CLAIMED) {
 HumanTaskManagerService htm = getHumanTaskManagerService();
 outputMessage = new MessageWrapperImpl();
 outputMessage.setMessage(
 htm.createOutputMessage(taskBean.getID()).getObject()
);
 }
 }
 catch(Exception e) {
 ; //...ignore errors for simplicity
 }
 return outputMessage
}
}
}

```

The MyHandler managed bean implements the `com.ibm.jsf.handler.ItemListener` interface so that it can register itself as an item listener to list handlers. When the user clicks an item in the list, the MyHandler bean is notified in its `itemChanged( Object item )` method about the selected item. The handler checks the item type and then stores a reference to the associated `TaskInstanceBean` object. To use this interface, add an entry to the appropriate list handler in the `faces-config.xml` file.

The MyHandler bean provides the `getInputMessage` and `getOutputMessage` methods. Both of these methods return a `MessageWrapper` object. The methods delegate the calls to the referenced task instance bean. If the task instance bean returns null, for example, because a message is not set, the handler creates and stores a new, empty message. The Message component displays the messages provided by the MyHandler bean.

Your JSF application now contains a JavaServer page that can render data objects and primitive types.

### Message component: Tag definitions:

The Business Process Choreographer Explorer Message component renders `commonj.sdo.DataObject` objects and primitive types, such as integers and strings, in a JavaServer Faces (JSF) application.

The Message component consists of the JSF component tag: `bpe:form`.

### Component class

`com.ibm.bpe.jsf.component.MessageComponent`

### Example syntax

```

<bpe:form model="#{TaskInstanceDetailsBean.inputMessageWrapper}"
 simplification="true" readOnly="true"
 styleClass4table="messageData"
 styleClass4output="messageDataOutput">
</bpe:form>

```

## Tag attributes

Table 37. *bpe:form* attributes

Attribute	Required	Description
model	yes	A value binding expression that refers to either a <code>commonj.sdo.DataObject</code> object or a <code>com.ibm.bpc.clientcore.MessageWrapper</code> object.
simplification	no	If this attribute is set to true, properties with a cardinality of zero or one are shown. By default, this attribute is set to true.
readOnly	no	If this attribute is set to true, a read-only form is rendered. By default, this attribute is set to false.
style4validinput	no	The cascading style sheet (CSS) style for rendering input that is valid.
style4invalidinput	no	The CSS style for rendering input that is not valid.
styleClass4validInput	no	The CSS class name for rendering input that is valid.
styleClass4invalidInput	no	The CSS class name for rendering input that is not valid.
styleClass4output	no	The CSS style class name for rendering the output elements.
styleClass4table	no	The class name of the CSS table style for rendering the tables rendered by the message component.
buttonStyleClass	no	The CSS style for the buttons that work on arrays or lists.

## Mapping of client model objects

The client model objects implement the corresponding interfaces of the Business Process Choreographer API.

This wrapping of the interfaces provides locale-sensitive labels and converters for a set of properties. The following table shows the mapping of the Business Process Choreographer interfaces to the corresponding client model objects.

Table 38. How Business Process Choreographer interfaces are mapped to client model objects

Business Process Choreographer interface	Client model object class
<code>com.ibm.bpe.api.ActivityInstanceData</code>	<code>com.ibm.bpe.clientmodel.bean.ActivityInstanceBean</code>
<code>com.ibm.bpe.api.ActivityServiceTemplateData</code>	<code>com.ibm.bpe.clientmodel.bean.ActivityServiceTemplateBean</code>
<code>com.ibm.bpe.api.ProcessInstanceData</code>	<code>com.ibm.bpe.clientmodel.bean.ProcessInstanceBean</code>
<code>com.ibm.bpe.api.ProcessTemplateData</code>	<code>com.ibm.bpe.clientmodel.bean.ProcessTemplateBean</code>
<code>com.ibm.task.api.Escalation</code>	<code>com.ibm.task.clientmodel.bean.EscalationBean</code>
<code>com.ibm.task.api.Task</code>	<code>com.ibm.task.clientmodel.bean.TaskInstanceBean</code>
<code>com.ibm.task.api.TaskTemplate</code>	<code>com.ibm.task.clientmodel.bean.TaskTemplateBean</code>





---

## Chapter 6. Deploying

---

### Installing business process and human task applications

You can distribute Service Component Architecture (SCA) Enterprise JavaBeans (EJB) modules that contain business processes or human tasks, or both, to deployment targets. A deployment target can be a server or a cluster.

Verify that the business process container or task container is installed and configured for each application server or cluster on which you want to install your application.

Before you install a business process or human task application, make sure that the following conditions are true:

- The servers on which you want to install the application are running.
- In each cluster, at least one server on which you want to install Enterprise JavaBeans modules with processes or tasks is running.

You can install business process and task applications from the administrative console, from the command line, or by running an administrative script, for example. When you run an administrative script to install a business process application or a human task application, a server connection is required. Do not use the `-conntype NONE` option as an installation option.

1. If you are installing an application on a cluster, verify that the application uses the data source that is named after the cluster.

For example, if the application was generated using the default data source BPEDB, change the data source for the application to `BPEDB_cluster_name`, where `cluster_name` is the name of the cluster on which you installed the application.

2. Install the application.

All business process templates and human task templates are put into the start state. You can create process instances and task instances from these templates.

Before you can create process instances or task instances, you must start the application.

### Deployment of models

When WebSphere Integration Developer or service deploy generates the deployment code for your process, the constructs in the process or task model are mapped to various Java 2 Enterprise Edition (J2EE) constructs and artifacts. All deployment code is packaged into the enterprise application (EAR) file. Each new version of a model that is to be deployed must be packaged into a new enterprise application.

When you install an enterprise application that contains business process model or human task model J2EE constructs, the model constructs are stored as *process templates* or *task templates*, as appropriate, in the Business Process Choreographer database. If the database system is not running, or if it cannot be accessed, the deployment fails. Newly installed templates are, by default, in the started state. However, the newly installed enterprise application is in the stopped state. Each installed enterprise application can be started and stopped individually.

New versions of a process template or task template have the same name, but a different valid-from attribute. You can deploy many different versions of a process template or task template, each in a different enterprise application. However, no two versions of the same process can have the same valid-from date. If you want to install different versions of the same process, specify a different valid-from date for each version. All the different process versions are stored in the database.

If you do not specify a valid-from date, the date is determined as follows:

- For a human task, the valid-from date is the date on which the application was installed.
- For a business process, the valid-from date is the date on which the process was modeled.

## Deploying business process applications interactively

You can install an application interactively at runtime using the wsadmin tool and the installInteractive script. You can use this script to change settings that cannot be changed if you use the administrative console to install the application.

Perform the following steps to install business process applications interactively.

1. Start the wsadmin tool.

In the *profile\_root/bin* directory, enter wsadmin.

2. Install the application.

At the wsadmin command-line prompt, enter the following command:

```
$AdminApp installInteractive application.ear
```

where *application.ear* is the qualified name of the enterprise archive file that contains your process application. You are prompted through a series of tasks where you can change values for the application.

3. Save the configuration changes.

At the wsadmin command-line prompt, enter the following command:

```
$AdminConfig save
```

You must save your changes to transfer the updates to the master configuration repository. If a scripting process ends and you have not saved your changes, the changes are discarded.

## Configuring process application data source and set reference settings

You might need to configure process applications that run SQL statements for the specific database infrastructure. These SQL statements can come from information service activities or they can be statements that you run during process installation or instance startup.

When you install the application, you can specify the following types of data sources:

- Data sources to run SQL statements during process installation
- Data sources to run SQL statements during the startup of a process instance
- Data sources to run SQL snippet activities

The data source required to run an SQL snippet activity is defined in a BPEL variable of type `tDataSource`. The database schema and table names that are required by an SQL snippet activity are defined in BPEL variables of type `tSetReference`. You can configure the initial values of both of these variables.

You can use the wsadmin tool to specify the data sources.

1. Install the process application interactively using the wsadmin tool.
2. Step through the tasks until you come to the tasks for updating data sources and set references.

Configure these settings for your environment. The following example shows the settings that you can change for each of these tasks.

3. Save your changes.

### **Example: Updating data sources and set references, using the wsadmin tool**

In the **Updating data sources** task, you can change data source values for initial variable values and statements that are used during installation of the process or when the process starts. In the **Updating set references** task, you can configure the settings related to the database schema and the table names.

Task [24]: Updating data sources

```
//Change data source values for initial variable values at process start
```

```
Process name: Test
// Name of the process template
Process start or installation time: Process start
// Indicates whether the specified value is evaluated
//at process startup or process installation
Statement or variable: Variable
// Indicates that a data source variable is to be changed
Data source name: MyDataSource
// Name of the variable
JNDI name:[jdbc/sample]:jdbc/newName
// Sets the JNDI name to jdbc/newName
```

Task [25]: Updating set references

```
// Change set reference values that are used as initial values for BPEL variables
```

```
Process name: Test
// Name of the process template
Variable: SetRef
// The BPEL variable name
JNDI name:[jdbc/sample]:jdbc/newName
// Sets the JNDI name of the data source of the set reference to jdbc/newName
Schema name: [IISAMPLE]
// The name of the database schema
Schema prefix: []:
// The schema name prefix.
// This setting applies only if the schema name is generated.
Table name: [SETREFTAB]: NEWTABLE
// Sets the name of the database table to NEWTABLE
Table prefix: []:
// The table name prefix.
// This setting applies only if the prefix name is generated.
```

## **When you can install a process application on a cluster in which no servers are running**

This topic explains the exceptional circumstances in which you might need to install an application on a cluster that has no running servers.

During the installation of a business process application on a server, the Java Naming and Directory Interface (JNDI) name of the data source of the corresponding business process container must be resolved. You cannot, therefore,

install an application without a server connection. In a Network Deployment (ND) environment, this server is the deployment manager.

## Restrictions lifted

If you want to install a business process application on a cluster in an ND environment, no server in the cluster need be running if the following conditions are true:

- The required data sources are defined at the cell level.
- The process application does not specify human tasks.

For process applications that have no human tasks, the data source lookup operation is accomplished within the namespace of the deployment manager, when a lookup operation in the namespace of the application server previously failed. If the application is successfully installed, ignore any error messages in the `SystemOut.log` file that indicate a failure of the data source lookup operation within the application server namespace.

## When it will work

- The lookup operation within the deployment manager namespace is successful only if the data source JNDI name is defined at the cell level.
- If you use the wizard to configure a business process container or human task container on a stand-alone server, the data source is defined at the server level. The same is true if you use the configuration script `bpeconfig.jacl`, which is provided in the `ProcessChoreographer/config` directory of your application server installation. In this case, you must define the data source manually at the cell level and use this data source when you install the business process container.
- If you configure a business process container with the wizard on a cluster member, the data source is automatically defined at the cell level. The JNDI name is scoped by the cluster name. The same is true if you use the configuration script `bpeconfig.jacl`, which is provided in the `ProcessChoreographer/config` directory of your application server installation. In this case, you do not need to change anything manually.

## When it will not work

Process applications that contain human tasks require an additional JNDI name lookup operation to locate the staff plug-in provider. Therefore, to help ensure successful installation of such applications, make sure that the cluster includes a running server.

## Scoping side effects

A side effect of the name lookup is that if an application server is not running and a data source is defined on its server or node level with the same name as a data source at the cell level, the cell level data source takes precedence. This means that you might end up using different data sources during deployment than you use at run time.

**Attention:** Avoid name clashes. If you define data sources at the cell level manually, use JNDI names that have the scope of the cluster name or server name and node name appended to them, for example, `jdbc/BPEDB_cluster_name`.

## Uninstalling business process and human task applications, using the administrative console

To uninstall an enterprise application that contains business processes or human tasks, perform the following actions:

1. Stop all process and task templates in the application.

This action prevents the creation of process and task 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**, then select an Enterprise JavaBeans (EJB) module. If you have more than one EJB module, select the EJB module that corresponds to the Service Component Architecture (SCA) module that contains the business process or human task. You can find the corresponding EJB module by appending EJB to the SCA module name. For example, if your SCA module was named TestProcess, the EJB module is TestProcessEJB.jar.
- d. Under Additional Properties, click **Business Processes** or **Human Tasks**, or both, as appropriate.
- e. Select all process and task templates by clicking the appropriate check box.
- f. Click **Stop**.

Repeat this step for all EJB modules that contain business process templates or human task templates.

2. Verify that the database, at least one application server for each cluster, and the stand-alone server where the application is deployed are running.

In a Network Deployment (ND) environment, the deployment manager, all ND-managed stand-alone application servers, and at least one application server must be running for each cluster where the application is installed.

3. Verify that no process instances or task instances are running nor that any are in end states with the autoDelete flag set to false.

If necessary, an administrator can use Business Process Choreographer Explorer to delete any process or task instances.

4. 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 or task instances still exist in the application.
- c. Select again the application that you want to uninstall, and click **Uninstall**.
- d. Click **Save** to save your changes.

The application is uninstalled.

## Uninstalling business process and human task applications, using administrative commands

Administrative commands provide an alternative to the administrative console for uninstalling applications that contain business processes or human tasks.

If global security is enabled, verify that your user ID has operator authorization.

Ensure that the server process to which the administration client connects is running.

- In an ND environment, the server process is the deployment manager.
- In a stand-alone environment, the server process is the application server.

To ensure that the administrative client automatically connects to the server process, do not use the `-conntype NONE` option as a command option.

The following steps describe how to use the `bpcTemplates.jacl` script to uninstall applications that contain business process templates or human task templates. You must stop a template before you can uninstall the application to which it belongs. You can use the `bpcTemplates.jacl` script to stop and uninstall templates in one step.

Before you uninstall applications, you can delete process instances or task instances associated with the templates in the applications, for example, using Business Process Choreographer Explorer. You can also use the `-force` option with the `bpcTemplates.jacl` script to delete any instances associated with the templates, stop the templates, and uninstall them in one step.

**CAUTION:**

**Because this option deletes all process instance and task instance data, you should use this option with care.**

1. Change to the Business Process Choreographer samples directory. Type the following:

```
cd install_root/ProcessChoreographer/admin
```

2. Stop the templates and uninstall the corresponding application.

```
install_root/bin/wsadmin -f bpcTemplates.jacl
 [-user user_name]
 [-password user_password]
 -uninstall application_name
 [-force]
```

Where:

*user\_name*

If global security is enabled, provide the user ID for authentication.

*user\_password*

If global security is enabled, provide the user password for authentication.

*application\_name*

If global security is enabled, provide the user password for authentication.

**-force**

Causes any running instances to be stopped and deleted before the application is uninstalled.

The application is uninstalled.

---

## Chapter 7. Monitoring

---

### Monitoring business processes and human tasks

Monitoring of processes and human tasks is controlled through the monitoring pane in the WebSphere Integration Developer. This approach has to be followed regardless of whether audit trailing is to be enabled or whether events are to be emitted.

WebSphere Process Server includes the Common Event Infrastructure that provides standard formats and mechanisms for managing event data.

Business Process Choreographer sends events whenever situations occur that require monitoring and the Common Event Infrastructure service is available. These events adhere to the Common Base Event specification. You can use generic tools to process these events.

You can also use Java snippets to create and send user data events. For more information, see the Common Event Infrastructure documentation on sending events.

### Situation-independent event data

Common Base Events that are emitted on behalf of business processes and human tasks can contain information that is independent of the situation for which the event was created. This event data is the same for all business process and human task events.

The following table shows the values of the attributes of the `CommonBaseEvent` element and of the `sourceComponentID` element that these events can contain.

Attribute	Description
<b>CommonBaseEvent element</b>	
<code>creationTime</code>	The time at which the event is created in universal coordinated time (UTC).
<code>globalInstanceId</code>	The identifier of the Common Base Event instance. This ID is automatically generated.
<code>sequenceNumber</code>	Sequential numbering issued by the event factory.
<code>severity</code>	The impact that the event has on business processes or on human tasks. This attribute is set to 10 (information).
<code>version</code>	Set to 1.0.1.
<code>extensionName</code>	The value depends on the object that creates the event and on the event.
<b>sourceComponentId element</b>	
<code>component</code>	For business processes and human tasks: Set to WPS#, followed by the identification of the current platform and the version identification of the underlying software stack.
<code>componentIdType</code>	Set to the string: "ProductName".
<code>executionEnvironment</code>	A string that identifies the operating system.



Attribute	Description
instanceId	The identifier of the server. This identifier has the format <i>cell name/node name/server name</i> . The delimiters are platform dependent.
location	Set to the host name of the executing server.
locationType	Set to the IP address or host name.
processId	The process identifier of the operating system.
subcomponent	For business processes, set to BFM. For human tasks, set to HTM.
threadId	The thread identifier of the Java Virtual Machine (JVM).
componentType	For business processes, set to:  www.ibm.com/namespaces/autonomic/Workflow_Engine  For human tasks, set to:  www.ibm.com/xmlns/prod/websphere/scdl/human-task

## Business process events

Events that are emitted on behalf of business processes consist of situation-independent data and data that is specific to business process events. The attributes and elements that are specific to business process events are described.

Business process events can have the following categories of event content.

### Event data specific to business processes

In business processes, events relate to processes, activities, scopes, links, and variables. The object-specific content of each of these event types is described.

If not specified otherwise, the object-specific content is written as *extendedDataElement* XML elements of type string.

### Process

Events of process instances have the following object-specific event content:

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived
processTemplateValidFrom	The date from which the template is valid
processTemplateId	The identifier of the process template
processInstanceDescription	Optional: The description of the process instance

Attribute	Description
processInstanceExecutionState	<p>A string value that represents the state of the process. It has the format: <i>state number-state description</i>. This attribute can have one of the following values:</p> <ul style="list-style-type: none"> <li>1 - STATE_READY</li> <li>2 - STATE_RUNNING</li> <li>3 - STATE_FINISHED</li> <li>4 - STATE_COMPENSATING</li> <li>5 - STATE_FAILED</li> <li>6 - STATE_TERMINATED</li> <li>7 - STATE_COMPENSATED</li> <li>8 - STATE_TERMINATING</li> <li>9 - STATE_FAILING</li> <li>10 - STATE_INDOUBT</li> <li>11 - STATE_SUSPENDED</li> <li>12 - STATE_COMPENSATION_FAILED</li> </ul>
PayloadType	The string full

### Activity and scope

Activities and scopes have the following object-specific event content:

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived.
processTemplateValidFrom	The date from which the template is valid.
activityTemplateName	Optional: The name of the activity template from which the instance was derived.
activityInstanceDescription	Optional: The description of the activity instance.

Attribute	Description
activityKind	<p>A string value that identifies the activity kind. This value has the format: <i>kind number-kind description</i>. This attribute can have one of the following values:</p> <ul style="list-style-type: none"> <li>3 - KIND_EMPTY</li> <li>21 - KIND_INVOKE</li> <li>23 - KIND_RECEIVE</li> <li>24 - KIND_REPLY</li> <li>25 - KIND_THROW</li> <li>26 - KIND_TERMINATE</li> <li>27 - KIND_WAIT</li> <li>29 - KIND_COMPENSATE</li> <li>30 - KIND_SEQUENCE</li> <li>32 - KIND_SWITCH</li> <li>34 - KIND_WHILE</li> <li>36 - KIND_PICK</li> <li>38 - KIND_FLOW</li> <li>42 - KIND_SCRIPT</li> <li>43 - KIND_STAFF</li> <li>44 - KIND_ASSIGN</li> <li>45 - KIND_CUSTOM</li> <li>46 - KIND_RETHROW</li> <li>47 - KIND_FOR_EACH_SERIAL</li> <li>48 - KIND_FOR_EACH_PARALLEL</li> <li>1000 - SQLSnippet</li> <li>1001 - RetrieveSet</li> <li>1002 - InvokeInformationService</li> <li>1003 - AtomicSQLSnippetSequence</li> </ul>
state	<p>A string value that represents the state of the activity. It has the format: <i>state number-state description</i>. Note that the state codes for activities are different from those used for processes. This attribute can have one of the following values:</p> <ul style="list-style-type: none"> <li>1 - STATE_INACTIVE</li> <li>2 - STATE_READY</li> <li>3 - STATE_RUNNING</li> <li>4 - STATE_SKIPPED</li> <li>5 - STATE_FINISHED</li> <li>6 - STATE_FAILED</li> <li>7 - STATE_TERMINATED</li> <li>8 - STATE_CLAIMED</li> <li>9 - STATE_TERMINATING</li> <li>10 - STATE_FAILING</li> <li>11 - STATE_WAITING</li> <li>12 - STATE_EXPIRED</li> <li>13 - STATE_STOPPED</li> </ul>
bpellId	<p>A string value that represents the wpc:id attribute of the activity.</p>
PayloadType	<p>The payload type. The value of the string can be one of: none, digest, or full. The value depends on the setting in WebSphere Integration Developer and whether business object (BO) content is written to the event. If an event does not contain a business object, the value is always set to full.</p>

## Link

Links have the following object-specific event content:

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived
processTemplateValidFrom	The date from which the template is valid
flowBpelId	A string value that represents the wpc:id attribute of the flow activity that contains the link
elementName	The name of the link that was evaluated
description	A description of the link. This attribute is only included if specified in the process model.
PayloadType	The string full

## Variable

Variables have the following object-specific event content.

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived.
processTemplateValidFrom	The date from which the template is valid.
variableName	The name of the variable that was changed.
variableData	Emitted when WBI Monitor compatible events are requested. An XML representation of the content of the variable. Each property of the data object is reported in the form of a nested extended data element. The element type may be of type 'boolean' or 'string', with an appropriate value.
variableData_BO	Emitted when non-WBI Monitor compatible events are requested. This element is of type 'noValue' and contains an XML representation of the content of the variable. Each property of the data object is reported in the form of a nested extended data element.
bpelId	A string value that represents the wpc:id attribute of the activity.
PayloadType	The payload type. The value of the string can be one of: none, digest, or full. The value depends on the setting in WebSphere Integration Developer and whether business object content is written to the event. If an event does not contain a business object, the value is always set to full.

## Situations in business process events

Business process events can be emitted in different situations. The data for these situations is described in situation elements.

Business process events can contain one of the following situation elements.

Situation name	Content of the Common Base Event	
Start	categoryName is set to StartSituation.	
	situationType	
	Type	StartSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	START_COMPLETED
Stop	categoryName is set to StopSituation.	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	STOP_COMPLETED
Destroy	categoryName is set to DestroySituation.	
	situationType	
	Type	DestroySituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
Fail	categoryName is set to StopSituation.	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	UNSUCCESSFUL
	situationQualifier	STOP_COMPLETED
Report	categoryName is set to ReportSituation.	
	situationType	
	Type	ReportSituation
	reasoningScope	EXTERNAL
	reportCategory	STATUS

## Business process events

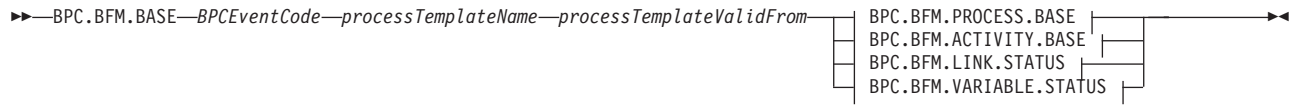
Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer. A list of all the events that can be emitted by business processes can be found here.

The following types of events can be caused by business process:

- “Process events” on page 299
- “Activity events” on page 301
- “Activity scope events” on page 303
- “Link events” on page 304
- “Variable events” on page 305

## XML syntax

The payloads for business process events have the following syntax:



Where:

### *BPCEventCode*

The Business Process Choreographer event code that identifies the number of the event type. Possible event codes are listed in the following tables.

### *processTemplateName*

The name of the process template.

### *processTemplateValidFrom*

The valid from attribute of the process template.

The name of event elements are in uppercase, for example BPC.BFM.BASE, and the names of extended data elements are in mixed case, for example, *BPCEventCode*. Except where indicated, all data elements are of the type string.

## Key to table columns

The columns in the following tables contain:

**Code** Contains the number of the event. This value is provided as the *BPCEventCode* extended data element for all BPC.BFM.BASE elements.

### **Extension name**

Contains the string value that is used as the value of the *extensionName* attribute of the Common Base Event. This is also the name of the XML extended data element provide additional data about the event.

If WebSphere Business Integration Modeler is used to generate the Business Process Execution Language (BPEL) and the monitoring specification, the extension name can be extended by a hash character (#) followed by additional characters. Also, events that emit message data contain additional *extendedDataElements*. Refer to the documentation for WebSphere Business Integration Modeler for more information.

### **Situation**

Refers to the situation name of the business process event. For details of situations, see "Situations in business process events" on page 297.

### **Event nature**

A pointer to the event situation for a business process element in the EventNature parameter, as they are displayed in WebSphere Integration Developer.

## Process events

The following table describes all process events.

Code	Description	Extension name	Situation	Event nature
21000	Process started	BPC.BFM.PROCESS.START	Start	ENTRY

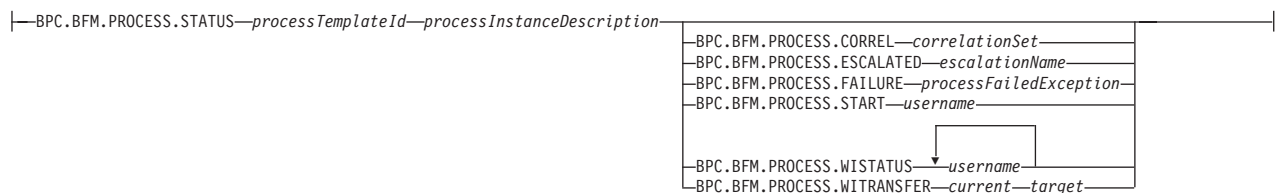
Code	Description	Extension name	Situation	Event nature
21001	Process suspended	BPC.BFM.PROCESS.STATUS	Report	SUSPENDED
21002	Process resumed	BPC.BFM.PROCESS.STATUS	Report	RESUMED
21004	Process completed	BPC.BFM.PROCESS.STATUS	Stop	EXIT
21005	Process terminated	BPC.BFM.PROCESS.STATUS	Stop	TERMINATED
21019	Process restarted	BPC.BFM.PROCESS.START	Report	RESTARTED
21020	Process deleted	BPC.BFM.PROCESS.STATUS	Destroy	DELETED
42001	Process failed	BPC.BFM.PROCESS.FAILURE	Fail	FAILED
42003	Process compensating	BPC.BFM.PROCESS.STATUS	Report	COMPENSATING
42004	Process compensated	BPC.BFM.PROCESS.STATUS	Stop	COMPENSATED
42009	Process terminating	BPC.BFM.PROCESS.STATUS	Report	TERMINATING
42010	Process failing	BPC.BFM.PROCESS.STATUS	Report	FAILING
42027	Correlation set initialized	BPC.BFM.PROCESS.CORREL	Report	CORRELATION
42041	Process work item deleted	BPC.BFM.PROCESS.WISTATUS	Report	WI_DELETED
42042	Process work item created	BPC.BFM.PROCESS.WISTATUS	Report	WI_CREATED
42046	Process compensation failed	BPC.BFM.PROCESS.STATUS	Fail	COMPFAILED
42047	Process event received	BPC.BFM.PROCESS.STATUS	Report	EV_RECEIVED
42049	Process event escalated	BPC.BFM.PROCESS.ESCALATED	Report	EV_ESCALATED
42056	Process work item transferred	BPC.BFM.PROCESS.WITRANSFER	Report	WI_TRANSFERRED

The payloads for process events have the following syntax:

### BPC.BFM.PROCESS.BASE

▶▶—BPC.BFM.PROCESS.BASE—*processInstanceExecutionState*—| BPC.BFM.PROCESS.STATUS |————▶▶

### BPC.BFM.PROCESS.STATUS:



Where:

*processInstanceExecutionState*

The current execution state of the process in the following format: <state code>-<state name>

*processTemplateId*

The ID of the process template.

*processInstanceDescription*

The description of the process instance.

*correlationSet*

The correlation set instance, in the following format:

```
<?xml version="1.0"?>
<correlationSet name="correlation set name">
 <property name="property name"
 value="property value"/>*
</correlationSet>
```

*escalationName*

The name of the escalation.

*processFailedException*

The exception message that lead to the failure of the process.

*username*

For BPC.BFM.PROCESS.START this is the name of the user who requested the start or restart of the process. For BPC.BFM.PROCESS.WISTATUS this is a list of users whose work item was created or deleted.

*current*

The user name of the current owner of the work item. This is the user whose work item has been transferred away.

*target*

The user name of the new owner of the work item.

For process events, the following event correlation sphere identifiers are also written to the Common Base Event as context data elements:

- The ECSCurrentID provides the ID of the process instance.
- The ECSParentID provides the value of the ECSCurrentID before the process instance start event of the current process.

**Activity events**

The following table describes all activity events.

Code	Description	Extension name	Situation	Event nature
21006	Activity ready	BPC.BFM.ACTIVITY.STATUS	Start	CREATED
21007	Activity started	For invoke activities: BPC.BFM.ACTIVITY.MESSAGE. For all other activity types: BPC.BFM.ACTIVITY.STATUS	Start	ENTRY
21011	Activity completed	For invoke, staff, receive, and reply activities: BPC.BFM.ACTIVITY.MESSAGE. For all other activity types: BPC.BFM.ACTIVITY.STATUS	Stop	EXIT
21021	Claim canceled	BPC.BFM.ACTIVITY.STATUS	Report	DEASSIGNED
21022	Activity claimed	BPC.BFM.ACTIVITY.CLAIM	Report	ASSIGNED
21027	Activity terminated	BPC.BFM.ACTIVITY.STATUS	Stop	TERMINATED



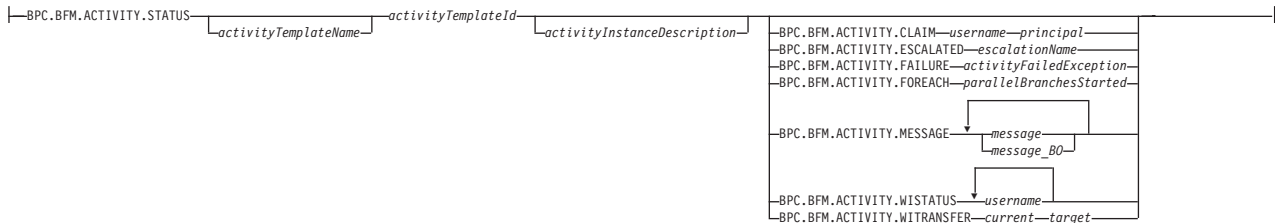
Code	Description	Extension name	Situation	Event nature
21080	Activity failed	BPC.BFM.ACTIVITY.FAILURE	Failed	FAILED
21081	Activity expired	BPC.BFM.ACTIVITY.STATUS	Report	EXPIRED
42005	Activity skipped	BPC.BFM.ACTIVITY.STATUS	Report	SKIPPED
42012	Activity output message set	BPC.BFM.ACTIVITY.MESSAGE	Report	OUTPUTSET
42013	Activity fault message set	BPC.BFM.ACTIVITY.MESSAGE	Report	FAULTSET
42015	Activity stopped	BPC.BFM.ACTIVITY.STATUS	Stop	STOPPED
42031	Activity force retried	BPC.BFM.ACTIVITY.STATUS	Report	FRETRIED
42032	Activity force completed	BPC.BFM.ACTIVITY.STATUS	Stop	FCOMPLETED
42036	Activity has message received	BPC.BFM.ACTIVITY.MESSAGE	Report	EXIT
42037	Loop condition true	BPC.BFM.ACTIVITY.STATUS	Report	CONDTRUE
42038	Loop condition false	BPC.BFM.ACTIVITY.STATUS	Report	CONDFALSE
42039	Work item deleted	BPC.BFM.ACTIVITY. WISTATUS	Report	WI_DELETED
42040	Work items created	BPC.BFM.ACTIVITY. WISTATUS	Report	WI_CREATED
42050	Activity escalated	BPC.BFM.ACTIVITY.ESCALATED	Report	ESCALATED
42054	Activity work items refreshed	BPC.BFM.ACTIVITY. WISTATUS	Report	WI_REFRESHED
42055	Work item transferred	BPC.BFM.ACTIVITY. WITRANSFER	Report	WI_TRANSFERRED
42057	For each - activity branches started	BPC.BFM.ACTIVITY. FOREACH	Report	BRANCHES_STARTED

The payloads for activity events have the following syntax:

### BPC.BFM.ACTIVITY.BASE

►► BPC.BFM.ACTIVITY.BASE—*activityKind*—*state*—*bpelId*—| BPC.BFM.ACTIVITY.STATUS |

### BPC.BFM.ACTIVITY.STATUS:



Where:

- activityKind*  
The activity kind, for example, sequence or invoke. The format is: <kind code>-<kind name>
- state*  
The current state of the activity instance in the format: <state code>-<state name>
- bpelId*  
The wpc:id attribute of the activity in the BPEL file. It is unique for activities inside a process model.
- activityTemplateName*  
The name of the activity template.
- activityTemplateId*  
The internal ID of the activity template.
- activityInstanceDescription*  
The description of the activity instance.
- username*  
For BPC.BFM.ACTIVITY.CLAIM this is the user for whom the task has been claimed. For BPC.BFM.ACTIVITY.WISTATUS this is a list users who are associated with the work item.
- principal*  
The name of the user who has claimed the activity.
- escalationName*  
The name of the escalation.
- activityFailedException*  
The exception that caused the activity to fail.
- parallelBranchesStarted*  
The number of branches started.
- message* or *message\_BO*  
The input or the output message for the service as a string or Business Object (BO) representation. The format depends on whether the **Monitor Compatible Events** option was selected on the **Event Monitor** tab in WebSphere Integration Developer.
- current*  
The user name of the current owner of the work item. This is the user whose work item has been transferred away.
- target*  
The user name of the new owner of the work item.

For activity events, the following event correlation sphere identifiers are also written to the Common Base Event as context data elements:

- The *ECSCurrentID* provides the ID of the activity.
- The *ECSParentID* provides the ID of the containing process.

### Activity scope events

The following table describes all activity scope events.

Code	Description	Extension name	Situation	Event nature
42020	Scope started	BPC.BFM.ACTIVITY.STATUS	Start	ENTRY
42021	Scope skipped	BPC.BFM.ACTIVITY.STATUS	Report	SKIPPED
42022	Scope failed	BPC.BFM.ACTIVITY.FAILURE	Fail	FAILED

Code	Description	Extension name	Situation	Event nature
42023	Scope terminating	BPC.BFM.ACTIVITY.STATUS	Report	FAILING
42024	Scope terminated	BPC.BFM.ACTIVITY.STATUS	Stop	TERMINATED
42026	Scope completed	BPC.BFM.ACTIVITY.STATUS	Stop	EXIT
42043	Scope compensating	BPC.BFM.ACTIVITY.STATUS	Report	COMPENSATING
42044	Scope compensated	BPC.BFM.ACTIVITY.STATUS	Stop	COMPENSATED
42045	Scope compensation failed	BPC.BFM.ACTIVITY.STATUS	Fail	COMPFAILED
42048	Scope event received	BPC.BFM.ACTIVITY.STATUS	Report	EV_RECEIVED
42051	Scope event escalated	BPC.BFM.ACTIVITY.ESCALATED	Report	EV_ESCALATED

Activity scope events are a type of activity events, whose syntax is described above for BPC.BFM.ACTIVITY.STATUS.

For activity scope events, the following event correlation sphere identifiers are also written to the Common Base Event as context data elements:

- The ECSCurrentID provides the ID of the scope.
- The ECSParentID provides the ID of the containing process.

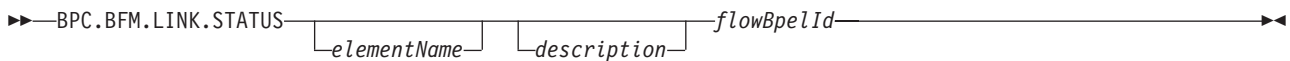
### Link events

The following table describes all link events.

Code	Description	Extension name	Situation	Event nature
21034	Link evaluated true	BPC.BFM.LINK.STATUS	Report	CONDTRUE
42000	Link evaluated false	BPC.BFM.LINK.STATUS	Report	CONDFALSE

The payloads for link events have the following syntax:

### BPC.BFM.LINK.STATUS



Where:

*elementName*

The name of the link.

*description*

The description of the link.

*flowBpelId*

The ID of the flow activity where the link is defined.

For link events, the following event correlation sphere identifiers are also written to the Common Base Event as context data elements:

- The ECSCurrentID provides the ID of the source activity of the link.
- The ECSParentID provides the ID of the containing process.

## Variable events

The following table describes the variable events.

Code	Description	Extension name	Situation	Event nature
21090	Variable update	BPC.BFM.VARIABLE.STATUS	Report	CHANGED

The payloads for variable events have the following syntax:

### BPC.BFM.VARIABLE.STATUS



Where:

*variableName*

The name of the variable.

*variableData* or *variableData\_BO*

If the variable *variableName* has not been initialized, there is no *variableData* or *VariableData\_BO* element. The variable's data contents either a string or Business Object (BO) representation. The format depends on whether the **Monitor Compatible Events** option was selected on the **Event Monitor** tab in WebSphere Integration Developer.

*bpelId* The Business Process Choreographer ID for the variable.

For the variable event, the following event correlation sphere identifiers are written to the Common Base Event as context data elements:

- The ECSCurrentID provides the ID of the containing scope or process.
- The ECSParentID is the ECSCurrentID before the process instance start event of the current process.

## Human task events

Events that are emitted on behalf of human tasks consist of situation-independent data and data that is specific to human task events. The attributes and elements that are specific to human task events are described.

Human task events can have the following categories of event content.

### Event data specific to human tasks

Events are created on behalf of tasks and escalations. The object-specific content of each of these event types is described.

## Tasks

If not specified otherwise, the content is written as extendedDataElements of type string.

Task events have the following object-specific event content.

Attribute	Description
taskTemplateName	The name of the task template from which the instance was derived.
taskTemplateValidFrom	The date from which the template is valid.
taskTemplateId	The identifier of the task template from which the instance is derived.
taskInstanceDescription	Optional: The description of the task instance.
PayloadType	The payload type. The value of the string can be one of: none, digest, or full. The value depends on the setting in WebSphere Integration Developer and whether business object content is written to the event. If an event does not contain a business object, the value is always set to full.

## Escalation

Escalations have the following object-specific event content:

Attribute	Description
taskTemplateName	The name of the task template from which the instance was derived.
taskTemplateValidFrom	The date from which the template is valid.
taskTemplateId	The identifier of the task template from which the instance is derived.
escalationName	The name of the escalation.
escalationInstanceDescription	Optional: The description of the escalation instance.
PayloadType	The payload type. The value of the string can be one of: none, digest, or full. The value depends on the setting in WebSphere Integration Developer and whether business object content is written to the event. If an event does not contain a business object, the value is always set to full.

## Situations in human task events

Human task events can be emitted in different situations. The data for these situations are described in situation elements.

Human task events can contain one of the following situation elements.

Situation name	Content of the Common Base Event	
Start	categoryName is set to StartSituation.	
	situationType	
	Type	StartSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	START_COMPLETED
Stop	categoryName is set to StopSituation.	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	STOP_COMPLETED
Destroy	categoryName is set to DestroySituation.	
	situationType	
	Type	DestroySituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
Fail	categoryName is set to StopSituation.	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	UNSUCCESSFUL
	situationQualifier	STOP_COMPLETED
Report	categoryName is set to ReportSituation.	
	situationType	
	Type	ReportSituation
	reasoningScope	EXTERNAL
	reportCategory	STATUS

## Human task events

Human task events are sent if monitoring is requested for the elements of the task in WebSphere Integration Developer. A list of all the events that can be emitted by human tasks can be found here.

The following types of events can be caused by human tasks:

- “Task events” on page 309
- “Escalation events” on page 310

**Note:** Events are only emitted for ad-hoc tasks if the business relevance flag is set to true in the task model.

## XML syntax

The payloads for human task events have the following syntax:

### BPC.HTM.BASE

▶▶—BPC.HTM.BASE—*HTMEventCode*—| BPC.HTM.TASK.BASE |—————▶▶

### BPC.HTM.TASK.BASE:

|—BPC.HTM.TASK.BASE—*taskTemplateName*—*taskTemplateValidFrom*—*taskTemplateId*—| BPC.HTM.TASK.STATUS | BPC.HTM.ESCALATION.STATUS |

Where:

*HTMEventCode*

The Business Process Choreographer event code that identifies the number of the event type. Possible event codes are listed in the following tables.

*taskTemplateName*

The name of the task template.

*taskTemplateValidFrom*

The date and time from which the task template is valid.

*taskTemplateId*

The ID of the template.

The name of event elements are in uppercase, for example BPC.HTM.BASE, and the names of extended data elements are in mixed case, for example, *HTMEventCode*. Except where indicated, all data elements are of the type string.

## Key to table columns

The **Code** column contains the number of the event. The value is written to the Common Base Event as an extended data element with the name *HTMEventCode*. The columns are as follows:

### Extension name

Contains the string value that is used as the value of the `extensionName` attribute of the Common Base Event.

If WebSphere Business Integration Modeler is used to create the underlying task model, the extension name for events that contain message data in their payload can be extended by a hash character (#) followed by additional characters. These additional characters are used to distinguish Common Base Events that carry different message objects. Events that emit message data also contain additional nested `extendedDataElements` in order to report the contents of the data object. Refer to the documentation for WebSphere Business Integration Modeler for more information.

### Situation

Refers to the situation name of the human task event. For details of situations, see “Situations in human task events” on page 306.

### Event nature

A pointer to the event situation for a business process element in the `EventNature` parameter, as they are displayed in WebSphere Integration Developer.

## Task events

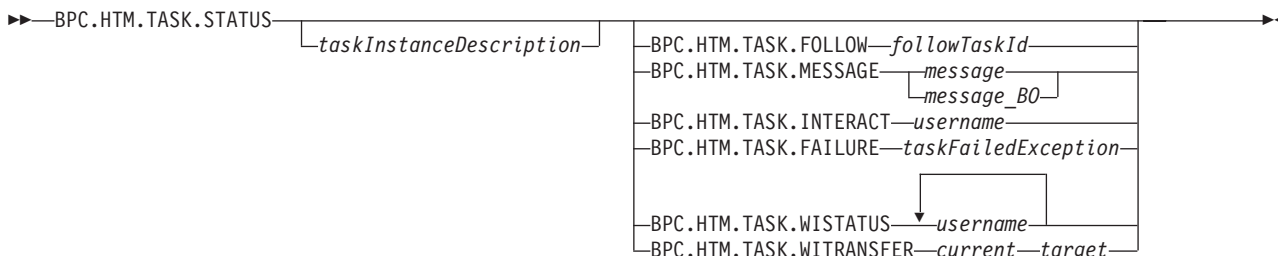
The following table describes all task events.

Code	Description	Extension name	Situation	Event nature
51001	Task created	BPC.HTM.TASK.INTERACT	Report	CREATED
51002	Task deleted	BPC.HTM.TASK.STATUS	Destroy	DELETED
51003	Task started	BPC.HTM.TASK.STATUS	Start	ENTRY
51004	Task completed	BPC.HTM.TASK.STATUS	Stop	EXIT
51005	Claim canceled	BPC.HTM.TASK.STATUS	Report	DEASSIGNED
51006	Task claimed	BPC.HTM.TASK.INTERACT	Report	ASSIGNED
51007	Task terminated	BPC.HTM.TASK.STATUS	Stop	TERMINATED
51008	Task failed	BPC.HTM.TASK.FAILURE	Fail	FAILED
51009	Task expired	BPC.HTM.TASK.STATUS	Report	EXPIRED
51010	Waiting for subtasks	BPC.HTM.TASK.STATUS	Report	WAITFORSUBTASK
51011	Subtasks completed	BPC.HTM.TASK.STATUS	Stop	SUBTASKCOMPLETED
51012	Task restarted	BPC.HTM.TASK.STATUS	Report	RESTARTED
51013	Task suspended	BPC.HTM.TASK.STATUS	Report	SUSPENDED
51014	Task resumed	BPC.HTM.TASK.STATUS	Report	RESUMED
51015	Task completed and follow-on task started	BPC.HTM.TASK.FOLLOW	Report	COMPLETEDFOLLOW
51101	Task properties updated	BPC.HTM.TASK.STATUS	Report	UPDATED
51103	Output message updated	BPC.HTM.TASK.MESSAGE	Report	OUTPUTSET
51104	Fault message updated	BPC.HTM.TASK.MESSAGE	Report	FAULTSET
51201	Work item deleted	BPC.HTM.TASK.WISTATUS	Destroy	WI_DELETED
51202	Work items created	BPC.HTM.TASK.WISTATUS	Report	WI_CREATED
51204	Work item transferred	BPC.HTM.TASK.WITRANSFER	Report	WI_TRANSFERRED
51205	Work items refreshed	BPC.HTM.TASK.WISTATUS	Report	WI_REFRESHED

The payloads for task events have the following syntax:



## BPC.HTM.TASK.STATUS



Where:

*taskInstanceDescription*

The description of the task.

*followTaskId*

The ID of the task that was started as a follow-on-task.

*message* or *message\_BO*

A string or business object representation that contains the input or output message. The format depends on whether the **Monitor Compatible Events** option was selected on the **Event Monitor** tab in WebSphere Integration Developer.

*taskFailedException*

A string containing the *faultNameSpace* and *faultName* separated by a semicolon (;).

*username*

For BPC.HTM.TASK.INTERACT this is the name of the user associated with the task. For BPC.BPC.TASK.WISTATUS this is a list of users whose work item was created or deleted.

*current*

The name of the current user. This is the user whose work item has been transferred away.

*target* The user name of the work item receiver.

For task events, the following identifiers of event correlation spheres are written as the context data elements to the Common Base Event:

- The ESCcurrentID provides the ID of the task instance.
- The ECSParentID is the ECSCurrentID before the task instance event.

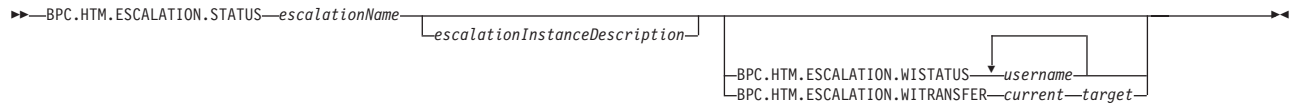
### Escalation events

The following table describes all task escalation events.

Code	Description	Extension name	Situation
53001	Escalation fired	BPC.HTM.ESCALATION. STATUS	Report
53201	Work item deleted	BPC.HTM.ESCALATION. WISTATUS	Destroy
53202	Work item created	BPC.HTM.ESCALATION. WISTATUS	Report
53204	Escalation transferred	BPC.HTM.ESCALATION. WITRANSFER	Report
53205	Work item refreshed	BPC.HTM.ESCALATION. WISTATUS	Report

The payloads for escalation events have the following syntax:

### **BPC.HTM.ESCALATION.STATUS**



Where:

*EscalationName*

The name of the escalation.

*escalationInstanceDescription*

The description of the escalation.

*username*

This is a list of users whose work item was escalated.

*current*

The name of the current user. This is the user whose work item has been transferred away.

*target* The user name of the work item receiver.

For task events, the following identifiers of event correlation spheres are written as the context data elements to the Common Base Event:

- The ESCcurrentID provides the ID of the escalation.
- The ECSParentID provides the ID of the associated task instance.



---

## Chapter 8. Tuning business processes

Use this task to improve the performance of business processes.

After successfully running business processes, you can perform this task to improve performance.

1. Define how to measure the baseline performance, and which measurements you want to optimize.

For example, for some business applications, it is preferable to reduce the response time for end-users under peak-load conditions. For other applications, the rate that the system can process transactions might be more important than the actual duration of each transaction.

2. Make baseline measurements.

Make the baseline measurements under conditions of load, time-of-day, and day-of-week that are appropriate for tuning your application. Normally, the most important baseline measurements are the throughput and response times. Throughput values are measured after a specific bottleneck capacity is reached, for example 100% CPU load, disk I/O at maximum, or network I/O at 100%. Reliable response time values are best measured for a single process instance during low server utilization.

3. Tune the processes.

Depending on whether your application uses long-running processes or microflows, perform one of the following steps:

- To tune long-running processes, perform the steps that are described in “Tuning long-running processes” on page 314. These processes tend to run for a long time, but can be interrupted by events or human interaction. Their performance therefore depends on the performance of the Business Process Choreographer database and the messaging service.
- To tune microflows, perform the steps that are described in “Tuning microflows” on page 321. These processes tend to run for only a short time. They use the database only for audit logging, if enabled, and to retrieve the template information. They do not use messaging support for storing persistent data. These processes involve no human interaction.

4. Tune the application.

Many different options are available to achieve the same functionality in an application, and some of them are more efficient than others. Identify and review any performance-critical code. Maximize asynchronicity, and ensure that actions are not unnecessarily serialized. Try to minimize the amount and complexity of data submitted to the process, as serialization/deserialization costs are directly related to the size and complexity of data objects used in the process. Consider shortening timeouts that do not result in error conditions. Identify opportunities to cache the results of database queries.

5. Review the current configuration for performance bottlenecks that can be eliminated.

Possibilities to consider include:

- Installing more processors, more memory, and faster disks.
- Storing the database logs on different physical disks from the data, and distributing the data on several disks.
- Using DB2, rather than Cloudscape, for optimal performance.

6. Repeat the benchmark measurements under similar load conditions to those of the baseline measurements.  
Keep a permanent record of the application performance measurements to objectively measure any future changes in performance.

The business processes are configured to run measurably faster.

---

## Tuning long-running processes

Use this task to improve the performance of long-running business processes.

Long-running processes can include user-interaction, asynchronous invocations, multiple receives, picks, and event handlers, for example; they use database and messaging subsystems for storing persistent states. The following topics describe how to improve the performance of long-running processes.

### Specifying initial database settings

Use this task to specify initial DB2 database settings. Note that this information is provided only as an example.

**Attention:** The following information relates to the Business Process Choreographer database. For information about tuning a WebSphere default messaging database, see *Tuning and problem solving for messaging engine data stores in the WebSphere Application Server for z/OS information center*.

For additional information on creating databases in WebSphere Process Server for z/OS see *Considerations for creating the database* and *Creating databases and storage groups* in the *Installing and configuring WebSphere Process Server for z/OS PDF*.

To achieve good database operation, specify the initial database settings. You will fine-tune the settings later, in “Fine-tuning the database” on page 319.

1. Separate the log files from the data files.

Putting the database log file on a disk drive that is separate from the data tends to improve performance, provided that sufficient disk drives are available. If few disk drives are available, distributing the table spaces, as described in the previous section, is usually more beneficial than putting the database log on a separate drive.

For example, if you use DB2 on a Windows system, you can change the location of the log files for the database named BPEDB to the F:\db2logs directory, by entering the following command:

```
db2 UPDATE DB CFG FOR BPEDB USING NEWLOGPATH F:\db2logs
```

2. Create table spaces.

After you create the database, explicitly create table spaces. Example scripts to create table spaces are provided by Business Process Choreographer in the ProcessChoreographer subdirectory of your WebSphere Application Server installation. Customize these scripts to accommodate the needs of a particular scenario. Your goal, when creating the table spaces, is to distribute input and output operations over as many disk drives as possible that are available to DB2. By default, these scripts create the following table spaces:

#### AUDITLOG

Contains the audit trail tables for processes and tasks. Depending on

the degree of auditing that is used, access to tables in this table space can be significant. If auditing is turned off, tables in this table space are not accessed.

### **COMP**

Contains the compensation tables for business processes from Business Process Choreographer Version 5. Depending on the percentage of compensable processes and activities, the tables in this table space might require high disk bandwidth. If compensation is not used within business processes, the tables in this table space are not used.

### **INSTANCE**

Holds the process instance and task tables. It is always used intensively, regardless of the kind of long-running process that is run. Where possible, spread this table space over several disk drives.

### **SCHEDTS**

Contains the tables that are used by the WebSphere scheduling component. Access to tables in the scheduler table space is usually low, because of the caching mechanisms used in the scheduler.

### **STAFFQRY**

Contains the tables that are used to temporarily store staff query results that are obtained from staff registries like Lightweight Directory Access Protocol (LDAP). When business processes contain many person activities, tables in this table space are frequently accessed.

### **TEMPLATE**

Contains the tables that store the template information for processes and tasks. The tables are populated during the deployment of an application. At run time the access rate is low. The data is not updated, and only new data is inserted during deployment.

### **WORKITEM**

Holds the tables that are required for work item processing. Work items are used for human task interaction. Depending on the number of human tasks in the business processes, access to the tables in this table space can vary from a low access rate to significantly high access rate. The access rate is not zero, even when no explicit human tasks are used, because work items are also generated to support administration of long-running processes.

To create a database for high performance, perform the following actions:

- a. Create the database.
- b. Create the table spaces on the desired disks.

For example, the following script is based on the createTablespaceDb2.ddl file that is located in the ProcessChoreographer subdirectory of your WebSphere Application Server installation. It creates table spaces using three different disk drives on a Windows system:

```
-- Scriptfile to create tablespaces for DB2 UDB
-- Replace occurrence of @location@ in this file with the location
-- where you want the tablespace containers to be stored, then run:
-- db2 connect to BPEDB
-- db2 -tf createTablespaceDb2.ddl
```

```
CREATE TABLESPACE TEMPLATE MANAGED BY SYSTEM USING('D:/BPE/TEMPLATE');
```

```
CREATE TABLESPACE STAFFQRY MANAGED BY SYSTEM USING('D:/BPE/STAFFQRY');
```

```

CREATE TABLESPACE AUDITLOG MANAGED BY SYSTEM USING('E:/BPE/AUDITLOG');
CREATE TABLESPACE COMP MANAGED BY SYSTEM USING('D:/BPE/COMP');
CREATE TABLESPACE INSTANCE MANAGED BY SYSTEM USING('D:/BPE/INSTANCE', 'E:/BPE/INSTANCE');
CREATE TABLESPACE WORKITEM MANAGED BY SYSTEM USING('F:/BPE/WORKITEM');
CREATE TABLESPACE SCHEDTS MANAGED BY SYSTEM USING(' F:/BPE/SCHEDTS');

```

Notice how the INSTANCE table space is distributed across two drives.

c. Create the tables.

Create Business Process Choreographer tables by running the script provided for the respective database. For DB2, for example, use the createSchemaDb2.ddl file in the ProcessChoreographer directory.

3. Tune the database.

Use a capacity planning tool for your initial database settings.

If you are using DB2, start the DB2 configuration advisor from the DB2 Control Center, by selecting **DB2 configuration advisor** from the pop-up menu of the Business Process Choreographer database. Do the following actions:

a. Allocate memory to DB2.

For **Server**, allocate to DB2 only as much memory as is physically available to it without swapping.

b. Specify the type of workload.

For **Workload**, select **Mixed** (queries and transactions).

c. For **Transactions**, specify the length of the transactions and the estimated number of transactions to be processed each minute.

Select **More than 10**, to indicate that long transactions are used.

Then, in the **Transactions per minute** field, select the estimated number of transactions processed each minute. To determine this number, assume that each activity in the process has one transaction. The number of transactions performed in one minute is then as follows:

*number of transactions performed each minute = number of processes completed each minute \* number of activities in each process*

d. Tune the database for faster transaction performance and slower recovery.

For **Priority**, select **Faster transaction performance**.

e. If possible, tune the database populated with the typical amount of data in production. For **Populated**, select **Yes**. Otherwise, select **No**.

f. Tune the parallel connections setting.

For **Connections**, specify the maximum number of parallel connections that can be made to the application server. Guidelines for determining this value are as follows:

- The number of database connections required is determined by the number of Java DataBase Connectivity (JDBC) connections to the WebSphere Application Server. The JDBC connections are provided by the JDBC connection pool, which is in the WebSphere Application Server. For  $p$  JDBC connections,  $p * 1.1$  database connections are required. How to estimate a realistic value for  $p$  is described in "Tuning the application server" on page 318.
- If Business Process Choreographer and the database are installed on the same physical server, Business Process Choreographer needs no remote

database connections. However, because remote connections might be required for remote database management, specify a low value, rather than zero.

- If Business Process Choreographer and DB2 are installed on separate servers, set the number of remote applications in accordance with the rule previously described for local connections.

g. For **Isolation**, select **Read stability**. This isolation level is required for Business Process Choreographer.

The configuration advisor displays suggested changes. You can either apply the changes now, or save them to a file to apply later.

Your long-running processes are running as fast as possible under the current environment and loading conditions.

## Planning messaging engine settings

Use this task to plan your initial settings for the messaging engine.

To achieve the best performance for long-running processes, tune the message queuing system for maximum performance of persistent messages.

If you use WebSphere Platform Messaging, follow the instructions given in Service integration in the WebSphere Application Server for z/OS information center, to set up and tune the data stores for the messaging engines.

If you use the IBM WebSphere MQ messaging product, rather than the default messaging services, complete the following steps.

1. Tune MQ parameter settings.

Tune the following MQ parameter settings:

- Log file pages
- Log buffer page
- Log primary files
- Log secondary files
- Log default path
- Maximum channels
- Channel application bind type

The default locations for both the persistent queue data and the MQ logs is the MQ installation directory. Put the data storage for the persistent queues and the WebSphere MQ logs on different disk drives. By changing the path to the log file to refer to another disk drive, you can change the location for the MQ logs. Make these changes before you create the queue managers for Business Process Choreographer.

2. Tune WebSphere MQ service properties for Business Process Choreographer.

These values must be set before you create the queue managers that are used by Business Process Choreographer. Set each parameter to its maximum value, as shown in the following table:



Table 39. Tuning WebSphere MQ service properties for Business Process Choreographer

Parameter	Value	Comment
Log file pages	16384	On Windows systems, not all versions of WebSphere MQ support setting the number of log file pages to 16384 by using the MQ administration tools. In this case, change the value of the Windows registry key:  HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\ CurrentVersion\Configuration\LogDefaults  to:  16384
Log primary files	10	
Log secondary files	53	
Log buffer pages	512	

3. Tune the queue manager properties.

Specify the queue manager properties for the maximum number of channels and the type of binding for the channel application, as shown in the following table:

Table 40. Tuning queue manager properties

Queue manager properties	Value
Maximum channels	Use the default
Channel application bind type	FASTPATH

Your queue manager is operating optimally.

## Tuning the application server

Use this task to tune the application server.

Before you start this task, you must have specified the initial settings for the database, as described in “Specifying initial database settings” on page 314.

To ensure that the business process container can perform optimally, you need to adjust the server settings.

1. Estimate the application server resources that you need for each business process container.
  - a. One data source to read and write business process state information to a database: BPEDataSourceDb2 in the server scope DB2 Universal JDBC Driver Provider (XA)
  - b. Calculate the maximum concurrency of transactions,  $t$ , for the process navigation by adding the following:
    - The maximum number of clients concurrently connected through the Business Process Choreographer API
    - The number of concurrent endpoints defined in the JMS activation specification BPEInternalActivationSpec

- The number of concurrent endpoints defined in the JMS activation specification `HTMInternalActivationSpec`

**Note:** To view the activation specifications for the process server, in the administrative console, click **Resources** → **JMS Providers** → **Default messaging** → **JMS activation specification**.

- Calculate the number of parallel JDBC connections required to the process server database,  $p = 1.1 * t$

**Note:** The value of  $p$  must not be greater than the number of connections allowed by the database.

- Calculate the number of parallel JDBC connections required to the messaging database,  $m = t + x$ , where  $x$  is the number of additional JMS sessions to allow for overload situations where additional messages are generated and must be served
  - Set the SQL Statement cache size to 30
- Tune the JDBC provider settings for the process server database (BPEDB).
    - Set **Max Connections** to the value  $p$ . The value of  $p$  must not be greater than the number of connections allowed by the database.
    - Set the **SQL Statement cache size** to 300.
  - Tune the JDBC provider settings for the messaging database. Set **Max Connections** to the value  $m$ .
  - Tune the heap size.
 

Here are some guidelines for the size of the server heap:

    - 256 MB is too low, and results in poor performance.
    - 512 MB is adequate as an initial heap size for many systems.
    - 1024 MB is a reasonable upper limit.
  - Tune any services that are used by your business processes. Make sure that your supporting services are tuned to cope with the degree of concurrency and load demands that Business Process Choreographer makes on the service.

The application server is tuned for improved performance.

## Fine-tuning the messaging provider

Use this task to improve the performance of your messaging provider.

If you use WebSphere Platform Messaging, refer to Tuning and problem solving for messaging engine data stores.

The performance of your messaging provider is improved.

## Fine-tuning the database

Use this task to fine-tune the database.

The business process container and business processes must be running.

A common problem is that the database runs out of lock list space, resulting in lock escalation, which severely impacts performance. Depending on the structure of the business processes run, you might therefore need to customize the settings of certain performance-related parameters in your database management system.

**Note:** If you are not using DB2, refer to your database management system documentation for information about monitoring the performance of the database, identifying and eliminating bottlenecks, and fine-tuning its performance. The rest of this topic offers advice for DB2 users.

1. Tune the lock list space, to help ensure optimum performance.

Check the db2diag.log file for your DB2 instance. Look for entries like the following example:

```
2005-07-24-15.53.42.078000 Instance:DB2 Node:000
PID:2352(db2syscs.exe) TID:4360 Appid:*LOCAL.DB2.027785142343
data management sqlEscalateLocks Probe:4 Database:BPEDB
```

```
ADM5503E The escalation of "10" locks on table "DB2ADMIN.ACTIVITY_INSTANCE_B_T"
to lock intent "X" has failed. The SQLCODE is "-911".
```

This type of message indicates that the parallelism for business process applications has improved to the point where the number of available locks is now too small. Increase the LOCKLIST value to approximately  $10 * p$ , where  $p$  is your estimate for the maximum number of parallel JDBC connections that are required at any time. For example, if you sized your Business Process Choreographer database, BPEDB, with a value of  $p=50$ , enter the following command:

```
db2 UPDATE DB CFG FOR BPEDB USING LOCKLIST 500
```

2. If you used the DB2 configuration advisor, your database throughput is probably pretty good. You can, however, further improve the performance, in the following ways:

- Follow the best practices for database tuning that are described in the DB2 online documentation, books, and articles.
- Use DB2 monitors, and examine the db2diag.log file for more information about bottlenecks within the database.
- Regularly run runstats on your database.
- Tune the following DB2 parameters:

#### **LOCKLIST**

See the description in step 1.

#### **AVG\_APPLS**

It is better to set this parameter too high rather than too low. For example, if there are a maximum of 20 connected applications, set AVG\_APPLS to 50.

#### **LOGBUFSZ**

Increasing the size of the buffer for the DB2 log decreases how often a full log buffer must be written to disk.

#### **LOG\_FILSIZ**

Increasing the size of the log files reduces how often they are switched.

3. Adjust database and database manager settings according to workload requirements. After the configuration advisor has configured the database, you can also tune the following settings:

#### **MINCOMMIT**

A value of 1 is strongly recommended. The DB2 Configuration Advisor may suggest other values.

#### **NUM\_IOSERVERS**

Must match the number of physical disks that the database resides on. You should have at least as many IOSERVERS as you have disks.

IOSERVERs do not use many system resources, so it is better to set a value that is too high rather than too low.

Your long-running processes are running as fast as possible under the current environment and loading conditions.

---

## Tuning microflows

Use this task to improve the performance of microflows.

Microflows run in memory, without any user-interaction or persistent messaging support. Database access is required only if audit logging or Common Event Infrastructure (CEI) are enabled for the microflow. The processing of a microflow occurs in a single thread, and normally, in a single transaction. The performance of microflows mainly depends on the services called. However, if the memory available for the server is too small, the performance of microflows will be reduced.

1. Tune the Java Virtual Machine (JVM) heap size.

By increasing the Java heap size, you can improve the throughput of microflows, because a larger heap size reduces the number of garbage collection cycles that are required. Keep the value low enough to avoid heap swapping to disk. For guidelines on the size of the server heap, see the relevant step in “Tuning the application server” on page 318.

2. Tune the JVM garbage collection. Using the Throughput Garbage Collector achieves the best throughput, however the garbage collection pauses can be 100-1000 ms, depending on the heap size. If response time is more important than throughput, use the Low Pause Garbage Collector.
3. Make sure that there are enough database connections. You need at least one JDBC connection to the process database for each concurrently running microflow. It must be enough, not only for the connections to the database itself, but also connections in the connection pool of the data source.
4. Tune the Object Request Broker (ORB) thread pool size. If remote clients connect to the server-side ORB, make sure that there are enough threads available in the ORB Thread Pool.
5. Tune the default thread pool size. To increase the number of microflows that can run concurrently, you must increase the default thread pool size. To change the value, using the Administrative Console, click **Application Servers** → *server* → **Thread pools** → **Default**.

Your microflows are running as fast as possible under the current environment and loading conditions.

---

## Tuning business processes that contain human tasks

There are various ways to improve the performance of business processes that contain human tasks.

The following topics describe how to tune business processes that contain human tasks.

## Reduce concurrent access to human tasks

When two or more people try to claim the same human task, only one person will succeed. The other person is denied access.

Only one person can claim a human task. If several people attempt to work with the same human task at the same time, the probability of collision increases. Collisions cause delays, because of lock waits on the database or rollbacks. Some ways to avoid or reduce the incidence of collision are as follows:

- If concurrent access is high, limit the number of users who can access a particular human task.
- Avoid unnecessary human task queries from clients, by using intelligent claim mechanisms. For example, you might take one of the following steps:
  - Try to claim another item from the list if the first claim is unsuccessful.
  - Always claim a random human task.
  - Reduce the number of potential owners for the task, for example, by assigning the task to a group with fewer members.
  - Limit the size of the task list by specifying a threshold on the query used to retrieve the list. Also consider using filtering to limit the number of hits. You can filter for properties of a task, for example, only showing tasks with priority one or tasks that are due within 24 hours from now. For an inline task, you can also filter for business data that is associated with the task using custom properties or query properties. To perform such filtering, you must specify an appropriate where clause on the query that retrieves the task list.
  - Minimize or avoid dynamic staff queries, that is, ones that use variables.
  - Use a client caching mechanism for human task queries, to avoid running several queries at the same time.

## Reduce query response time

Reduce the time that the database takes to respond to queries.

When you use a custom client, make sure that the queries set a threshold. From a usability viewpoint, retrieving hundreds or thousands of items is typically undesirable, because the larger the number of database operations, the longer the task takes to complete, and because a person can manage only a small number of results at a time. By specifying a threshold, you minimize database load and network traffic, and help to ensure that the client can present the data quickly.

A better way to handle a query that returns a large number of items might be to rewrite the query, to return a smaller result set of items. You can do this by querying work items for only a certain process instance or work items with only a certain date.

You can also reduce the query result by using filter criteria.

## Avoid scanning whole tables

When you use the query application programming interfaces (APIs), to list the objects in the database, you can specify filters that narrow the results you want to retrieve. In these filters, you can specify the values and ranges of object attributes.

When database queries are processed, the filter information is translated into WHERE clauses in a Structured Query Language (SQL) statement. These WHERE clauses map the object attributes to column names in the affected database tables.

If your query specifies a filter that does not translate to an indexed table column, the SQL statement will probably cause the table to be scanned. This scanning impacts performance negatively and increases the risk of deadlocks. Although this performance impact can be tolerated if it happens only a few times a day, it could adversely affect efficiency if it took place several times a minute.

In such circumstances, a custom index can dramatically reduce the impact. In a real customer situation, a custom index helped to reduce the API response time from 25 seconds to 300 milliseconds. Instead of reading 724 000 rows of the database table, only six rows had to be read.

Depending on the filter criteria that you specify, some columns might not be included in an index. If this is the case, and if a table scan is used, resulting in slow query performance, check the access path of the statement, using DB2 Explain, for example. If necessary, define a new index.



---

## Chapter 9. Troubleshooting Business Process Choreographer

---

### Troubleshooting the Business Process Choreographer configuration

Use this topic to solve problems relating to the configuration of the business process container, or the human task container.

The purpose of this section is to aid you in understanding why the configuration of your business process container or human task container is not working as expected and to help you resolve the problem. The following tasks focus on problem determination and finding solutions to problems that might occur during the configuration of the business process container or the human task container.

#### Business Process Choreographer log files

This describes where to find the log files for your Business Process Choreographer configuration.

##### Profile creation

The profile actions for Business Process Choreographer write to the `bpcaugment.log` file in the logs directory.

If you select the sample configuration option in the profile wizard, it invokes the `bpeconfig.jacl` script, and actions are logged in the `bpeconfig.log` file in the logs directory.

##### Administrative scripts

All of the Business Process Choreographer scripts that are run using `wsadmin` are logged in the `wsadmin.traceout` file. However, because this file is overwritten each time that `wsadmin` is invoked, make sure that you save this log file before invoking `wsadmin` again.

##### Configuration-related scripts

The script files `bpeconfig.jacl`, `taskconfig.jacl`, `clientconfig.jacl`, and `bpeunconfig.jacl` write their log files in the logs directory with the names `bpeconfig.log`, `taskconfig.log`, `clientconfig.log`, and `bpeunconfig.log`. The configuration scripts `setUpEventCollector.bat` (.sh on Unix systems) and `setupObserver.bat` (.sh on Unix systems) write their log files in the logs directory to the files `setupObserver.log` and `setupEventCollector.log`, respectively. Also check the `wsadmin.traceout` file.

##### Administrative utility scripts

The administrative scripts in the `util` subdirectory of the `ProcessChoreographer` directory do not write their own log files. Check the `wsadmin.traceout` file and the application server log files.

##### Configuration checker

The `bpecheck.jacl` script file, found in the `ProcessChoreographer/config` directory, can be used to check for common configuration problems. The results are written to the `bpecheck.log` file in the logs directory.



## Enabling tracing for Business Process Choreographer

This describes what to do before contacting support.

### Enabling tracing

Business Process Choreographer tracing uses the standard WebSphere Process Server tracing mechanism. This must be enabled in the normal way.

The trace specification is as follows:

```
=info:com.ibm.bpe.=all:com.ibm.task.*=all:com.ibm.ws.staffsupport.*=all
```

where `com.ibm.bpe.*=all` traces business processes and `com.ibm.task.*=all` traces human tasks. The remaining aspects of human tasks, the staff plug-ins, are traced by `com.ibm.ws.staffsupport`.

### What to send support

After enabling tracing, recreate your problem scenario then provide the following files:

- SystemOut.log
- SystemErr.log
- trace.log

These files are located in `install_root/profiles/profile_name/logs/`

The WebSphere Application Server FFDC log, located in the `ffdc` folder, also contains information helpful to support in the absence of a trace.

## The task container application fails to start

Startup bean named `ejb/htm/TaskContainerStartUpBean` forced the application to stop.

### Symptom

The following errors are written to the SystemOut.log file:

```
WSVR0037I: Starting EJB jar: taskejb.jar
NMSV0605W: A Reference object looked up from the context "java:"
with the name "comp/env/scheduler/DefaultUserCalendarHome"
was sent to the JNDI Naming Manager and an exception resulted.
Reference data follows:
Reference Factory Class Name: com.ibm.ws.naming.util.IndirectJndiLookupObjectFactory
Reference Factory Class Location URLs:
Reference Class Name: java.lang.Object
Type: JndiLookupInfo
Content: JndiLookupInfo:
jndiName="com/ibm/websphere/scheduler/calendar/DefaultUserCalendarHome";
providerURL=""; initialContextFactory=""
:
StartBeanInfo E STUP0005E: Startup bean named ejb/htm/TaskContainerStartUpBean
forced application to stop.
ApplicationMg W WSVR0101W: An error occurred starting, TaskContainer_utxt1b10Node01_server1
ApplicationMg A WSVR0217I: Stopping application: TaskContainer_utxt1b10Node01_server1
EJBContainerI I WSVR0041I: Stopping EJB jar: taskejb.jar
```

## Reason

You get this error if the SchedulerCalendars application is not available when the TaskContainer application starts.

## Resolution

Either install the SchedulerCalendars application manually, or if it is already installed, add a new target mapping for it.

In a default profile, the SchedulerCalendars application is available automatically as a WebSphere system application. However, in a custom profile it is not available automatically.

The bpeconfig.jacl script tries to install the SchedulerCalendars application, but this is not always possible.

If you use the administrative console install wizard to configure Business Process Choreographer in an ND environment, you must install the SchedulerCalendars application manually.

## Troubleshooting the Business Process Choreographer database and data source

Use this task to solve problems with the Business Process Choreographer database and data source.

Both the business process container and the human-task container need a database. Without the database, enterprise applications that contain business processes and human tasks will not work.

- If you are using DB2:
  - If you use the DB2 Universal JDBC driver type 4 and get DB2 internal errors such as "com.ibm.db2.jcc.a.re: XAER\_RMERR : The DDM parameter value is not supported. DDM parameter code point having unsupported value : 0x113f DB2ConnectionCorrelator: NF000001.PA0C.051117223022" when you test the connection on the Business Process Choreographer data source or when the server starts up, perform the following actions:
    1. Check the class path settings for the data source. In a default setup the WebSphere variable `${DB2UNIVERSAL_JDBC_DRIVER_PATH}` can point to the WebSphere Process Server embedded DB2 Universal JDBC driver which is found in the `universalDriver_wbi` directory.
    2. The version of the driver might not be compatible with your DB2 server version. Make sure that you use the original `db2jcc.jar` files from your database installation, and not the WebSphere Process Server embedded DB2 Universal JDBC driver. If required, changed the value of the WebSphere variable `${DB2UNIVERSAL_JDBC_DRIVER_PATH}` to point to your original `db2jcc.jar` file.
    3. Restart the server.
  - If the `db2diag.log` file of your DB2 instance contains messages like `ADM5503E` as illustrated below:

```
2004-06-25-15.53.42.078000 Instance:DB2 Node:000
PID:2352(db2syscs.exe) TID:4360 Appid:*LOCAL.DB2.027785142343
data management sqlEscalateLocks Probe:4 Database:BPEDB
```

```
ADM5503E The escalation of "10" locks on table "GRAALFS .ACTIVITY_INSTANCE_I"
to lock intent "X" has failed. The SQLCODE is "-911"
```

Increase the LOCKLIST value. For example to set the value to 500, enter the following DB2 command:

```
db2 UPDATE DB CFG FOR BPEDB USING LOCKLIST 500
```

This can improve performance significantly.

- To avoid deadlocks, make sure your database system is configured to use sufficient memory, especially for the bufferpool. For DB2, use the DB2 Configuration Advisor to determine reasonable values for your configuration.
- If you get errors mentioning the data source implementation class `COM.ibm.db2.jdbc.DB2XADataSource`:
  - Check that all WebSphere environment variables that are used in the `server.policy` file, have been set correctly. For example, `DB2_INSTALL_ROOT` and `DB2_JDBC_DRIVER_PATH`.
  - Check that the class path definition for your JDBC provider is correct, and that it does not have two entries.
  - Check that the component-managed authentication alias is set to `cellName/BPEAuthDataAliasDbType_Scope`. Where, `cellName` is the name of the cell, `DbType` is the database type, and `Scope` is the scope of the definition.
- If you are using a remote DB2 for z/OS database, and you get SQL code 30090N in the `SystemOut.log` file when the application server attempts to start the first XA transaction with the remote database, perform the following:
  - Make sure that the instance configuration variable `SPM_NAME` points to the local server with a host name not longer than eight characters. If the host name is longer than eight characters, define a short alias in the `etc/hosts` file.
  - Otherwise, you might have invalid syncpoint manager log entries in the `sqllib/spmlog` directory. Try clearing the entries in the `sqllib/spmlog` directory and restart.
  - Consider increasing the value of `SPM_LOG_FILE_SZ`.
- If you are using Cloudscape:
  - If you get a "Too many open files" error on Linux or UNIX systems, increase the number of file handles available, for example, to 4000 or more. For more information about how to increase the number of available file handles, refer to the documentation for your operating system.
  - If you get a "Java class not found" exception when trying to invoke Cloudscape tools, make sure that you have set up the Java environment, and that your `classpath` environment variable includes the following JAR files:
    - `db2j.jar`
    - `db2jtools.jar`
    - `db2jcc.jar`
    - `db2jcvview.jar`
  - If you cannot connect to your Cloudscape database using the Cloudscape tools (like `ij` or `cvview`) and you get the following exception:

```
ERROR XJ040: Failed to start database 'c:\WebSphere\AppServer\profiles\profile_name\databases\BPEDB',
see the next exception for details.
ERROR XSDB6: Another instance of Cloudscape may have already booted the database
c:\WebSphere\AppServer\profiles\profile_name\databases\BPEDB.
```

you must stop your WebSphere Application Server before using these tools because only one application can access the Cloudscape database at a time.

- If you get a database error when installing an enterprise application that contains a business process or human task. When an enterprise application is

installed, any process templates and task templates are written into the Business Process Choreographer database. Make sure that the database system used by the business process container is running and accessible.

- If you have problems using national characters. Make sure that your database was created with support for Unicode character sets.
- 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 tables (or to run the scripts to create them).

---

## Troubleshooting business process and human tasks

Use this topic to solve problems relating to business processes and human tasks.

The following tasks focus on troubleshooting problems that can happen during the execution of a business process or task.

### Troubleshooting the installation of business process and human task applications

When installing an application containing business processes, human tasks, or both in an ND environment, you get an exception in the deployment manager SystemErr.log file

#### Symptom

When installing an application containing business processes, human tasks, or both in an ND environment, you find the following exception in the deployment manager SystemErr.log file:

```
SystemErr R com.ibm.ws.management.commands.sib.SIBAdminCommandException:
CWSJA0012E: Messaging engine not found.
at com.ibm.ws.management.commands.sib.SIBAdminCommandHelper.createDestination
(SIBAdminCommandHelper.java:787)
at com.ibm.ws.management.commands.sib.CreateSIBDestinationCommand.afterStepsExecuted
(CreateSIBDestinationCommand.java:459)
at com.ibm.websphere.management.cmdframework.provider.AbstractTaskCommand.execute
(AbstractTaskCommand.java:547)
at com.ibm.ws.sca.internal.deployment.sib.SIBAdminHelper.call(SIBAdminHelper.java:136)
at com.ibm.ws.sca.internal.deployment.sib.SIBAdminHelper.createSIBDestination
(SIBAdminHelper.java:112)
at com.ibm.ws.sca.internal.deployment.sib.SIBAdmin.createDestination(SIBAdmin.java:327)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.createDestination
(SIBDestinationTask.java:263)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.preInstallModule
(SIBDestinationTask.java:71)
at com.ibm.ws.sca.internal.deployment.SCATaskBase.installModule(SCATaskBase.java:57)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.processArtifacts
(SIBDestinationTask.java:228)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.install
(SIBDestinationTask.java:287)
at com.ibm.ws.sca.internal.deployment.SCAInstallTask.performInstallTasks
(SCAInstallTask.java:116)
at com.ibm.ws.sca.internal.deployment.SCAInstallTask.performTask
(SCAInstallTask.java:61)
at com.ibm.ws.management.application.SchedulerImpl.run(SchedulerImpl.java:253)
at java.lang.Thread.run(Thread.java:568)
```

#### Reason

The bus member for the "SCA.SYSTEM.cellName.Bus" bus is missing.

## Resolution

In the administrative console, click **Service Integration** → **Buses** → **SCA.SYSTEM.cellName.Bus**. In the Topology section, click **Bus members**. Add the server or cluster where you want to install the business process or human task application as a bus member, then restart the affected server or cluster and try installing the application again.

## Troubleshooting the execution of business processes

This describes the solutions to common problems with business process execution.

In Business Process Choreographer Explorer, you can search for error message codes on the IBM technical support pages.

1. On the error page, click the **Search for more information** link. This starts a search for the error code on the IBM technical support site. This site only provides information in English.
2. Copy the error message code that is shown on the error page to the clipboard. The error code has the format CWWBcnnnnc, where each c is a character and nnnn is a 4-digit number. Go to the WebSphere Process Server technical support page.
3. Paste the error code into the **Additional search terms** field and click **Go**.

Solutions to specific problems are in the following topics.

### **ClassCastException when stopping an application containing a microflow**

The SystemOut.log file contains ClassCastException exceptions around the time when an application containing a microflow had been stopped.

#### **Reason**

When an application is stopped, the classes contained in the EAR file are removed from the class path. However, microflow instances may still be executing that need these classes.

#### **Resolution**

Perform the following actions:

1. Stop the microflow process template first. From now on, it is not possible to start new microflow instances from that template.
2. Wait for at least the maximum duration of the microflow execution so that any running instances can complete.
3. Stop the application.

### **XPath query returns an unexpected value from an array**

Using an XPath query to access a member in an array returns an unexpected value.

#### **Reason**

A common cause for this problem is assuming that the first element in the array has an index value of zero. In XPath queries in arrays, the first element has the index value one.

## Resolution

Check that your use of index values into arrays start with element one.

### **An activity has stopped because of an unhandled fault (Message: CWWBE0057I)**

The system log contains a CWWBE0057I message, the process is in the state "running", but it does not proceed its navigation on the current path.

## Reason

Invoke activities, inline human tasks, and Java snippets are put in a stopped state, if all of the following happen:

- A fault is raised by the activity
- The fault is not handled on the enclosing scope
- The `continueOnError` attribute of the activity is set to false

## Resolution

The solution to this problem requires actions at two levels:

1. An administrator must repair the stopped activity instance manually. For example, to force complete or force retry the stopped activity instance.
2. The reason for the failure must be investigated. In some cases the failure is caused by a modeling error that must be corrected in the model.

For example, if you use the WebSphere Scheduler default calendar, and have an expiration time with 'Timeout' defined for your activity, make sure that the definition of the time period is in the correct format, in particular make sure that there is no blank between the number and the unit of time. Examples of correctly specified timeout periods:

- 1minute
- 2hours 4minutes 1second
- 1day 1hour

### **A microflow is not compensated**

A microflow has called a service, and the process fails, but the undo service is not called.

## Resolution

There are various conditions that must be met to trigger the compensation of a microflow. Check the following:

1. Log on to the Business Process Choreographer Explorer and click **Failed Compensations** to check whether the compensation service has failed and needs to be repaired.
2. The compensation of a microflow is only triggered when the transaction for the microflow is rolled back. Check whether this is the case.
3. The `compensationSphere` attribute of the microflow must be set to required.
4. A compensation service is only run, if the corresponding forward service has not participated in the microflow's transaction. Ensure that the forward service does not participate in the navigation transaction, for example, on the reference of the process component, set the Service Component Architecture (SCA) qualifier `suspendTransaction` to True.

## A long-running process appears to have stopped

A long-running process is in the state running, but it appears that it is doing nothing.

### Reason

There are various possible reasons for such behavior:

1. A navigation message has been retried too many times and has been moved to the retention or hold queue.
2. A reply message from the Service Component Architecture (SCA) infrastructure failed repeatedly.
3. The process is waiting for an event, timeout, or for a long-running invocation or task to return.
4. An activity in the process is in the stopped state.

### Resolution

Each of the above reasons requires different corrective actions:

1. Check if there are any messages in the retention or hold queue, as described in the PDF for administering.
2. Check if there are any in the failed event management view of the administrative console.
  - If there are any failed events from Service Component Architecture (SCA) reply messages, reactivate the messages.
  - Otherwise, either force complete or force retry the long-running activity.
3. Check if there are activities in the stopped state, and repair these activities. If your system log contains a CWWBE0057I message you might also need to correct your model as described in Message: CWWBE0057I.

## Invoking a synchronous subprocess in another EAR file fails

When a long-running process calls another process synchronously, and the subprocess is located in another enterprise archive (EAR) file, the subprocess invocation fails.

Example of the resulting exception:

```
com.ibm.ws.sca.internal.ejb.util.EJBStubAdapter com.ibm.ws.sca.internal.ejb.util.EJBStubAdapter#003
Exception:
java.rmi.AccessException: CORBA NO_PERMISSION 0x49424307 No; nested exception is:
org.omg.CORBA.NO_PERMISSION: The WSCredential does not contain a forwardable token.
Please enable Identity Assertion for this scenario.
vmcid: 0x49424000 minor code: 307 completed: No
at com.ibm.CORBA.iiop.UtilDelegateImpl.mapSystemException(UtilDelegateImpl.java:202)
at javax.rmi.CORBA.Util.mapSystemException(Util.java:84)
```

### Reason

Common Secure Interoperability Version 2 (CSIv2) identity assertion must be enabled when calling a synchronous subprocess in another EAR file.

### Resolution

Configure CSIv2 inbound authentication and CSIv2 outbound authentication.



## **Unexpected exception during execution (Message: CWWBA0010E)**

Either the queue manager is not running or the Business Process Choreographer configuration contains the wrong database password.

### **Resolution**

Check the following:

1. If the `systemout.log` file contains "javax.jms.JMSEException: MQJMS2005: failed to create MQQueueManager", start the queue manager.
2. Make sure that the database administrator password stored in the Business Process Choreographer configuration matches the one set in the database.

## **Event unknown (Message: CWWBE0037E)**

An attempt to send an event to a process instance or to start a new process instance results in a "CWWBE0037E: Event unknown." exception.

### **Reason**

A common reason for this error is that a message is sent to a process but the receive or pick activity has already been navigated, so the message cannot be consumed by this process instance again.

### **Resolution**

To correct this problem:

- If the event is supposed to be consumed by an existing process instance, you must pass correlation set values that match an existing process instance which has not yet navigated the corresponding receive or pick activity.
- If the event is supposed to start a new process instance, the correlation set values must not match an existing process instance.

For more information about using correlation sets in business processes, see technote 1171649.

## **Cannot find nor create a process instance (Message: CWWBA0140E)**

An attempt to send an event to a process instance results in a 'CreateRejectedException' message.

### **Reason**

A common reason for this error is that a message is sent to a receive or pick activity that cannot instantiate a new process instance because its `createInstance` attribute is set to no and the values that are passed with the message for the correlation set which is used by this activity do not match any existing process instances.

### **Resolution**

To correct this problem you must pass a correlation set value that matches an existing process instance.

For more information about using correlation sets in business processes, see Correlation sets in BPEL processes.



## Uninitialized variable or NullPointerException in a Java snippet

Using an uninitialized variable in a business process can result in diverse exceptions.

### Symptoms

Exceptions such as:

- During the execution of a Java snippet or Java expression, that reads or manipulate the contents of variables, a NullPointerException is thrown.
- During the execution of an assign, invoke, reply or throw activity, the BPEL standard fault "uninitializedVariable" (message CWWBE0068E) is thrown.

### Reason

All variables in a business process have the value null when a process is started, the variables are not pre-initialized. Using an uninitialized variable inside a Java snippet or Java expression leads to a NullPointerException.

### Resolution

The variable must be initialized before it is used. This can be done by an assign activity, for example, the variable needs to occur on the to-spec of an assign, or the variable can be initialized inside a Java snippet.

## Standard fault exception "missingReply" (message: CWWBE0071E)

The execution of a microflow or long-running process results in a BPEL standard fault "missingReply" (message: CWWBE0071E), or this error is found in the system log or SystemOut.log file.

### Reason

A two-way operation must send a reply. This error is generated if the process ends without navigating the reply activity. This can happen in any of the following circumstances:

- The reply activity is skipped.
- A fault occurs and corresponding fault handler does not contain a reply activity.
- A fault occurs and there is no corresponding fault handler.

### Resolution

Correct the model to ensure that a reply activity is always performed before the process ends.

## Parallel paths are sequentialized

There are two or more parallel invoke activities inside a flow activity, but the invoke activities are run sequentially.

### Resolution

- To achieve real parallelism, each path must be in a separate transaction. Set the 'transactional behavior' attribute of all the parallel invoke activities to 'commit before' or 'requires own'.
- If you are using Cloudscape as the database system, the process engine will serialize the execution of parallel paths. You cannot change this behavior.

## Copying a nested data object to another data object destroys the reference on the source object

A data object, Father, contains another data object, Child. Inside a Java snippet or client application, the object containing Child is fetched and set on a substructure of data object, Mother. The reference to Child in data object Father disappears.

### Reason

The reference to Child is moved from Father to Mother.

### Resolution

When such a data transformation is performed in a Java snippet or client application, and you want to retain the reference in Father, copy the data object before it is assigned to another object. The following code snippet illustrates how to do this:

```
BOCopy copyService = (BOCopy)ServiceManager.INSTANCE.locateService
 ("com/ibm/websphere/bo/BOCopy");
DataObject Child = Father.get("Child");
DataObject BCopy = copyService.copy(Child);
Mother.set("Child", BCopy);
```

## CScope is not available

Starting a microflow or running a navigation step in a long-running process fails with an assertion, saying: 'postcondition violation !(cscope != null)'.

### Reason

In certain situations, the process engine uses the compensation service, but it was not enabled.

### Resolution

Enable the compensation service as described in the PDF for administration.

## Working with process-related or task-related messages

Describes how to get more information about Business Process Choreographer messages that are written to the display or a log file.

Messages that belong to Business Process Choreographer are prefixed with either *CWWB* for process-related messages, or *CWTK* for task-related messages. The format of these messages is *PrefixComponentNumberTypeCode*. The type code can be:

- I** Information message
- W** Warning message
- E** Error message

When processes and tasks run, messages are either displayed in Business Process Choreographer Explorer, or they are added to the `SystemOut.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 Business Process Choreographer messages, check the `activity.log` file by using the WebSphere log analyzer.

1. Start the WebSphere log analyzer.

Run the following script: `install_root/bin/waslogbr.sh`

2. **Optional:** Click **File > Update database > WebSphere Application Server Symptom Database** to check for the newest version of the symptom database.
3. **Optional:** Load the activity log.
  - a. Select the activity log file
    - *install\_root/profiles/profile\_name/logs/activity.log* file
  - b. click **Open**.

## Troubleshooting Business Process Choreographer Explorer

Use this to solve problems relating to the Business Process Choreographer Explorer.

Use the following information to solve problems relating to Business Process Choreographer Explorer.

- If you try to access Business Process Choreographer Explorer with a browser, but get the error message 'HTTP 404 - File not found', try the following:
  - Use the administrative console to make sure that the Web client application `BPCEXplorer_node_name_server_name` is actually deployed and running on the server.
  - In the administrative console, on the page for the application, under "View Deployment Descriptor", verify that the context root is the one you used when setting up the Business Process Choreographer Explorer.
- If you get an error message when using Business Process Choreographer Explorer, click the **Search for more information** link on the error page. This starts a search for the error code on the IBM technical support site. This site only provides information in English. Copy the error message code that is shown on the Business Process Choreographer Explorer Error page to the clipboard. The error code has the format `CWWBcnnnc`, where each `c` is a character and `nnnn` is a 4-digit number. Go to the WebSphere Process Server technical support page. Paste the error code into the **Additional search terms** field and click **Go**.
- If you get a `StandardFaultException` with the standard fault `missingReply` (message `CWWBE0071E`), this is a symptom of a problem with your process model. For more information about solving this, see "Troubleshooting the administration of business processes and human tasks" on page 337.
- If you can log onto Business Process Choreographer Explorer, but some items are not displayed, or if certain buttons are not enabled, this indicates a problem with your authorization.

Possible solutions to this problem include:

- Use the administrative console to turn security on.
- Check that you are logged onto Business Process Choreographer Explorer using the correct identity. If you log on with a user ID that is not a process or task administrator, all administrative views and options will be invisible or not enabled.
- Use WebSphere Integration Developer to check or modify the authorization settings defined in the business process.
- Error message `CWWBU0001E`: "A communication error occurred when the `BFMConnection` function was called" or "A communication error occurred when the `HTMConnection` function was called". This error can indicate that the business process container or human task container, respectively, has been stopped and the client could not connect to the server. Verify that the business process container and the human task container are running and accessible. The nested exception might contain further details about the problem.

- Error message WWBU0024E: "Could not establish a connection to local business process EJB with a reason "Naming Exception". This error is thrown if users attempt to log on while the business process container is not running. Verify that the application BPEContainer\_InstallScope is running, where *InstallScope* is either the cluster\_name or hostname\_servername.

**Related tasks**

"Troubleshooting the execution of business processes" on page 330

This describes the solutions to common problems with business process execution.

## Troubleshooting the administration of business processes and human tasks

This article describes how to solve some common problems with business processes and human tasks.

The following information can help you to debug problems with your business processes and human tasks.

- The administrative console stops responding if you try to stop a business process application while it still has process instances. Before you try to stop the application, you must stop the business processes so that no new instances are created, and do one of the following:
  - Wait for all of the existing process instances to end in an orderly way.
  - Terminate and delete all of the process instances.

Only then can you stop the process application. For more information about preventing this problem, refer to technote 1166009.

- The administrative console stops responding if you try to stop a human task application while it still has task instances. To stop the application, you must:
  1. Stop the human tasks so that no new instances are created.
  2. Perform one of the following:
    - Wait for all of the existing task instances to end in an orderly way.
    - Terminate and delete all task instances.
  3. Stop the task application.

## Troubleshooting the staff service, staff plug-ins, and staff resolution

Use the following information to help solve problems relating to staff assignments of people to authorization roles.

Topics covered are:

- Errors during staff verb deployment
- Entries in the staff repository are not reflected in work item assignments
- Unexpected staff assignment for tasks or processes
- Stopped staff activities
- Changes to the staff repository that are not immediately reflected in work item assignments
- Error and warning messages relating to staff resolution
- Issues with group work items and the "Group" verb

You can also search for additional information in the Technical support search page.

### Errors during staff verb deployment

If you are using the LDAP staff plug-in, deployment might fail due to incorrect values of the plug-in configuration parameters. Make sure that all mandatory parameters are set. To set the BaseDN parameter to the root of the LDAP directory tree, specify an empty string; set the BaseDN parameter to two apostrophe (') characters ("). Do not use double quotation marks ("). Failure to set the BaseDN parameter results in a NullPointerException exception at deployment time.

### Entries in the staff repository are not reflected in work item assignments

The maximum number of user IDs retrieved by a staff query is specified by the Threshold variable, which is defined in the XSL transformation file in use. The XSL transformation file used for the LDAP staff plug-in is, for example, LDAPTransformation.xsl, which is located in *install-root/ProcessChoreographer/Staff* on Linux and UNIX platforms and in *install-root\ProcessChoreographer\Staff* on Windows platforms. The default Threshold value is 20. To change this value:

1. Create a new staff plug-in provider configuration, providing your own version of the XSL file
2. Adapt the following entry in the XSL file according to your needs:

```
<xsl:variable name="Threshold">20</xsl:variable>
```

**Note:** If you specify a large Threshold value, it might result in a decrease in performance. For this reason, do not specify a value greater than 100.

### Unexpected staff assignments for tasks or process instances

Default staff assignments are performed if you do not define staff verbs for certain roles for your tasks, or if staff resolution fails or returns no result. These defaults might result in unexpected user authorization; for example, a process starter receives process administrator rights. In addition, many authorizations are inherited by dependent artifacts. For example, the process administrator may also become the administrator of all inline tasks.

The following tables illustrate which defaults apply for which situation:

Table 41. Roles for business processes

Roles for business processes	If the role is not defined in the process model ...	If the role is defined in the process model, but staff resolution fails or does not return proper results ...
Process administrator	Process starter becomes process administrator	The following exception occurs and the process is not started:  EngineAdministratorCannotBeResolvedException
Process reader	No reader	No reader

Table 42. Roles for inline human tasks and their escalations

Roles for inline human tasks and their escalations	If the role is not defined in the task model ...	If the role is defined in the task model, but staff resolution fails or does not return proper results ...
Task administrator	Only inheritance applies	Only inheritance applies
Task potential instance creator	Everybody becomes potential instance creator	Everybody becomes potential instance creator

Table 42. Roles for inline human tasks and their escalations (continued)

Roles for inline human tasks and their escalations	If the role is not defined in the task model ...	If the role is defined in the task model, but staff resolution fails or does not return proper results ...
Task potential starter	Everybody becomes potential starter	Everybody becomes potential starter
Task potential owner	Everybody becomes potential owner	Administrators become potential owners
Task editor	No editor	No editor
Task reader	Only inheritance applies	Only inheritance applies
Escalation receiver	Administrators become escalation receivers	Administrators become escalation receivers

The following inheritance rules apply for inline tasks:

- Process administrators become administrators for all inline tasks, their subtasks, follow-on tasks, and escalations.
- Process readers become readers for all inline tasks, their subtasks, follow-on tasks, and escalations.
- Task administrators become administrators for all subtasks, follow-on tasks, and escalations of all these tasks.
- Task readers become readers for all subtasks, follow-on tasks, and escalations of all these tasks.
- Members of any task role become readers for this task's escalations, subtasks, and follow-on tasks
- Escalation receivers become readers for the escalated task.

Table 43. Roles for stand-alone human tasks and their escalations

Roles for stand-alone human tasks and their escalations	If the role is not defined in the task model ...	If the role is defined in task model, but staff resolution fails or does not return correct results ...
Task administrator	Originator becomes administrator	The exception AdministratorCannotBeResolvedException is thrown and the task is not started
Task potential instance creator	Everybody becomes potential instance creator	Everybody becomes potential instance creator
Task potential starter	Originator becomes potential starter	The exception CannotCreateWorkItemException is thrown and the task is not started
Potential owner	Everybody becomes potential owner	Administrators become potential owners
Editor	No editor	No editor
Reader	Only inheritance applies	Only inheritance applies
Escalation receiver	Administrators become escalation receivers	Administrators become escalation receivers

The following inheritance rules apply for stand-alone tasks:

- Task administrators become administrators for all subtasks, follow-on tasks, and escalations of all these tasks.

- Task readers become readers for all subtasks, follow-on tasks, and escalations of all these tasks.
- Members of any task role become readers for this task's escalations, subtasks, and follow-on tasks
- Escalation receivers become readers for the escalated task.

**Note:** When a method is invoked via the Business Flow Manager API, members of the J2EE role BPESystemAdministrator have administrator authorization, and members of the J2EE role BPESystemMonitor have reader authorization.

**Note:** When a method is invoked via the Human Task Manager API, members of the J2EE role TaskSystemAdministrator have administrator authorization, and members of the J2EE role TaskSystemMonitor have reader authorization.

### Stopped staff activities

If you encounter one or more of the following problems:

- Human tasks cannot be claimed, even though the business process started navigating successfully.
- The SystemOut.log file contains the following message: CWWB0057I: 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. Human tasks and processes that use people authorization 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** and make sure the **Enable global security** check box is selected.
2. Check that the user registry is configured. In the administrative console, go to **Security** → **User Registries** and check the **Active user registry** attribute.
3. Restart the activity, if stopped.

### Changes to the staff repository are not immediately reflected in work-item assignments

Business Process Choreographer caches the results of staff assignments evaluated against a staff directory, such as a Lightweight Directory Access Protocol (LDAP) server, in the runtime database. When changes occur in the staff directory, these are not immediately reflected in the database cache.

The *Administration guide* describes three ways to refresh this cache:

- **Refreshing staff query results, using the Administrative Console.** Use this method if you have major changes and need to refresh the results for almost all staff queries.
- **Refreshing staff query results, using administrative commands.** Use this method if you write administration scripts using the wsadmin tool, or if you want to immediately refresh only a subset of the staff query results.
- **Refreshing staff query results, using the refresh daemon.** Use this method to set up a regular and automatic refresh of all expired staff query results.



**Note:** None of these methods can refresh the group membership association of a user for the Group verb. This group membership is cached in the user's login session (WebSphere security LTPA token), which by default expires after two hours. Also note that the group membership list of the process starter ID that is used for process navigation, is never refreshed.

### **Error and warning messages relating to staff resolution**

Some common errors can occur when accessing a staff repository during staff resolution. To see details for these errors, you can enable tracing with the following trace settings: `com.ibm.bpe.*=all`  
`com.ibm.task.*=all:com.ibm.ws.staffsupport.ws.*=all`

The following common error situations are indicated by warning or error messages:

- Could not connect to LDAP server in the trace.log file indicates failure to connect to the LDAP server. Check your network settings, the configuration (especially the provider URL) for the staff plug-in provider you use, and verify whether your LDAP server requires an SSL connection.
- `javax.xml.transform.TransformerException: org.xml.sax.SAXParseException: Element type "xsl:template" must be followed by either attribute specifications, ">" or "/>"` in the System.out or System.err files indicates that the LDAPTransformation.xsl file cannot be read. Check your staff plug-in provider configuration and the configured XSLT file for errors.
- LDAP object not found. `dn: uid=unknown,cn=users,dc=ibm,dc=com [LDAP: error code 32 - No Such Object]` in the trace.log file indicates that an LDAP entry cannot be found. Check the task model's staff verb parameters and the LDAP directory content for mismatches in the task model.
- Requested attribute "uid" not found in: `uid=test222,cn=users,dc=ibm,dc=com` in the trace.log file indicates that an attribute cannot be found in the queried LDAP object. Check the task model's staff verb parameters and the LDAP directory content for mismatches in the task model. Also check the XSLT file of your staff provider configuration for errors.

### **Issues with group work items and the "Group" verb**

When using the Group verb, some special situations can occur:

- Group members are not authorized, although the group name is specified:
  - Specify the group short name when using the Local OS registry for WebSphere security, and the group dn when using the LDAP registry.
  - Make sure that you respect the case sensitivity of the group name.

One possible reason for this situation is that you have configured the LDAP user registry for WebSphere security and selected the **Ignore case for authorization** option. If so, either deselect the option, or specify LDAP group dn in all uppercase.

- Changes in group membership are not immediately reflected in authorization. This might happen, when the affected user is still logged on. The group membership of a user is cached in her login session, and (by default) expires after two hours. You can either wait for the login session to expire (default is two hours), or restart the application server.



The refresh methods offered by Human Task Manager do not apply for this verb. Note that the group membership list of the process starter is never refreshed.

## Using process-related and task-related audit trail information

Explains the event types and database structures for business processes and human tasks.

Logging must be enabled for the business process container, the task container, or both.

If logging is enabled, whenever a significant step during the running of a business process or a human task occurs, information is written to the audit log or Common Event Infrastructure (CEI) log. For more information about CEI, refer to the *Monitoring WebSphere Process Server* PDF. The following topics describe the event types and database structures for business processes and human tasks.

### Audit event types for business processes

This describes the types of events that can be written to the audit log during the processing of business processes.

For an event to be logged, the following conditions must be met:

- The corresponding audit logging type is enabled for the business process container
- The event must be enabled for the corresponding entity in the process model

The following tables list the codes for audit events that can occur while business processes are running.

Table 44. Process instance events

Audit event	Event code
PROCESS_STARTED	21000
PROCESS_SUSPENDED	21001
PROCESS_RESUMED	21002
PROCESS_COMPLETED	21004
PROCESS_TERMINATED	21005
PROCESS_RESTARTED	21019
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
PROCESS_COMPENSATION_FAILED	42046
PROCESS_EVENT_RECEIVED	42047

Table 44. Process instance events (continued)

Audit event	Event code
PROCESS_EVENT_ESCALATED	42049
PROCESS_WORKITEM_TRANSFERRED	42056

Table 45. 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
ACTIVITY_LOOPED	42002
ACTIVITY_SKIPPED	42005
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_SKIPPED	42034
ACTIVITY_UNDO_COMPLETED	42035
ACTIVITY_MESSAGE_RECEIVED	42036
ACTIVITY_LOOP_CONDITION_TRUE	42037
ACTIVITY_LOOP_CONDITION_FALSE	42038
ACTIVITY_WORKITEM_DELETED	42039
ACTIVITY_WORKITEM_CREATED	42040
ACTIVITY_ESCALATED	42050
ACTIVITY_WORKITEM_REFRESHED	42054
ACTIVITY_WORKITEM_TRANSFERRED	42055
ACTIVITY_PARALLEL_BRANCHES_STARTED	42057

Table 46. Events related to variables

Audit event	Event code
VARIABLE_UPDATED	21090

Table 47. Control link events

Audit event	Event code
LINK_EVALUATED_TO_TRUE	21034
LINK_EVALUATED_TO_FALSE	42000

Table 48. Process template events

Audit event	Event code
PROCESS_INSTALLED	42006
PROCESS_UNINSTALLED	42007

Table 49. Scope instance events

Audit event	Event code
SCOPE_STARTED	42020
SCOPE_SKIPPED	42021
SCOPE_FAILED	42022
SCOPE_FAILING	42023
SCOPE_TERMINATED	42024
SCOPE_COMPLETED	42026
SCOPE_COMPENSATING	42043
SCOPE_COMPENSATED	42044
SCOPE_COMPENSATION_FAILED	42045
SCOPE_EVENT_RECEIVED	42048
SCOPE_EVENT_ESCALATED	42051

## Audit event types for human tasks

This describes the types of events that can be written to the audit log during the processing of human tasks.

For an event to be logged, the following conditions must be met:

- The corresponding audit logging type is enabled for the human task container
- The event must be enabled for the corresponding entity in the task model

The following tables list the codes for audit events that can occur while human tasks are running.

Table 50. Task instance events

Audit event	Event code
TASK_CREATED	51001
TASK_DELETED	51002
TASK_STARTED	51003
TASK_COMPLETED	51004
TASK_CLAIM_CANCELLED	51005
TASK_CLAIMED	51006
TASK_TERMINATED	51007
TASK_FAILED	51008

Table 50. Task instance events (continued)

Audit event	Event code
TASK_EXPIRED	51009
TASK_WAITING_FOR_SUBTASK	51010
TASK_SUBTASKS_COMPLETED	51011
TASK_RESTARTED	51012
TASK_SUSPENDED	51013
TASK_RESUMED	51014
TASK_COMPLETED_WITH_FOLLOW_ON	51015
TASK_UPDATED	51101
TASK_OUTPUT_MESSAGE_UPDATED	51103
TASK_FAULT_MESSAGE_UPDATED	51104
TASK_WORKITEM_DELETED	51201
TASK_WORKITEM_CREATED	51202
TASK_WORKITEM_TRANSFERRED	51204
TASK_WORKITEM_REFRESHED	51205

Table 51. Task template events

Audit event	Event code
TASK_TEMPLATE_INSTALLED	52001
TASK_TEMPLATE_UNINSTALLED	52002

Table 52. Escalation instance events

Audit event	Event code
ESCALATION_FIRED	53001
ESCALATION_WORKITEM_DELETED	53201
ESCALATION_WORKITEM_CREATED	53202
ESCALATION_WORKITEM_TRANSFERRED	53204
ESCALATION_WORKITEM_REFRESHED	53205

## Structure of the audit trail database view for business processes

The AUDIT\_LOG\_B database view provides audit log information about business processes.

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 template events (PTE)
- Process instance events (PIE)
- Activity instance events (AIE)
- Events related to variables (VAR)
- Control link events (CLE)
- Scope-related events (SIE).

For a list of the audit event type codes, see “Audit event types for business processes” on page 342.

The following table describes the structure of the AUDIT\_LOG\_B audit trail view. It lists the names of the columns, the event types, and gives a short description for the column.

Inline tasks are logged in the AUDIT\_LOG\_B audit trail view and not in the TASK\_LOG audit trail view. For example, claiming an inline participating task results in an ACTIVITY\_CLAIMED event; a task-related event is not generated.

Table 53. Structure of the AUDIT\_LOG\_B audit trail view

Name	PTE	PIE	AIE	VAR	CLE	SIE	Description
AIID			x				The ID of the activity instance that is related to the current event.
ALID	x	x	x	x	x	x	Identifier of the audit log entry.
EVENT_TIME	x	x	x	x	x	x	Timestamp of when the event occurred in Coordinated Universal Time (UTC) format.
EVENT_TIME_UTC	x	x	x	x	x	x	Timestamp of when the event occurred in Coordinated Universal Time (UTC) format.
AUDIT_EVENT	x	x	x	x	x	x	The type of event that occurred.
PTID	x	x	x	x	x	x	Process template ID of the process that is related to the current event.
PIID		x	x	x	x	x	Process instance ID of the process instance that is related to the current event.
VARIABLE_NAME				x			The name of the variable related to the current event.
SIID						x	The ID of the scope instance related to the event.
PROCESS_TEMPL_NAME	x	x	x	x	x	x	Process template name of the process template that is related to the current event.
TOP_LEVEL_PIID		x	x	x	x	x	Identifier of the top-level process that is related to the current event.
PARENT_PIID		x	x	x	x	x	Process instance ID of the parent process, or null if no parent exists.
VALID_FROM	x	x	x	x	x	x	Valid-from date of the process template that is related to the current event.
VALID_FROM_UTC	x	x	x	x	x	x	Valid-from date of the process template that is related to the current event in Coordinated Universal Time (UTC) format.
ATID			x				The ID of the activity template related to the current event.
ACTIVITY_NAME			x			x	Name of the activity on which the event occurred.

Table 53. Structure of the AUDIT\_LOG\_B audit trail view (continued)

Name	PTE	PIE	AIE	VAR	CLE	SIE	Description
ACTIVITY_KIND			x				<p>Kind of the activity on which the event occurred. Possible values are:</p> <p>KIND_EMPTY 3            KIND_INVOKE 21            KIND_RECEIVE 23            KIND_REPLY 24            KIND_THROW 25            KIND_TERMINATE 26            KIND_WAIT 27            KIND_COMPENSATE 29            KIND_SEQUENCE 30            KIND_SWITCH 32            KIND_WHILE 34            KIND_PICK 36            KIND_FLOW 38            KIND_SCRIPT 42            KIND_STAFF 43            KIND_ASSIGN 44            KIND_CUSTOM 45            KIND_RETHROW 46            KIND_FOR_EACH_SERIAL 47            KIND_FOR_EACH_PARALLEL 49</p> <p>These are the constants defined for ActivityInstanceData.KIND_*</p>
ACTIVITY_STATE			x				<p>State of the activity that is related to the event. Possible values are:</p> <p>STATE_INACTIVE 1            STATE_READY 2            STATE_RUNNING 3            STATE_SKIPPED 4            STATE_FINISHED 5            STATE_FAILED 6            STATE_TERMINATED 7            STATE_CLAIMED 8            STATE_TERMINATING 9            STATE_FAILING 10            STATE_WAITING 11            STATE_EXPIRED 12            STATE_STOPPED 13</p> <p>These are the constants defined for ActivityInstanceData.STATE_*</p>
CONTROL_LINK_NAME					x		Name of the link that is related to the current link event.
PRINCIPAL		x	x	x	x	x	Name of the principal. This is not set for PROCESS_DELETED events.
VARIABLE_DATA				x			Data for variables for variable updated events.

Table 53. Structure of the AUDIT\_LOG\_B audit trail view (continued)

Name	PTE	PIE	AIE	VAR	CLE	SIE	Description
EXCEPTION_TEXT		x	x			x	Exception message that caused an activity or process to fail. Applicable for:  PROCESS_FAILED ACTIVITY_FAILED SCOPE_FAILED
DESCRIPTION		x	x	x	x	x	Description of activity or process, containing potentially resolved replacement variables.
CORR_SET_INFO		x					The string representation of the correlation set that was initialized at process start time. Provided with the processCorrelationSetInitialized event (42027).
USER_NAME		x	x				The name of the user whose work item has been changed. This is applicable for the following events: <ul style="list-style-type: none"> <li>• Process instance work item deleted</li> <li>• Activity instance work item deleted</li> <li>• Process instance work item created</li> <li>• Activity instance work item created</li> </ul>
ADDITIONAL_INFO		x	x			x	The contents of this field depends on the type of the event:  <b>ACTIVITY_WORKITEM_TRANSFERRED,</b> <b>PROCESS_WORK_ITEM_TRANSFERRED</b> The name of the user that received the work item.  <b>ACTIVITY_WORKITEM_CREATED,</b> <b>ACTIVITY_WORKITEM_REFRESHED,</b> <b>ACTIVITY_ESCALATED</b> The list of all of the users for which the work item was created or refreshed, separated by ';'. If the list contains only one user, the USER_NAME field is filled with the user name of this user and the ADDITIONAL_INFO field will be empty (null).  <b>PROCESS_EVENT_RECEIVED,</b> <b>SCOPE_EVENT_RECEIVED</b> If available, the type of operation that was received by an event handler. The following format is used: '{' port type namespace '}' port type name ':' operation name. This field is not set for 'onAlarm' events.

## Structure of the audit trail database view for human tasks

The TASK\_AUDIT\_LOG database view provides audit log information about human tasks.

Inline tasks are logged in the AUDIT\_LOG\_B view. All other task types are logged in the TASK\_AUDIT\_LOG view.

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 task entities. The audit event types depend on the entity to which the event refers. The audit event types include:

- Task instance events (TIE)
- Task template events (TTE)
- Escalation instance events (EIE)

The following table describes the structure of the TASK\_AUDIT\_LOG audit trail view. It lists the names of the columns, the event types, and gives a short description for the column.

Inline tasks are logged in the AUDIT\_LOG\_B audit trail view and not in the TASK\_AUDIT\_LOG audit trail view. For example, claiming an inline participating task results in an ACTIVITY\_CLAIMED event; a task-related event is not generated.

Table 54. Structure of the TASK\_AUDIT\_LOG audit trail view

Name	TIE	TTE	EIE	Description
ALID	x	x	x	The identifier of the audit log entry.
AUDIT_EVENT	x	x	x	The type of event that occurred. For a list of audit event codes, see "Audit event types for human tasks" on page 344.
CONTAINMENT_CTX_ID	x	x		The identifier of the containing context, for example, ACOID, PTID, or PIID.
DESCRIPTION	x		x	Resolved description string, where placeholders in the description are replaced by their current values. All affected languages are logged together in this column, formatted as an XML document. Only languages with descriptions containing placeholders for create-like events, or that have been explicitly updated for update-like events, are logged.
ESIID			x	The identifier of the escalation instance that is related to the current event.
ESTID			x	The identifier of the escalation template that is related to the current event.
EVENT_TIME	x	x	x	The time when the event occurred in Coordinated Universal Time (UTC) format.
FAULT_NAME	x			The name of the fault message. This attribute is applicable to the following events:  TASK_FAILED TASK_FAULT_MESSAGE_UPDATED



Table 54. Structure of the TASK\_AUDIT\_LOG audit trail view (continued)

Name	TIE	TTE	EIE	Description
FAULT_NAME_SPACE	x			The namespace of the fault message type. This attribute is applicable to the following events:  TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FOLLOW_ON_TKIID	x			The ID of the follow-on task instance.
MESSAGE_DATA	x			Contents of the newly created or updated input, output, or fault message.
NAME	x	x	x	The name of the task instance, task template, or escalation instance that is associated with the event.
NAMESPACE	x	x		The namespace of the task instance, task template, or escalation instance that is associated with the event.
NEW_USER				The new owner of a transferred or created work item. If the value is made available via the USERS field, this value may be null . Also see the field USERS. This attribute applies to the following events:
	x			TASK_WORKITEM_CREATED
	x			TASK_WORKITEM_TRANSFERRED
			x	ESCALATION_WORKITEM_CREATED
OLD_USER				The previous owner of a transferred work item. This attribute is applicable to the following events:
	x			TASK_WORKITEM_TRANSFERRED
	x			TASK_WORKITEM_DELETED
			x	ESCALATION_WORKITEM_TRANSFERRED
PARENT_CONTEXT_ID				The ID of the parent context of the task, for example, an activity template or a task instance. This is only set for subtasks and follow-on tasks.
	x			
PARENT_TASK_NAME	x			The name of the parent task instance or template. This is only set for subtasks and follow-on tasks.
PARENT_TASK_NAMESP	x			The namespace of the parent task instance or template. This is only set for subtasks and follow-on tasks.
PARENT_TKIID	x			The identifier of the parent task instance.
PRINCIPAL	x	x	x	The name of the principal whose request triggered the event.
TASK_KIND	x	x		The kind of the task. Possible values are:  KIND_HUMAN 101 KIND_ORIGINATING 103 KIND_PARTICIPATING 105 KIND_ADMINISTRATIVE 106

Table 54. Structure of the TASK\_AUDIT\_LOG audit trail view (continued)

Name	TIE	TTE	EIE	Description
TASK_STATE	x			<p>The state of the task or task template. Possible values for task templates are:</p> <p>STATE_STARTED 1 STATE_STOPPED 2</p> <p>Possible values for task instances are:</p> <p>'1' :STATE_INACTIVE' '2' :STATE_READY' '3' :STATE_RUNNING' '5' :STATE_FINISHED' '6' :STATE_FAILED' '7' :STATE_TERMINATED' '8' :STATE_CLAIMED' '12' :STATE_EXPIRED' '101':FORWARDED'</p>
TKIID	x		x	The identifier of the task instance.
TKTID	x	x		The identifier of the task template.
TOP_TKIID	x			The identifier of the top task instance.
USERS	x		x	The new user IDs assigned to a task or escalation work item. If the value is made available via the NEW_USER field, this may have the value null. See the field NEW_USER for a list of events to which this attribute applies.
VALID_FROM		x		Valid-from date of the task template that is related to the current event.
WORK_ITEM_REASON	x		x	<p>The reason for the assignment of the work item. Possible values are:</p> <p>POTENTIAL_OWNER 1 EDITOR 2 READER 3 OWNER 4 POTENTIAL_STARTER 5 STARTER 6 ADMINISTRATOR 7 POTENTIAL_SENDER 8 ORIGINATOR 9 ESCALATION_RECEIVER 10 POTENTIAL_INSTANCE_CREATOR 11</p> <p>The reason is set for all events related to work items: ESCALATION_RECEIVER is set for escalation work item related events, while the other reasons apply to task work item related events.</p>



---

## Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### COPYRIGHT LICENSE:

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

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

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

---

## Programming interface information

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

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

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

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

---

## Trademarks and service marks

IBM and related trademarks: <http://www.ibm.com/legal/copytrade.shtml>

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

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



WebSphere Process Server for z/OS, Version 6.0.2









Printed in USA