WebSphere® Process Server for Multiplatforms

**Version 6.1.0**

IBM

**Business Process Choreographer**

**WebSphere**® Process Server for Multiplatforms

Version 6.1.0

IBM

**Business Process Choreographer**

**21 December 2007**

This edition applies to version 6, release 1, modification 0 of WebSphere Process Server for Multiplatforms (product number 5724-L01) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

# Contents

**Chapter 4. Configuring Business
Process Choreographer . . . . . . . 141**

# Part 1. Business processes and human tasks in WebSphere Process Server

# Chapter 1. About business processes

A business process is a set of business-related activities that are invoked 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 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, for example, assign or invoke activities.
  - Structured activities are activities that contain other activities, for example, sequence or while activities.
- The partner links, also known as interface partners or reference partners, that specify the interaction with external partners using WSDL interfaces.
- The variables that store the data that is exchanged with the process and passed between activities.
- Correlation sets that are used to correlate multiple service interactions 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 process execution.
- Compensation handlers that specify the compensation logic for a single activity, a group of activities, or a scope.

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 to-do tasks can be steps in the business process that involve a person, for example, completing a form, approving a document, and so on.
- Script activities for running inline Java™ code. The Java code can access all of the BPEL variables, correlation properties, partner links, and process and activity contexts.
- Information service activities to directly access WebSphere® Information Server or relational databases.
- Valid-from timestamps for process model versioning.
- Extensions for manually setting or controlling the transactional boundaries in a business process.
- Timeouts for activities.

  **Related tasks**

  Chapter 9, "Administering business processes and human tasks," on page 333
  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.

"Administering process templates and process instances" on page 333
Use the administrative console or the administrative commands to administer process templates. Use Business Process Choreographer Explorer to work with process instances.

"Stopping and starting process templates with the administrative console" on page 336
You can use the administrative console to start and stop each installed process template individually.

"Stopping and starting process templates with administrative commands" on page 336
Administrative commands provide an alternative to the administrative console for stopping and starting process templates. Use the administrative commands to stop all process templates within an enterprise application.

"Managing the process life cycle" on page 337
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" on page 338
You can start a new process instance from any of the process templates that you are authorized to use.

"Monitoring the progress of a process instance" on page 338
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.

"Suspending and resuming process instances" on page 339
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.

"Terminating process instances" on page 340
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.

"Deleting process instances" on page 341
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.

"Repairing processes and activities" on page 342
If the process runs into problems, you can analyze the process and then repair the activities.

"Restarting activities" on page 342
If you have repaired an activity, you can restart it using new input data.

"Forcing the completion of activities" on page 343
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.

"Administering compensation for microflows" on page 343
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.

**Related information**

⏏ Business Process Exection Language for Web Services, Version 1.1

⏏ OASIS Web Services Business Process Exection Language, Version 2.0

# Process templates

A process template is a process definition that is deployed and installed in the runtime environment.

In addition to the properties that are specified when the process is defined, an installed business process can also have one of the following states:

**Started**
> When a process template is created and started, new instances of the template can be started.

**Stopped**
> The process template must be stopped before the business process application can be uninstalled. When a process template is in the stopped state, no new instances of this template can be created and started.

## 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 in multiple transactions.
- Interacts with services synchronously and asynchronously.
- Its state is stored in the runtime database, 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 participates in the unit of work that can be either a global transaction or an activity session.

A microflow has the following characteristics:
- Runs in one transaction or activity session
- Normally runs for a short time
- Its state is transient, and it is therefore not stored in the runtime database
- It typically invokes services synchronously
- It can have only non-interruptible child processes
- It cannot contain:

- Human tasks
- Wait activities
- Non-initiating receive activities or pick activities

**Related concepts**

"Factors affecting business process interactions" on page 17
A number of factors affect the behavior of business processes in the various invocation scenarios. These include the interaction style, the type of business process, the operation type, and the service endpoint resolution.

# Process versioning

You can create new versions of your business process, so that multiple versions of the same processes can co-exist in a runtime environment.

You can include versioning information, such as a valid-from date, when you define the business process in WebSphere Integration Developer. The version of a process is determined by its valid-from date. This means that different versions of a process can have the same process name but they have different valid-from dates. The version of a process that is used at runtime 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 process is invoked is made either during modeling or when the process is deployed. The caller invokes a dedicated, statically-bound process. Even if another version of the process is valid according to the valid-from dates of the different versions, the current statically wired process is called, and all of the other versions are ignored.

An example of early-binding is an SCA wire. 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 process template is invoked happens when the caller invokes the process. In this case, the version of the process that is currently valid is used. The currently valid 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:

- Currently valid process templates are used for new process instances
- Process templates that are no longer valid are still used for existing long-running process instances
- Process templates that become valid in the future according to their valid-from date

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.

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 currently valid version of the process with a valid-from date that is not in the future.

# Process instances

A process instance is a stateful manifestation of a process template.

Business processes defined in Web Services Business Process Execution Language (WS-BPEL) represent stateful Web services, and as such, they can have long-running interactions with other Web services. Whenever a BPEL process is started, a new instance of that process is created that can communicate with other business partners. An instance completes when its last activity completes, a terminate activity runs, or the instance experiences a fault that is not handled by the process.

# Process life cycle

When a process is initiated, the processing of a business process instance is started, and it begins to interact with its environment. This means that certain interactions are only possible in certain process states, and these interactions, in turn, influence the state of the process instance.

## State transition diagrams for process instances

Processes change state whenever something of significance happens during the life cycle of the process instance. For example, an API request causes a process in the running state to be put into the suspended state. State transition diagrams show the state transitions that can occur during the process life cycle.

### Conventions used in these diagrams

The state transitions in the diagrams are indicated by numbers. These numbers are then explained in the supporting text. In addition, the diagrams contain the following types of symbols:

| Symbol | Explanation |
|---|---|
| running | Transient state. These states are not visible. |
| running | Persistent state. |
| finished | Transient end state. |
| finished | Persistent end state. |
| - - ①- ➤ | State transitions that are triggered automatically by Business Flow Manager. |
| ⑦→ | State transitions that are the result of an external interaction using an API. |
| ⑬· ·▶ | State transitions that are controlled by Business Flow Manager, or are the result of an external interaction using an API. |

## State transition diagram for microflow instances

A microflow is considered to be stateless because the process is always run in a transaction and instance information is not persisted for navigating the process instance. However, depending on the process definition and how Business Flow Manager is configured, the state of a microflow can be exposed in Common Base Events or in the audit log.

The following diagram shows the states that a microflow instance can have.
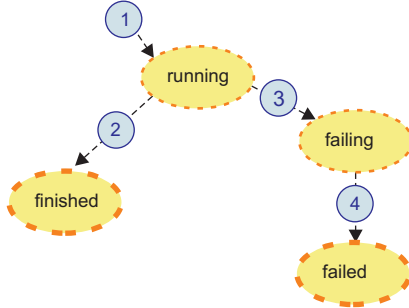
After normal initiation of the process instance, the first process state a process instance reaches is the running state (1). When a process instance runs normally through to completion, the process state changes from running to finished (2). If a fault reaches the process boundary, the process is put into the failing state (3). The process stays in the failing state while the fault handler runs. After this, the process instance is put into the failed state (4).

All of these state transitions are triggered by Business Flow Manager. After a microflow starts, you cannot influence these automatic steps.

## State transition diagram for long-running process instances

A long-running process runs in several transactions. The state of a long-running process is persisted, and it is therefore visible. The following diagram shows the state transitions that can occur for a long-running process instance.
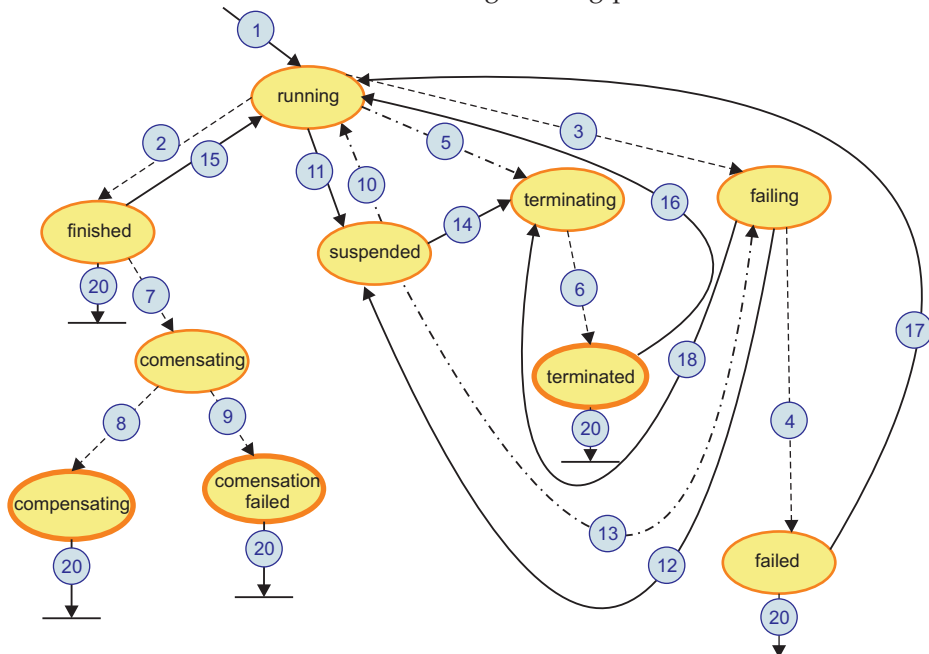
The running, finished, failing, and failed states, and the state transitions between them are the same as for microflows.

A process instance is terminated by either an external request, or a terminate activity. The termination of a process instance can span multiple navigation steps and therefore multiple chained transactions, for example, to terminate long-running activities or subprocesses. During this termination phase, the process instance is in the terminating state (5), (14), (18). When all of the long-running parts of the process are terminated, the state of the process instance also changes to terminated (6).

When a child process ends successfully and the parent process fails later, the child process can be compensated. During compensation, the child process is in the compensating state (7). If compensation ends successfully, the child process is put into the compensated state (8). If compensation is not successful, the child process is put into the compensation-failed state (9). These state transactions are initiated by the parent process automatically.

If the navigation of the process instance is still active, that is, it is in the running, or failing state, it can be suspended with an API request. It can then be reactivated either after a specified time, or by a resume request. The state of the process changes from running or failing to suspended (11), (12) with the suspend request, and from suspended to running or failing with the resume request (10), (13). A process in the suspended state can also be terminated (14). Only top-level process instances can be suspended and resumed. However, the suspend or resume state is propagated to the child processes.

When a process reaches one of the end states, finished, terminated, or failed, it can be started again with a restart API request (15), (16), (17). Only top-level process instances can be restarted, while only child process instances can be compensated.

A process instance can be deleted when it reaches an end state (20). The process can be deleted automatically if the **automatically delete on completion** attribute is set accordingly, or it can be triggered by an explicit delete request.

## Life cycle management 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 depends on how these processes are modeled.

For modularity and reuse, it often makes sense 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 a 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.

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 operations comprise suspend, resume, terminate, delete, and compensation. In a parent-child relationship, operations that manage the process life cycle can be taken only on top-level process instances.

The 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 operations on the calling process instance are not propagated to the subprocess instance.

A long-running process that is created and started using a one-way interface is considered to be a peer process. Its autonomy attribute is ignored at runtime.

**Child**  If the subprocess is part of a parent-child relationship, life cycle operations 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. The child process must be complete when it returns to its parent process, that is, the last operation of a child process must be its reply to the calling parent process. Make sure that all of the possible paths in the process logic end with a reply activity as the last operation in the path.

A microflow always runs as a child process, that is, its autonomy attribute is ignored.

A parent-child relationship can be established only between processes that interact directly. If another SCA component intercepts this interaction, it might prevent a parent-child relationship from being established, for example, an interface map component that is wired between the two process components.

# State transition diagram for activities

The state of an activity instance changes when a significant step in the execution of the activity instance occurs. The states and the state transitions depend on the type of activity.

States and state transitions are important in the life cycle of basic activities. Basic activities are grouped into the following activity types. The state transitions diagrams vary according to the activity type:

- Short-lived activities, such as assign, empty, reply, rethrow, throw, and terminate activities
- Activities that wait for an external event, such as receive and wait activities
- Pick (receive choice) activities
- Java snippet activities
- Invoke activities
- Human task activities

In contrast to the state diagrams for process instances, activity end states are not explicitly exposed. The life cycle of an activity depends on the enclosing process. Activities are always deleted with the process instance.

## Conventions used in these diagrams

The state transitions in the diagrams are indicated by numbers. These numbers are then explained in the supporting text. In addition, the diagrams contain the following types of symbols:

| Symbol | Explanation |
|---|---|
| running | Transient state. These states are not visible. |
| running | Persistent state. |
| - - - 1 - ➤ | State transitions that are triggered automatically by Business Flow Manager. |
| —— 7 —→ | State transitions that are the result of a user interaction, for example, by an API request. |
| —— 13 -·➤ | State transitions that are controlled by Business Flow Manager or by a user interaction. |

## State transition diagram for short-lived activity types

The following state diagram shows the states and the state transitions for simple, short-lived activity types, such as: assign, empty, reply, rethrow, throw, and terminate activities. It introduces the states: inactive, skipped, finished, and failed. These states are common to all of the basic activity types.
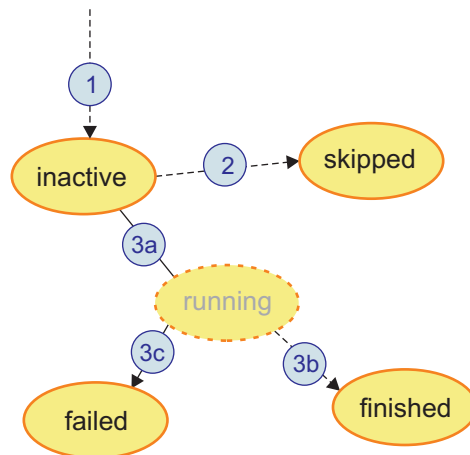


After an activity is created, it is in the inactive state (1). Activities that are enclosed in a flow can have multiple incoming links and a join condition. Before such an activity can start, all of the incoming links must be navigated. The **suppressJoinFailure** attribute of the activity and the outcome of the evaluation of the join condition determine the subsequent behavior of the activity:

- The join condition evaluates to `false` and the **suppressJoinFailure** attribute is set to `true`.

  The state of the activity changes to skipped (2), and the links leaving the activity are navigated as dead paths.

- The join condition evaluates to `false` and the **suppressJoinFailure** attribute is set to `false`.

  The activity remains in the inactive state because it has not been started, and a `bpws:joinFailure` standard fault is raised.

- The join condition evaluates to `true`.

  For activities that are not enclosed in a flow, this is the expected behavior. The activity is activated, and its state changes to running (3a). The activity

implementation is run, and then the activity state changes to finished (3b). If the implementation fails, for example, when the syntax of a copy statement in an assign activity is incorrect, the state of the activity changes to failed (3c). All short-lived activities are non-interruptible. As a result, the running state is never visible.

## State transition diagram for activities that wait for an external event

The following diagram shows the states and the state transitions that can occur during the life cycle of a wait or a receive activity.
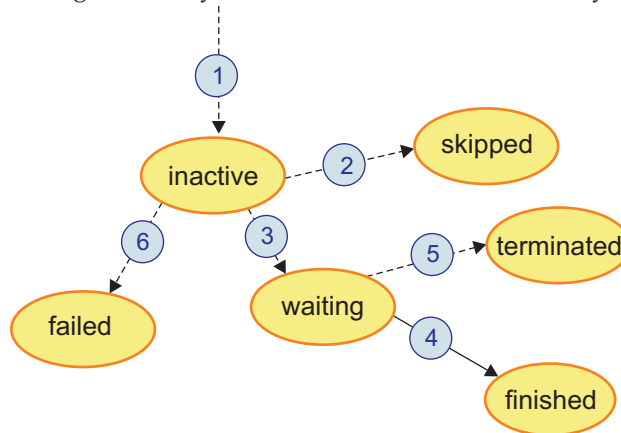


The starting phase of receive and wait activities is the same as for short-lived activities. However, after receive and wait activities are activated, the state changes to waiting instead of running (3). The receive or wait activity is now ready to receive an external request, or to wait for the specified timeout, before it can complete and move to the finished state (4). For a receive activity, the transition to the finished state is triggered by the message that is received. For a wait activity, this transition is done automatically after the specified wait time elapses.

The implementation of a wait or receive activity is more complex than the implementation of simple, short-lived activity types. The wait or receive activity might fail before the start of the activity completes, for example, when the evaluation of the wait time of a wait activity fails. This failure causes the activity state to change to failed (6) before it can reach the waiting state.

While the activity is in the waiting state, the enclosing process might receive a terminate request, or a fault occurs in a branch that is parallel to the wait or receive activity. If any of these events occur, the wait or receive activity is terminated, and the state of the activity changes to terminated (5).

## State transition diagram for a pick (receive choice) activity

The states and state transitions for pick activities (also known as receive choice activities) are shown in the following state diagram.
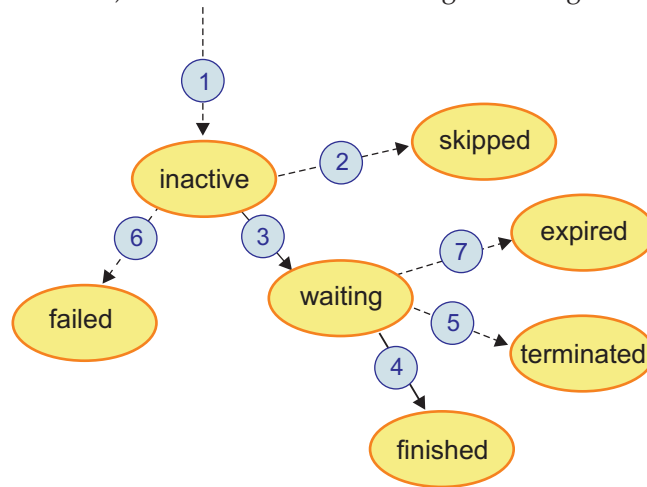


The states and state transitions (1) through (6) for pick activities are the same as those of receive activities.

In addition, a pick activity can expire when the on-alarm branch of a waiting pick activity is activated before a request for the pick activity arrives. The activity is then in the expired state (7).

## State transition diagram for a Java snippet activity

The following diagram shows the states and the state transitions that can occur during the life cycle of a Java snippet activity.
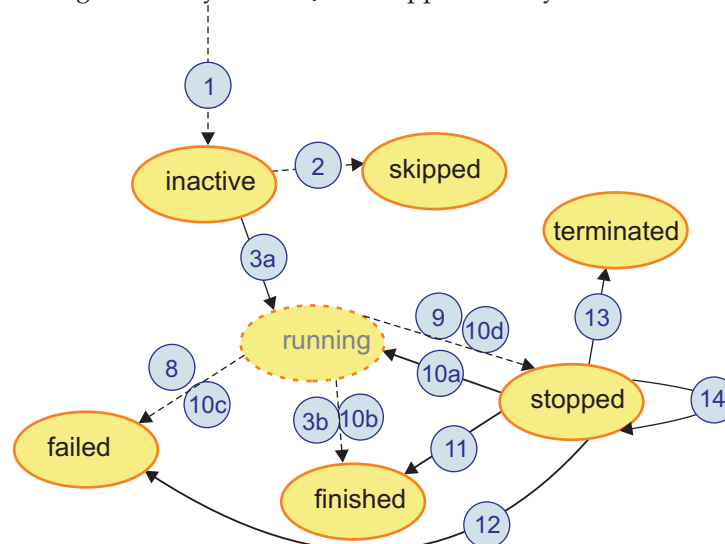


The runtime behavior of Java snippet activities is similar to that of short-lived activities; when a Java snippet activity is activated (3a), the implementation is run. Because the implementation of a Java snippet is short, the running state is not visible. If the Java snippet completes successfully, the state of the activity changes immediately to finished (3b).

If the Java snippet fails, the activity can behave in one of the following ways:

- The activity state changes to stopped (9). If the **continueOnError** attribute of the activity is set to `false` and the fault thrown by the Java snippet cannot be caught by a fault handler on the enclosing scope of the Java snippet, administrative interaction is required so that the process navigation can continue.
- The activity state changes to failed (8), and fault handling is started.

If the activity is in the stopped state, it can be force retried or force completed using an API request. If it is force retried, the activity is put into the running state and the implementation is run again (10a). Because the external or process-internal conditions have changed, the Java snippet can complete successfully, and the state of the activity changes to finished (10b). If the activity fails again, the administrator can use an implementation of the force-retry API to stop the activity (10d), or leave the failure to the process fault handling (10c).

If the enclosing process or scope is terminated while the activity is in the stopped state, the activity state changes to terminated (13).

If the activity is forced to complete, the implementation of the activity is not run again. The activity is put into the finished (11), failed (12), or stopped state (14).

## State transition diagram for an invoke activity

The following diagram shows the states and the state transitions that can occur during the life cycle of an invoke activity with an asynchronous implementation. The implementation is asynchronous if the service reply happens in a subsequent transaction to the service request transaction. The SCA qualifier of the process and the service component determine whether a service is called synchronously or asynchronously.
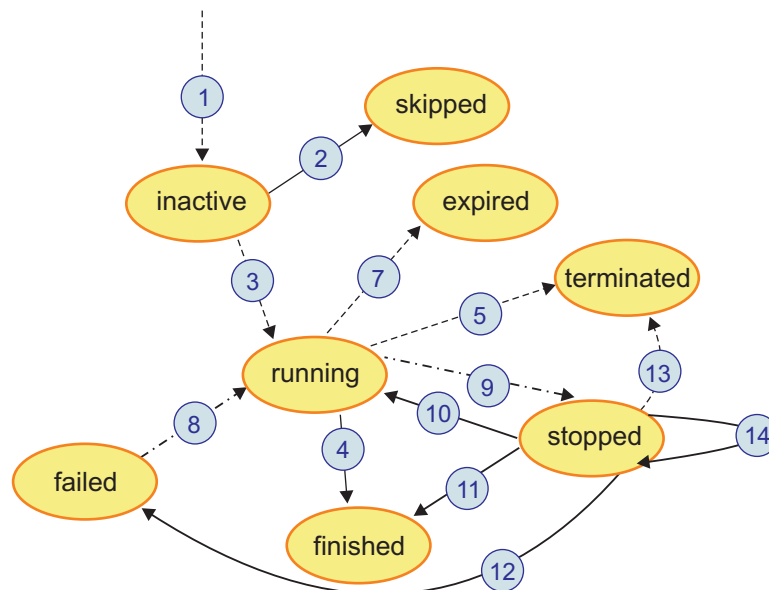


The activation of an invoke activity is the same as the activation of all of the other activity types (1), (2).

When an invoke activity runs normally through to completion, the activity is started and the state changes to running (3). If the service invocation returns successfully, the activity is put into the finished state (4).

As long as the service has not replied and the activity is in the stopped state, an administrator can force retry or force complete the activity. This can be useful if the service cannot reply, for example, because of a system outage. The state transitions from running to stopped (9), failed (8), and finished (4) can also be caused by the corresponding API. If the asynchronous service is a child process, then the activity cannot be force retried or force completed while it is in the running state.

As with Java snippet activities, an invoke activity can stop (9). It can then be repaired by administrative actions or terminated because the enclosing scope or process is also terminated (13).

Activities in the running state can expire if expiration is defined for the activity. The activity state is then expired (7) and a timeout fault is thrown. This fault can be handled by a fault handler.

If the enclosing scope of the activity is terminated, for example, because of a failure in a parallel path in the process, and the activity is in the running state, the activity is also terminated and put into the terminated state (5).

The state transitions for invoke activities with synchronous service calls are the same as those for Java snippets. The differences in the states and the state transitions between synchronous and asynchronous invocations are as follows:

- The running state for invoke activities with synchronous service calls is never visible.
- Expiration is not applicable for invoke activities with synchronous calls; the expired state can never be reached.
- An invoke activity with a synchronous service call is never terminated.

## State transition diagram for a human task activity

The following diagram shows the states and the state transitions that can occur during the life cycle of a human task activity.
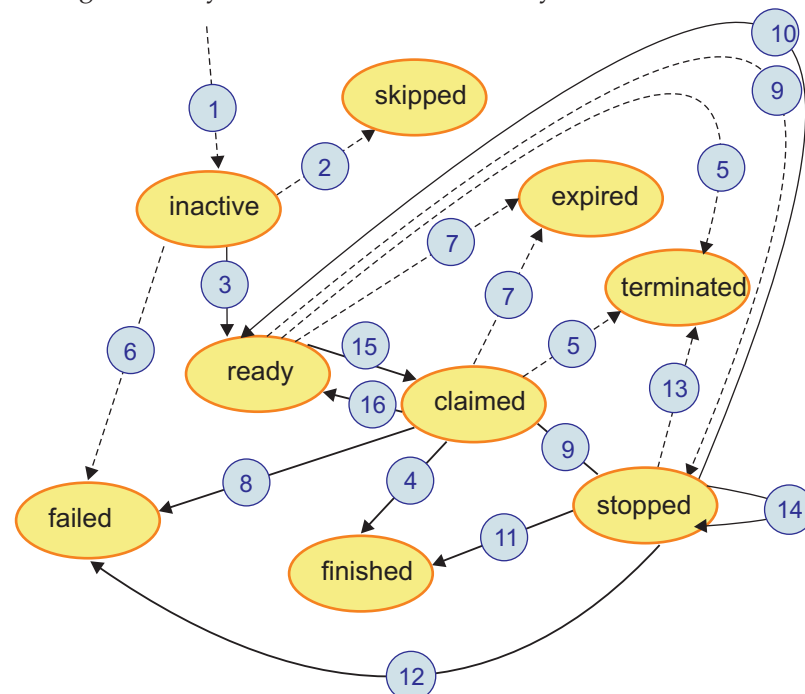
The runtime behavior of a human task activity is similar to that of an invoke activity. The running state of an invoke activity corresponds to the ready and claimed states of a human task activity. The ready state indicates that the activity is available to be worked on by a person. When someone claims the activity to work on it, the activity is put into the claimed state (15).

The person working on the activity provides the information that is required and completes the activity. The activity is then in the finished, failed, or stopped state. Alternatively, the person who claimed the activity might decide that it cannot be completed. The person then releases the activity for someone else to work on. In this case, the activity is returned to the ready state (16).

The other state transitions are the same as for invoke activities with asynchronous service calls.

**Related concepts**

"Fault handling and compensation handling in business processes" on page 25
A fault is any exceptional condition that can change the normal processing of a business process. A well-designed process should consider faults, and handle them whenever possible. Compensation is one way of handling faults.

# Invocation scenarios for business processes

A business process is an SCA (Service Component Architecture) component implementation type. It can expose services to other partners and consume services provided by other partners. A business process can be a service provider that is made available by the Business Process Choreographer APIs, an SCA service provider for other SCA service components, or an SCA client that invokes other SCA service components, including other business processes.

**Business processes as service providers that are made available by the Business Process Choreographer APIs**
You can use the Business Flow Manager API to instantiate business processes. Business Process Choreographer client applications also use this API to exploit business processes as service providers. These client applications can create and start business process instances, and query and manipulate existing process instances. The Business Flow Manager API is provided as an EJB, a Web service, and a JMS message interface that you can use to design EJB, Web service, and JMS clients.

**Business processes as SCA service providers for other SCA service components**
In this invocation scenario, a business process represents an SCA component that can be invoked by other SCA components acting as clients. As an implementation of an SCA component, services provided by a business process can be invoked from SCA clients and other SCA components. These mechanisms include:

- Wires for connecting an SCA client (reference) and the interface of a component that represents a business process
- SCA qualifier settings for component references and interfaces that determine aspects, such as interaction style, transaction behavior, and interaction reliability

**Business processes as SCA clients that invoke other SCA service components**
Conversely, because a business process is an implementation of an SCA component, services consumed by it can be provided by other SCA components or SCA imports. A BPEL partner link used in an outbound interaction is represented by an SCA reference. This reference can be wired

to other SCA components or imports, and SCA qualifiers can be used to associate quality-of-service attributes with the interaction.

**Business processes as SCA clients that invoke other business processes**
If both the SCA client and the SCA services are represented by business processes, you can select both of them on the SCA level and on the business process level. On the SCA level, you can use SCA wires to connect the SCA client to SCA services. On the business process level, you can associate partner links with the names of the business processes that act as service providers.

## Factors affecting business process interactions

A number of factors affect the behavior of business processes in the various invocation scenarios. These include the interaction style, the type of business process, the operation type, and the service endpoint resolution.

**Interaction style**
Operations provided by a business process can be invoked synchronously or asynchronously.

**Important:** Reasonable response times for synchronous interactions should not exceed a few seconds. If a request-response operation that is implemented by a business process does not return its results within a short time period, consider using an asynchronous interaction style to improve performance. A synchronous invocation of such operations results in blocked resources. It is also prone to timeout situations that are dependent on the system workload and are therefore non-deterministic.

**Business process type**
A business process can be a microflow or a long-running process. The characteristics of each of the process types affect invocation scenarios.

**WSDL operation type**
SCA references and SCA interfaces are associated with a WSDL port type containing one or more operations. Each operation can be a one-way or a request-response operation. A one-way operation implies a service execution the completion of which is not made known to the invoking client. The service execution ends with the successful invocation of the associated service. A request-response operation implies a service execution the completion of which is made known to the invoking client. The service execution ends when the result of the service execution is made available to the invoking client.

**Service endpoint resolution**
In the context of business processes, an invoking client can be associated with a service to be invoked in the following ways:

- An SCA wire statically associates an SCA reference to the interface of the invoked service. This is an SCA-level mechanism and it can be applied if the client side, the service side, or both are implemented as business processes.
- An endpoint reference can be set in the context of a partner link that is part of the business process acting as an SCA client. The endpoint reference uniquely determines the communication endpoint of the Web service that is to be invoked. Generally, any Web service can be invoked. An endpoint reference can be applied to either a Web service binding or an SCA binding.

- A business process template name can be set for a partner link that is part of a business process acting as an SCA client. The template name uniquely determines the name of another business process that is deployed in the same server or cluster.

**Related concepts**

"Business process types" on page 5
Business processes can be either long-running or microflows.

# Passing parameters 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 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 using data-typed variables.

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.

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 using data-typed variables.

If the invoked process modifies the BPEL variables, the input data from the calling service is also modified.

# Transactional behavior of business processes

Business processes are executed as part of transactions. The navigation of a business process can span multiple transactions in the case of long-running processes, or happen as part of one transaction in the case of microflows. Such navigation transactions can be triggered by external requests, internal messages, or responses from asynchronous services. When a transaction starts, the required activities are performed according to the process definitions. Invoked services can participate in the transaction.

### Related concepts

"Invocation scenarios for business processes" on page 16

A business process is an SCA (Service Component Architecture) component implementation type. It can expose services to other partners and consume services provided by other partners. A business process can be a service provider that is made available by the Business Process Choreographer APIs, an SCA service provider for other SCA service components, or an SCA client that invokes other SCA service components, including other business processes.

## Transactional behavior of microflows

Microflows are short-lived processes. They can run either in a transaction, or in an activity session as specified on the SCA component of the microflow. Microflows that are executed as part of a transaction are explained here.

Microflows are not interruptible. Therefore, a microflow cannot contain activities that wait for an external event, or for a user interaction, for example, human task activities.

Microflows are transient. The process instance state of a microflow is held in memory, and not stored in the runtime database. However, the state of a microflow instance can be persisted in the audit log or in Common Base Events.

The following diagram shows the transaction of a microflow and the services that the microflow interacts with. The services inside the transaction boundary participate in the microflow transaction; those outside the boundary do not participate in the transaction.
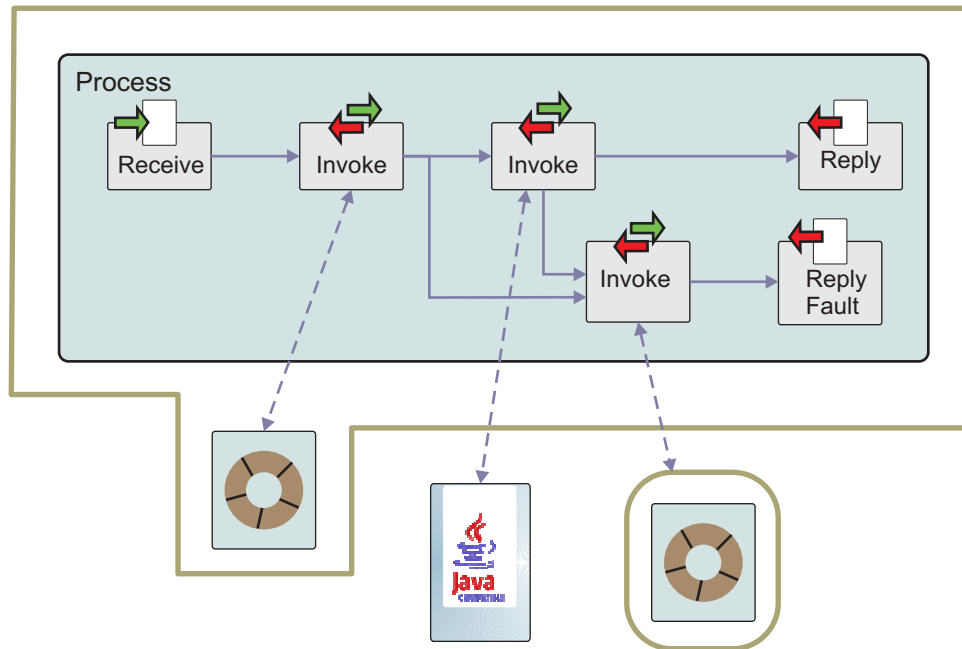
## Invoked services and microflow transactions

Although a microflow runs in one transaction, the execution of a microflow can involve more than one transaction. This is because a service that is called through an invoke activity can either participate in the transaction of the microflow, or it can run in its own transaction.

The following settings determine whether the service participates in the transaction of the microflow or runs in its own transaction.

- The interaction style that is used to call the service.

  The interaction style can be synchronous or asynchronous. The style is determined by the preferred interaction style of the target SCA component or the SCA import, and whether the operation is one-way operation or a request-response operation as shown in the following table:

*Table 1.*

| Preferred interaction style of the target component or import | One-way operation | Request-response operation |
|---|---|---|
| Any | Synchronous invocation | Synchronous invocation |
| Synchronous | Synchronous invocation | Synchronous invocation |
| Asynchronous | Asynchronous invocation | Synchronous invocation |

- The Service Component Architecture (SCA) transaction qualifiers that are specified for the process and the service that is called:
  - The **suspendTransaction** qualifier on the reference of the process component specifies whether the transaction context of the process is propagated to the services to be invoked.
  - The **joinTransaction** qualifier on the service interface specifies whether a service participates in the transaction of its caller if a transaction is propagated.

Based on these settings, the following rules apply to the invoked service:

**Synchronous invocation**

| joinTransaction | suspendTransaction = true | suspendTransaction = false |
|---|---|---|
| **joinTransaction = true** | The service does not participate in the microflow transaction | The service participates in the microflow transaction |
| **joinTransaction = false** | The service does not participate in the microflow transaction | The service does not participate in the microflow transaction |

If a service participates in a microflow transaction, the changes that are made by the service to the transactional resources are persisted only if the microflow transaction commits. If a service does not participate in the microflow transaction, the changes that are made by the service to the transactional resources might be persisted even if the transaction is rolled back. You can use compensation to undo the changes made by the service.

**Asynchronous invocation**

The service always runs in its own transaction. To ensure that the sending of the asynchronous SCA message participates in the current navigation transaction, the **asychronousInvocation** qualifier of the microflow must be set to `commit`.

**Related concepts**

"Compensation in business processes" on page 29
Compensation is the means by which operations in a process that have successfully completed can be undone.

## Transactional behavior of long-running processes

A long-running process spans multiple transactions. Each transaction is triggered by a Java Messaging Service (JMS) message or an external event. A transaction can trigger follow-on transactions by putting new messages in the messaging queue.

To allow navigation across transaction boundaries, the states of the process instance and its activity instances are persisted in the database.

The following diagram shows how each navigation step in a long-running process is performed in its own transaction. A navigation step can span multiple activities as shown by the invoke activity that calls a service. Also, multiple activities can be run in one transaction.
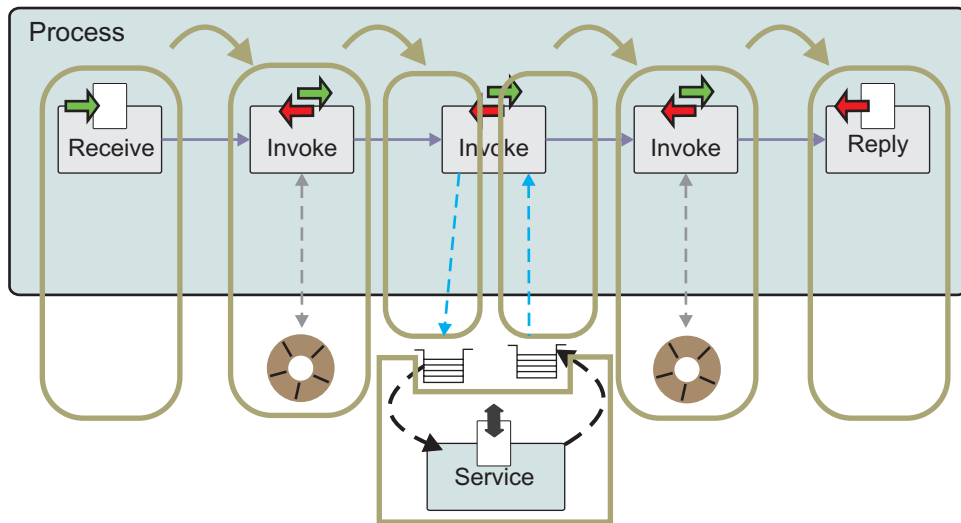
The following describes the transaction boundaries of a long-running process. You can influence transaction boundaries by using the transactional behavior attribute. However, Business Flow Manager can add or remove transaction boundaries at any time.

In general, a transaction boundary is needed in the following situations:

- When waiting for an external request, that is, upon reaching a receive activity or pick activity (also known as a receive choice activity) in the process navigation for which a corresponding request has not been received yet
- When scheduling a timer for a wait activity
- When invoking a service asynchronously using an invoke activity
- When invoking a human task activity

In addition, Business Flow Manager introduces transaction boundaries in the following situations. However, your process design must not rely on these boundaries as they can be overridden during process navigation, or they might change in the future.

- Upon receiving a request for a receive activity or pick activity that initiates the process
- When a fault is raised during process navigation
- Before and after an invoke activity is started that invokes a service synchronously, and this service does not participate in the transaction of the process
- When propagating life-cycle operations to child processes, for example, when a parent process is suspended, its child processes are suspended in subsequent transactions
- When the process instance is to be deleted automatically upon completion of the process
- When trying to recover from a failure that causes the rollback of a transaction spanning a series of activities
- Where specified using the transactional behavior attribute

If you need guaranteed transaction boundaries, it is a good practice to factor out the business logic that needs to be executed in a single transaction into a microflow, or *subprocess*. The logic of a microflow is always run in a single transaction.

## Influencing transaction boundaries

When you model a business process, you can suggest transaction boundaries for invoke, snippet, and human task activities by changing the transactional behavior attribute of the corresponding activity. The attribute can take one of the following values:

**Commit before**
> The current transaction is committed, and a new transaction is started. The activity with this attribute value becomes the first activity of the new transaction.

**Commit after**
> The activity participates in the current transaction. After the activity completes successfully, the transaction is committed, and a new one is started. A new transaction is started for each immediately following activity, and each subsequent activity becomes the first activity of one of these new transactions.

**Participates**
> The activity participates in the current transaction. Additional transaction boundaries are not set, neither before nor after the activity.
>
> In the following situations, this setting allows the transaction to continue with the navigation of the following activities depending on the values of their settings of the transactional behavior attributes.
> * If the invoke activity invokes the service asynchronously, the arrival of the response message triggers a new transaction. The transaction is very short because it commits immediately after the status of the invoke activity is updated.
> * In a sequence of human task activities, two transactions are needed for each human task activity, one to activate the human task activity and another to complete the human task activity. If you change the setting to `Participates`, you can reduce the number of transactions to one for each human task activity. This is because the completion of the previous human task activity, and the activation of the following activity is performed in the same transaction.
> * To enable server-controlled page flows that use the completeAndClaimSuccessor API.

**Requires own**
> The activity runs in its own transaction. This means that the current transaction is committed before the activity starts, and a new transaction starts after this activity completes.

The transactional behavior attribute is ignored if an invoke activity calls a synchronous service that does not participate in the current transaction. In this case, there is always a transaction boundary before the invoke activity is started, and after the invoke activity completes.

## Concurrent navigation of parallel branches in flow activities

To achieve concurrency in the navigation of parallel branches in a flow activity, a new transaction boundary is needed at the beginning of each branch so that each parallel activity is processed in a separate transaction. This means that the transactional behavior attribute of the first activity of each parallel branch must be set to `Commit before` or `Requires own` to achieve parallelism from the beginning of the flow.

**Note:** For Informix®, Oracle, and Derby database systems, navigation transactions for parallel branches in a process instance are serialized, that is they cannot be run in parallel. This is because the locks on database entities are not as granular as those for DB2® databases. However, services triggered asynchronously by such parallel branches still run in parallel; it is only the process navigation that is serialized for these database systems.

## Invoked services and transactions in long-running processes

A service that is called within a long-running process using an invoke activity can either participate in the current transaction of the long-running process, or it can run in its own transaction.

The following settings determine whether the service participates in the transaction of the long-running process or runs in its own transaction.
- The interaction style that is used to call the service.

   The interaction style can be synchronous or asynchronous. The style is determined by the preferred interaction style of the target SCA component or SCA import, and whether the operation is a one-way operation or a request-response operation as shown in the following table:

*Table 2.*

| Preferred interaction style of target component or import | One-way operation | Request-response operation |
|---|---|---|
| Any | Asynchronous invocation | Asynchronous invocation |
| Synchronous | Synchronous invocation | Synchronous invocation |
| Asynchronous | Asynchronous invocation | Asynchronous invocation |

- The Service Component Architecture (SCA) transaction qualifiers that are specified for the process and the service that is called:
  - The **suspendTransaction** qualifier on the reference of the process component specifies whether the transaction context of the process is propagated to the services to be invoked.
  - The **joinTransaction** qualifier on the service interface specifies whether a service participates in the transaction of its caller if a transaction is propagated.

Depending on the settings of the interaction style and the SCA qualifiers, the following rules apply to the invoked service:

**Synchronous invocation**

| joinTransaction | suspendTransaction = true | suspendTransaction = false |
|---|---|---|
| joinTransaction = true | The service does not participate in the transaction of the long-running process | The service participates in the transaction of the long-running process |
| joinTransaction = false | The service does not participate in the transaction of the long-running process | The service does not participate in the transaction of the long-running process |

If a service participates in the current transaction of the long-running process, the changes that are made by the service to the transactional resources are persisted only if the current transaction commits.

**Asynchronous invocation**
The service always runs in its own transaction. To ensure that the sending of the asynchronous SCA message participates in the current transaction, the **asychronousInvocation** qualifier of the long-running process must be set to `commit`.

### Recovery of a successful service invocation when a transaction rolls back

The recovery behavior depends on whether the called service participates in the current transaction.

An invoke activity calls a service that participates in the current transaction. The execution of the service is complete. If an error occurs after completion of the service and the transaction is rolled back to the state that the process was in before the transaction started, the effect of the called service is also rolled back. When the transaction is retried, the service is called again.

In contrast, if the called service does not participate in the current transaction and the called service returns a response, the response is stored in a separate transaction. If an error occurs after the response is stored, the current transaction is rolled back and the transaction is retried. During the retry the service is not called again, however, the stored response is restored and the navigation continues.

# Fault handling and compensation handling in business processes

A fault is any exceptional condition that can change the normal processing of a business process. A well-designed process should consider faults, and handle them whenever possible. Compensation is one way of handling faults.

**Related concepts**

"State transition diagram for activities" on page 10
The state of an activity instance changes when a significant step in the execution of the activity instance occurs. The states and the state transitions depend on the type of activity.

## Fault handling

When a fault occurs in a process, the navigation moves to the fault handler. Fault handlers can be specified on invoke activities, scopes, and on the process.

A fault handler can catch a specific fault name, fault type, or both. When a fault occurs, Business Flow Manager tries to match the fault with a fault handler. It

looks for a fault handler on the enclosing scope or on the activity where the fault occurred. It uses the following rules to select a fault handler:

- If the fault does not have any associated fault data, Business Flow Manager uses a fault handler with the matching fault name. Otherwise, it uses the catch-all fault handler if one is available. A fault without any data cannot be caught by a fault handler that has a fault variable defined.
- If the fault has associated fault data, Business Flow Manager uses a fault handler with the matching fault name and a fault variable with a type that matches the type of the fault data. If a fault handler is not found that matches the name and fault data type, it uses a fault handler without a fault name and a fault variable with a type that matches the type of the fault data. If a suitable fault handler cannot be found, it uses the catch-all fault handler if one is available. A fault with data cannot be caught by a fault handler that does not have a fault variable defined.

If a fault is raised that does not match any of these fault handler definitions, the default fault handler is started. The default fault handler is not specified explicitly. The default fault handler runs all of the available compensation handlers for the immediately enclosed scopes in the reverse order of completion of the corresponding scopes, and rethrows the fault to the next level, that is, the enclosing scope or the process. On this next level, Business Flow Manager again tries to match the fault to the fault handlers that are available.

If neither specific fault handlers nor a catch-all fault handler anywhere in the fault handler chain catches the fault, the fault reaches the process scope, and the process ends in the failed state. Even if a fault handler catches the fault on the process scope and handles it, the process still ends in the failed state.

## Designing fault handlers

When you design a fault handler, consider the following options:

- Catch a fault and try to correct the problem so that the business process continues through to normal completion.
- Catch a fault and find that it is not resolvable at this scope. In this case, you have the following additional options:
  - Throw a new fault.
  - Rethrow the original fault so that another scope can handle it.
  - If this is a request-response operation, reply with a fault.
  - Invoke a human task to correct the issue. If the fault handler cannot resolve the issue, you might need to roll back the process and compensate it.
  - For long-running processes, also consider using the **continueOnError** parameter on the process to handle the fault administratively.

## Raising faults

You can raise faults using throw and rethrow activities, or programmatically using a Java snippet activity.

To propagate faults to the caller of the process, you use the reply activity with a fault specification.

### Raising a fault using throw and rethrow activities

A throw activity in a business process can throw any type of fault, including standard faults, but the intended usage pattern is to throw business faults. An

exception thrown by a throw activity must be caught and handled within the business process. If a process with a request-response interface does not handle a fault in the process, the process ends with a bpws:missingReply standard fault. For a client application, this fault is returned in a StandardFaultException object.

You cannot return a business or standard fault with a throw activity. You must use a reply activity to return a business fault to the process client. A reply activity can only return a business fault that is defined on the interface that the process implements.

A rethrow activity can be used in a fault handler to rethrow the fault to the next enclosing scope. This might be useful when you want to do some fault handling on the current scope, such as triggering specific compensation handlers, and still want to make the enclosing scopes aware of this issue. You can also use a rethrow activity when the current fault handler cannot handle the fault, and you want the fault to be propagated to a fault handler that is defined on one of the enclosing scopes, or on the process.

The rethrow activity can only be used within a fault handler because existing faults can be rethrown only from fault handlers.

## Raising faults programmatically

You can raise faults programmatically in a Java snippet in a business process using the raiseFault method. You can raise a business fault in one of the following ways:

- `raiseFault(QName fault, String variableName);`
- `raiseFault(QName fault);`

The following example creates a fault called IncompleteData in the http://process/UpdateCustomerRecordProcess/Interface0/ namespace, and then throws this fault from a Java snippet.

```
javax.xml.namespace.QName fault = new javax.xml.namespace.QName
    ("http://process/UpdateCustomerRecordProcess/Interface0/", "IncompleteData");
raiseFault(fault);
```

If the thrown fault is not one that is declared on any WSDL interface, then specify the target namespace of the process as the namespace of the fault. You can then use a catch activity to catch this fault in a business process.

Do not throw a ServiceBusinessException object directly, but use the raiseFault message to do so.

## Using reply activities to provide a fault to the caller

The reply activity with a fault specification propagates the specified fault to the caller of the request-response operation. The reply activity can only return a fault that is defined on the interface that the process implements. This is useful when the business process cannot properly respond to the caught fault, but the process initiator can respond to it. For example, if the caller passes an account number that is not found by the business process, the process should reply to this service call with an AccountNotFound fault.

A reply activity with a fault specification does not complete the process and return to the caller immediately. The navigation of the process continues until it reaches an end state.

## Retrieving information about a fault

Your process must be able to handle system faults. You can catch system faults using either a fault handler that is defined to catch the runtimeFailure standard fault, or a catch-all fault handler. In some situations, you might need the information provided with the fault.

You can use one of the following constructs to retrieve this information:
- A fault variable that stores data in the event of a standard or system fault. To make use of typed variables to handle faults, you must manually create a StandardFaultType complex type.
- A catch-all fault handler. The catch-all fault handler does not have an associated variable. You can retrieve the fault data from a catch-all fault handler using the getCurrentFaultAsException method in a Java snippet activity. You must include this Java snippet activity in the catch-all fault handler. You can use the getCurrentFaultAsException method to retrieve data for any type of fault, not only system faults.

  The getCurrentFaultAsException method returns the fault as an exception object of the com.ibm.bpe.api.BpelException type. The BpelException object provides several operations to get more information about the fault, such as the fault name. The BpelException object wraps the exception instance. Thus, you can access the fault message and the root exception, as the following example shows:

  ```
  com.ibm.bpe.api.BpelException bpelexception =
  getCurrentFaultAsException();
  System.out.println("Fault  Name" +
  bpelexception.getFaultName())
  bpelexception.printStackTrace( System.out);
  Throwable rootCause = bpelexception.getRootCause()
  ```

## Continue on error behavior

This parameter is available for some activities when you define the process in WebSphere Integration Developer. It determines what happens to a process if a fault is raised by those activities and a fault handler is not defined for that fault.

If a fault is detected in an activity, then fault handling of the process is started. If the fault is not handled by the immediately enclosing scope and the **continueOnError** parameter is not set, then the activity is stopped and an administrative action is made available to continue the process. You can use the Business Process Choreographer APIs or the Business Process Choreographer Explorer to find the running process instance with the stopped activity. The activity can be either restarted or forced to complete. If the **continueOnError** parameter is set, then the standard fault handling is applied.

For example, you can use the forceRetry method to start an activity again with different input data, or the forceComplete method to complete an activity with output or fault data.

```
public interface BusinessFlowManagerService {
   public void forceRetry(String aiid, ClientObjectWrapper inputMessage,
         boolean continueOnError);
   ...
}
```

If you want the process to stop if the error occurs again, you must disable the **continueOnError** parameter.

When the completion of an activity is forced using the forceComplete method, it is not run again. Its output message is used to continue the process navigation.

# Compensation in business processes

Compensation is the means by which operations in a process that have successfully completed can be undone.

Compensation processing is a means of handling faults in a running process instance for which compensation is defined in the process model. Compensation reverses the effects of operations, which were committed up to when the fault 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 can be defined on the scope or process 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 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 transaction. When a process instance runs, undo actions for compensable activities are registered with the enclosing unit of work. Depending on the outcome of the 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 a fault occurs during compensation processing, the compensation action requires manual resolution to overcome the fault. You can use Business Process Choreographer Explorer to repair these compensation actions.

**Related concepts**

"Transactional behavior of microflows" on page 19
Microflows are short-lived processes. They can run either in a transaction, or in an activity session as specified on the SCA component of the microflow. Microflows that are executed as part of a transaction are explained here.

**Related information**

Using compensation in processes with Business Process Choreographer

# Recovery from infrastructure failures

Business Flow Manager provides a facility for handling temporary infrastructure failures.

A long-running process spans multiple 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 request
- Navigate according to the request
- Store state
- 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 message that cannot be processed is assumed to be damaged.

The responses to these causes are as follows:

| Cause | Response |
|-------|----------|
| Unavailable infrastructure | In normal processing mode, the message-driven bean tries, for a specified time, to recover from that situation. It tries to keep all of the messages available until the infrastructure is operational again. 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. 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 message is stored in the retention queue from where it is replayed automatically if a subsequent message can be processed successfully.
- 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.

## Retry limit

The retry limit defines the maximum number of times that a message can be transferred to the retention queue before being put in the hold queue.

To be put in the retention queue, the processing of a message must fail three times.

For example, if the retry limit is 5, a message must go to 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 in the hold queue. This means that a message must fail (3 * *RetryLimit*) + 2 times before it is put in the hold queue.

In a performance-critical application running in a reliable infrastructure, the retry limit should be small: one or two, for example. If the retry limit is set to zero, a repeatedly failing message is retried three times and then it goes immediately into the hold queue.

This Business Flow Manager property is specified in the administrative console, click **Servers** ⟶ **Application Servers** ⟶ *server_name*. Then, in the **Container Settings** section, click **Business Process Choreographer Container Settings** ⟶ **Business Process Choreographer Containers**. In the **Additional Properties** section, click **Business Flow Manager**.

## Retention queue message limit

The retention queue message limit defines the maximum number of messages that can be in the retention queue. If the retention queue overflows, the system goes into quiesce mode. To make the system enter quiesce mode as soon as a message fails, set the value to zero. To make the Business Flow Manager more tolerant of infrastructure failures, increase the value.

This Business Flow Manager property is specified in the administrative console, click **Servers** ⟶ **Application Servers** ⟶ *server_name*. Then, in the **Container Settings** section, click **Business Process Choreographer Container Settings** ⟶ **Business Process Choreographer Containers**. In the **Additional Properties** section, click **Business Flow Manager**.

## Retention queue

The retention queue holds failed messages that are replayed by moving them back to the internal work queue of the Business Flow Manager. A message is put in the retention queue if it fails three times. If the message fails (3 * *RetryLimit*) + 2 times, it is put in the hold queue. 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 by querying and replaying failed messages.

## Hold queue

The hold queue contains messages that have failed (3 * *RetryLimit*) + 2 times. The administrator can move the messages in this queue back to the internal queue by 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

Human Task Manager needs neither a retention queue, nor retry limits. It has only a hold queue, in which failed messages are placed, and from which, they can be replayed.

**Related tasks**

"Querying and replaying failed messages, using the administrative console" on page 285
This describes how to check for and replay any messages for business processes or human tasks that could not be processed.

"Querying and replaying failed messages, using administrative commands" on page 303
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.

"Refreshing the failed message counts" on page 286
Use the administrative console to refresh the count of failed messages for business processes or human tasks.

## Authorization and people assignment for processes

Authorization is used to assign specific privileges to particular users or particular groups of users. It determines what actions users are allowed to take on processes and activities.

The authorization for business processes is realized using human tasks. Authorization roles are used to define the sets of actions that are available to specific roles. The roles that are specified for the human task are inherited by the associated business processes and activities. So, for example, if you model an inline human task in a business process, the owner of the task automatically becomes the activity owner. Each activity role matches exactly one human task role. Business Flow Manager uses the activity roles for navigation and authorization.

## Authorization roles for business processes

A role is a set of people who share the same level of authorization. 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 concepts**

"People resolution" on page 77
People resolution retrieves user information from people directories based on a set of parameterized query expressions, known as people assignment criteria.

"Authorization roles for human tasks" on page 64
Actions that you can take on human tasks depend on your authorization role. This role can be a system-level J2EE role or an instance-based role.

## J2EE roles for business processes

J2EE roles are set up when Business Process Choreographer is configured. For J2EE role-based authorization, you must have a user registry configured and global security enabled.

The following Java 2 Platform, Enterprise Edition (J2EE) roles are supported for processes:

- BPESystemAdministrator. Users assigned to this role have all privileges. This role is also referred to as the system administrator for business processes.
- 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.
- JMSAPIUser. Business Flow Manager JMS API requests are run on behalf of the user ID this role is mapped to, regardless of who the caller is.

You can use the administrative console to change the assignment of users and groups to these roles.

## Instance-based roles for business processes and activities

A set of predefined authorization roles is provided for processes and activities. You can assign these roles to processes and activities when you model the process. The association of users to instance-based roles is determined at runtime using people resolution.

### Authorization roles for actions on processes

The people assigned to process roles are authorized to perform the following actions:

| Role | Authorized actions |
|------|--------------------|
| 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. Members of this role also automatically become the reader for activities, and the inline to-do tasks (including the subtasks, follow-on tasks, and escalations) that are associated with human task activities. |
| Process administrator | Administer process instances, intervene in a process that has been initiated, and create, delete, and transfer work items. Members of this role also automatically become the administrator for activities, and the inline to-do tasks (including the subtasks, follow-on tasks, and escalations) that are associated with human task activities. |
| Process activity administrator | Administer the activities of a process. |

The process starter is a role that is used by Business Flow Manager for process navigation and the invocation of external services. If a process instance still exists in the database, do not delete the user ID of the process starter from your user registry so that the navigation of the process can continue.

Users are assigned to these roles using human tasks.

| Role | People assignment |
|---|---|
| Process starter | The process starter can be specified by assigning an inline human task to the initiating receive or pick (receive choice) activity of a process. |
| Process reader | The process reader is specified by setting the reader role on the administration task that is associated with the process. This role is inherited by all of the activities in the process. |
| Process administrator | The process administrator is defined by an administration task that is assigned to the process. This role is inherited by all of the activities in the process. |
| Process activity administrator | The process activity administrator is defined by an administration task that is associated with the process. The administrator role defined on this task is also used as the process activity administrator. **Note:** This administration task is different from the one that is used to determine the process administrator. <br><br> The activity administration task that is defined on the process level is the default administration task for activities that do not have an administration task defined. |

## Authorization roles for actions on activities

When you model a human task and include it as a human task activity in a business process, the owner of the task automatically becomes the activity owner. Members of roles that are defined for a human task inherit the same role on the corresponding human task activity. Business Flow Manager uses the activity roles for navigation and authorization. The potential starters of an inline invocation task are the potential starters of the associated receive or pick (receive choice) activity, or the event handler.

The instance-based roles for activities 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 their work items to an administrator or a potential owner. |

| Role | Authorized actions |
|------|--------------------|
| Activity administrator | Repair activities that are stopped due to unexpected errors, and force complete long-running activities. |

### Default people assignments for process roles

Default people assignments are performed if you do not define people assignment criteria for certain roles, or if people resolution fails or returns no result. The following table shows which defaults apply.

| Roles for business processes | If the role is not defined in the process model ... |
|------------------------------|-----------------------------------------------------|
| Process administrator | Process starter becomes process administrator |
| Process reader | No reader |

In addition, if you do not define an invocation task to create and start the business process, the default people assignment criteria, **Everybody**, is used for the potential starters of the process.

## Authorization for creating and starting business processes

The set of users that are allowed to create and start a process is determined by the invocation task that is associated with the receive or pick (receive choice) activity that is used to create and start a new process instance, and also by the administration task that is associated with the process. The business process inherits the roles that you assign to these tasks.

You can use human tasks in the following ways to create and start business processes:

- Assign an inline invocation task to the initiating receive or pick (receive choice) activity of the process

  Some business processes might change sensitive business data and therefore only authorized personnel should be authorized to create and start these processes. For this type of business process, you can assign a human task to the initiating receive activity of the process by specifying an inline invocation task for the process template. The potential starters defined for the inline invocation task become the potential starters of the process.

  The process can be started either by creating and starting the invocation task using the Human Task Manger API, or by initiating the process using the Business Flow Manger API. Both ways result in the same authorization checks. If an inline task is not specified, anyone can start the process.

- Assign a stand-alone invocation task to the initiating receive or pick (receive choice) activity of the process

  You can also use a stand-alone invocation task that is wired to the business process to perform authorization checks when a process is started. However, consider the following points if you use a stand-alone invocation task:

  – The authorization check is done only if the process is started by the invocation task, that is, the check can be omitted when the process is started using the Business Flow Manager API or an SCA client that is directly wired to the process component.

- It uses the SCA infrastructure to invoke the process while an inline task uses an internal interface. An inline invocation task therefore performs better than a stand-alone task.
- You have no access to the process context in the people assignment criteria definition. This means that stand-alone tasks do not support dynamic people assignments based on the process context.

- Assign an administration task to the process.

  The administrator role of the administration task is inherited by the process. A process administrator can create and start a process.

## Authorization for administering business processes

You can use administration tasks to authorize a user, or group of users, to perform administrative actions on business processes and their associated activities

### Process administration

To define which users are allowed to perform administrative actions and to read the process data, you can specify an administration task as part of a long-running business process. The administrator and reader roles for the administration task determine who is the process administrator and the process reader. The process administrator can, for example, force terminate the process instance. An administration task is associated with every business process. If an administration task is not modeled in WebSphere Integration Developer for the process, a default administration task is created at runtime. This default task defines the process starter as the process administrator, and does not assign any readers to the process.

### Activity administration

You can model an administration task for each invoke or snippet activity. This task determines who is allowed to administer the activity in addition to the process administrators. You can also model a default administration task for activities on the process level that applies to every invoke or snippet activity that does not have an administration task assigned to it. The administrator role for an activity administration task determines who is allowed to administer the corresponding activity. The activity administrator and the process administrator can, for example, force retry the activity.

Invoke activities have an administration task associated with them. For synchronous invoke activities, this task is created only when the activity is stopped after an invocation failure. The administration task is then used to handle repair requests, such as force complete and force retry. For asynchronous invoke activities, the administration task is always created. Thus, an administrator can force retry or force finish the activity while the activity waits for the asynchronous response.

# Chapter 2. About human tasks

A human task is a component that allows people and services to interact.

Some human tasks represent to dos for people. These tasks can be initiated either by a person or by an automated service. Human tasks can be used to implement activities in business processes that require human interactions, such as manual exception handling and approvals. Other human tasks can be used to invoke a service, or to coordinate the collaboration between people. However, regardless of how a task is initiated, a person from a group of people, to which the task is assigned, performs the work associated with the task.

People are assigned to human tasks either statically, or by specifying criteria, such as a role or a group, that are resolved at runtime using a people directory. Alternatively the input data of a human task, or the data of a business process is used to find the right people to work on a task.

**Related tasks**

Chapter 9, "Administering business processes and human tasks," on page 333
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.

"Administering task templates and task instances" on page 344
Use the administrative console or the administrative commands to administer task templates. Use Business Process Choreographer Explorer to work with task instances.

"Stopping and starting task templates with the administrative console" on page 344
Use the administrative console to start and stop each installed task template individually.

"Stopping and starting task templates with the administrative commands" on page 345
Administrative commands provide an alternative to the administrative console for stopping and starting task templates. Use the administrative commands to stop all task templates within an enterprise application.

"Creating and starting a task instance" on page 346
You can create and start a task instance from any of the task templates that you are authorized to use.

"Working on your tasks" on page 346
To work on a task, you must claim the task and then perform the actions that are needed to complete it.

"Suspending and resuming task instances" on page 347
You can suspend task instances. You might want to do this, for example, to fix a problem that is causing the task instance to fail. When the prerequisites for the task are met, you can resume running the task instance.

"Managing priorities of human tasks" on page 348
You can use the priorities of human tasks to filter for tasks, and to sort your list of tasks.

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.

If you intend to be away from the office for a certain time, specify a substitute for your tasks.

If users are prevented from working on their tasks, for example, if they are on sick leave, specify a substitute for the user's tasks and a date range for which the delegation is applicable.

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.

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.

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 people directory 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.

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.

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. You can set up your system to send e-mails to designated people to inform them about the escalation.

## Task templates

A human task template contains the definition of a deployed task model that was created using WebSphere Integration Developer, or at runtime using the Business Process Choreographer APIs.

The template contains properties, such as the task name and priority, and aggregates artifacts, such as escalation templates, custom properties, and people query templates. In addition to the properties that are specified when the task template is modeled, an installed task template can also have one of the following states:

**Started**

> When a task template is started, new instances of the template can be started.

**Stopped**

> The task template must be stopped before the human task application can be uninstalled. When a task template is in the stopped state, no new instances of this template can be started.

You can model to-do or collaboration tasks at runtime by creating instances of the com.ibm.task.api.TaskModel class. You can then use these instances to either create a reusable task template, or directly create a run-once task instance. Modeling human tasks at runtime is based on the Eclipse Modeling Framework (EMF).

### Related tasks

"Creating task templates and task instances at runtime" on page 454
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.

## Kinds of human tasks

The task kind is derived from the task template kind that is assigned during modeling.

The kinds of human tasks are as follows:

**To-do (participating) tasks**

> Support Web-service-to-person interactions, which enable a person to implement a service. For example, a to-do task can be a human task activity in a business process.
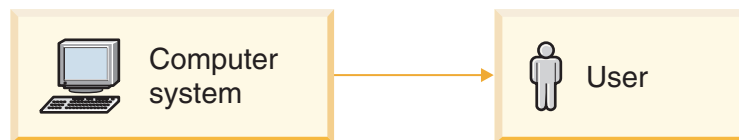


**Administration tasks**

> Administration tasks are used by administrators to solve technical problems that occur in processes. Currently, you can use administration tasks for business processes only.



**Invocation (originating) tasks**

> Support person-to-Web-service interactions, which enable people to create and start services. For example, a user can start a business process, or send it an event by means of an invocation task.
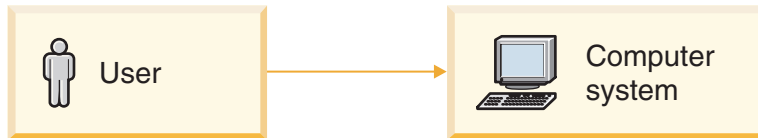
**Collaboration (pure human) tasks**

Support person-to-person interactions, which enable a person to share work with other people in a structured and controlled way. Collaboration tasks do not interact directly with business processes or other Web services.



# Versioning of human tasks

Use versioning when you want alternative instances of the same human task to co-exist in the runtime environment.

You can include versioning information when you model the stand-alone human task in WebSphere Integration Developer. The version of a task template is determined by its valid-from date. This means that different versions of a task template can have the same task template name, but they have different valid-from dates. The version of a task template that is used at runtime is determined by whether the task 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 task template is used is made either during modeling, or when the task model is deployed. The calling component invokes a dedicated, statically-bound task template according to the Service Component Architecture (SCA) wiring. Even if another version of the task template exists that is valid according to its valid-from dates, the current statically wired task template is used and all of the other versions are ignored.

An example of early-binding is an SCA wire. If you wire a stand-alone reference to a human task component, every invocation of the task template using this reference is targeted to the specific version that is represented by the human task component.

**Late binding**

In a late-binding scenario, the decision on which human task template is used is determined when the task instance is created. In this case, the version of the task template that is currently valid is used. A newer version of a task template supersedes all of the previous template versions. Existing task instances continue to run with the task template with which they were associated when they started. This leads to the following categories of task templates:

- Currently valid task templates are used for new task instances
- Task templates that are no longer valid might still be valid for running task instances
- Task templates that become valid in the future according to their valid-from date

An example of late binding is when a new task is invoked in Business
Process Choreographer Explorer. The instance that is created is always
based on the most recent version of the task template with a valid-from
date that is not in the future. Follow-on tasks and subtasks are always
invoked using late binding.

# Task instances

A task instance is a runtime occurrence of a task template.

# Stand-alone and inline tasks

The Service Oriented Architecture (SOA) patterns recommend the realization of
software solutions with a set of loosely coupled components. The human tasks that
follow the SOA patterns are called *stand-alone tasks*, while the human tasks that are
defined as part of a business process are called *inline tasks*.

The following table shows the task kinds that are available for stand-alone and
inline tasks:

*Table 3.*

| Implementation | Invocation task | To-do task | Collaboration task | Administration task |
|---|---|---|---|---|
| Stand alone | Yes | Yes | Yes | No |
| Inline | Yes | Yes | No | Yes |

## Stand-alone tasks

Stand-alone tasks follow the SOA pattern and they are loosely coupled with the
components that invoke them (to-do tasks), or the components that are invoked by
them (invocation tasks). They can be wired to another component using the Service
Component Architecture (SCA) infrastructure.

They communicate with their partner components exclusively with SCA means.
That is, to-do tasks receive input messages and return output or fault messages,
and invocation tasks send input messages and receive output or fault messages.
No further information exchange or life cycle control happens.

Because stand-alone tasks are modeled separately, they can be reused. Stand-alone
tasks always emit their Common Event Infrastructure (CEI) and audit log events as
Human Task Manager events.

Stand-alone tasks are made available as SCA components in the following ways:
- To-do tasks have an interface that can be wired to a client component.
- Invocation tasks have a reference that can be wired to the service to be invoked.
- Collaboration tasks are self-contained SCA components. Although collaboration
  tasks are stand-alone tasks, they have two human interfaces and therefore
  cannot be wired to a service component.

Administration tasks are available neither as stand-alone tasks nor as SCA
components.

The following rules apply to stand-alone tasks that are used with a business
process:

- Their life cycle is independent from the process.
  - By default, to-do tasks are created by the process, and they are deleted when the process is deleted.
  - Invocation tasks can create the process. However, they are not deleted when the process is deleted so that the results of the task can be seen.
- A to-do task is visible as an invoke activity in the business process.
- Invocation tasks are wired to receive or pick activities (also known as receive choice activities), or to an on-event event handler.
- Task descriptions, display names, and documentation support multiple languages in parallel.
- Stand-alone tasks have no access to the process context. They cannot access process variables, custom properties, or data from other process activities.

If a collaboration task is a top-level task, its life cycle is managed independently, and it is either deleted manually, or automatically after a specified length of time. If a collaboration task is a subtask or a follow-on task, its life cycle is managed by its parent or top-level task.

## Inline tasks

Inline tasks are an integral part of the business process. Inline tasks can be to-do tasks, invocation tasks, and administration tasks. Because collaboration tasks leverage the interaction between people and do not directly interact with processes, they cannot be inline tasks. Inline tasks are neither visible as SCA components (cannot be wired), nor are they reusable in other processes or activities.

Inline tasks have access to the process context and process data, such as process variables, custom properties, and activity data. This can be useful for tasks that involve the separation of duties. Inline to-do tasks emit their CEI and audit log events as Business Flow Manager human task activity events. Their subtasks and follow-on tasks emit events as Human Task Manager events.

The following rules apply to inline tasks:
- To-do tasks are human task activities in the process. They share the same state, but the human task activity does not reflect the task substates.
- Invocation tasks are associated with receive or pick (receive choice) activities, or on-event event handlers.
- Administration tasks are attached either to the process, or to an activity in the process.
- The life cycle is usually determined by the process.
  - To-do tasks and administration tasks are created by the business process, and deleted with the process.
  - If invocation tasks are created and started by the business process, their life cycle is determined by the process, and they are deleted with the process. If they are created and started using the Human Task Manager API, their life cycle is independent of the process, and their results can be displayed even after the process is deleted.
- To-do and invocation task descriptions, display names, and documentation support only one language.
- Inline tasks have no duration until expiration. However the human task activity that corresponds to a to-do task can have an expiration defined.

- Only inline invocation tasks have a duration until deletion, but it applies only if the task is started using the Human Task Manager API.
- The update action on inline tasks supports only a subset of task properties. Only task properties that have no representation in the process or activity can be updated. For more information on the update method, see the Javadoc for the HumanTaskManager interface in the com.ibm.task.api package.

Inline tasks are used for process authorization:
- The roles reader, administrator, potential owner, owner, and editor of a to-do task are identical to the corresponding roles of the human task activity in the process.
- The potential starter role of an inline invocation task determines who is allowed to invoke and send messages to the corresponding receive or pick (receive choice) activity, or on-event event handler. Note that the potential starter and potential instance creator roles have identical people assignments. If an inline invocation task is not defined, everybody is authorized to start the activity or event handler.
- The administrator and reader roles for a process administration task determine who is the process administrator or the process reader. The process administrator can, for example, force terminate the process instance.
- The administrator role for an activity administration task determines who is allowed to administer the corresponding activity. The activity administrator and the process administrator can, for example, force retry the activity.
- The process reader and process administrator authorization are inherited by every process activity or inline human task.

## The relationship of human tasks to business processes

Invocation tasks can be associated with receive or pick (receive choice) activities, or on-event event handlers. These tasks can be both inline or stand-alone tasks. If you are using the Business Flow Manager API, only inline invocation tasks can influence the authorization for invoking the receive activity. By default, everybody is allowed to send a message to receive or pick activities, or to on-event event handlers. This includes invoking a business process in the case of initiating receive activities.

An administration task is associated with every business process. The administration task determines who is authorized to administer and read the process. If an administration task is not modeled in WebSphere Integration Developer for the process, a default ad-hoc task is created at runtime. This default task defines the process starter as the process administrator and assigns no readers to the process.

You can model an administration task for each invoke or snippet activity. This task determines who is allowed to administer the activity in addition to the process administrators. You can also model a default activity administration task that applies to every invoke or snippet activity that has no explicit administration task assigned to it.

Invoke activities have an administration task associated with them. For snippet activities and synchronous invoke activities, this task is created only when the activity is stopped after an invocation failure. The administration task is then used to handle repair requests, such as force finish and force retry. For asynchronous

invoke activities, the administration task is always created. Thus, an administrator can force retry or force finish the activity while the activity waits for the asynchronous response.

Stand-alone to-do tasks can implement asynchronous invoke activities. These activities also have an administration task associated with them. Inline to-do tasks implement human task activities. An administration task is created for these activities at runtime.

**Related concepts**

"Life cycle of human tasks" on page 51
Human tasks support people when they interact with Web services or business processes. The interactions that can take place over the lifetime of a task depend on whether the task is a to-do task, a collaboration task, a invocation task, or an administration task. Certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the human task.

"Instance-based roles for business processes and activities" on page 33
A set of predefined authorization roles is provided for processes and activities. You can assign these roles to processes and activities when you model the process. The association of users to instance-based roles is determined at runtime using people resolution.

# Subtasks

Subtasks support people when they need to delegate parts of their assigned work to other people, but want to keep control over the overall result. They can also be used to invoke supporting services to help people accomplish the tasks that they are working on.

Subtasks can be created from stand-alone task templates that are stored in the Business Process Choreographer database, from task templates created at runtime, or by providing a new task model at runtime. The parent task can be a to-do task or a collaboration task, and it must have the **supportsSubtask** attribute set to `true`. The subtasks that are created can be either collaboration tasks or invocation tasks. These subtasks can, in turn, have subtasks or follow-on tasks.

There are no restrictions on either the input message type or the output message type. However, the starter of the subtask must provide an input message. When the subtask is finished, the owner of the parent task can map the subtask output data to the output message of the parent task.

## Authorization considerations

In addition to what is specified for subtask when it is started, the subtask also inherits the authorization roles from its parent task:

- The readers, editors, originator, and owner of the parent task become readers of the subtask and its escalations
- Administrators of the parent task become administrators of the subtask
- Escalation receivers of the parent task become readers of the subtask

## Life cycle considerations

When the first subtask is started, the parent task enters the waiting-for-subtask substate. It remains in this substate until the last subtask reaches one of the end

states finished, failed, expired, or terminated. Some life cycle operations (state changes) of the parent task are propagated to its subtasks. So, when the parent task is suspended, resumed, terminated, deleted, or it expires, all of its subtasks are also suspended, resumed, terminated, deleted, or expire. The escalated substate of parent tasks is not propagated; subtasks are not escalated when the parent task is escalated. Subtasks have their own escalations and their escalated substate is set only when one of their own escalations is triggered.

Some life cycle operations on a subtask can conflict with the life cycle operations of the parent task, and are therefore not allowed. These are mainly operations that influence the end of the life cycle of a subtask and need coordination with the parent. The following operations can be performed on subtasks:

- Life cycle operations that do not conflict with the parent task are always supported. These are operations, such as claim, cancel claim, complete, creation and start of subtasks or further follow-on tasks.
- Subtasks can expire.
- Subtasks can be suspended and resumed because work on a subtask might need to be stopped although work on the parent task continues.
- Subtasks can be terminated.
- Subtasks can have their own escalations so that the parent task owner and the subtask originator can better control the progress of the subtask.
- Auto-deletion settings are ignored for tasks that are started as subtasks. Subtasks are deleted when their parent task is deleted. The deletion of individual subtasks using the Business Process Choreographer APIs is not supported.

## Example: Interaction between a parent task and a collaboration task

The following figure shows a book publishing process with subtasks for the human task activity.
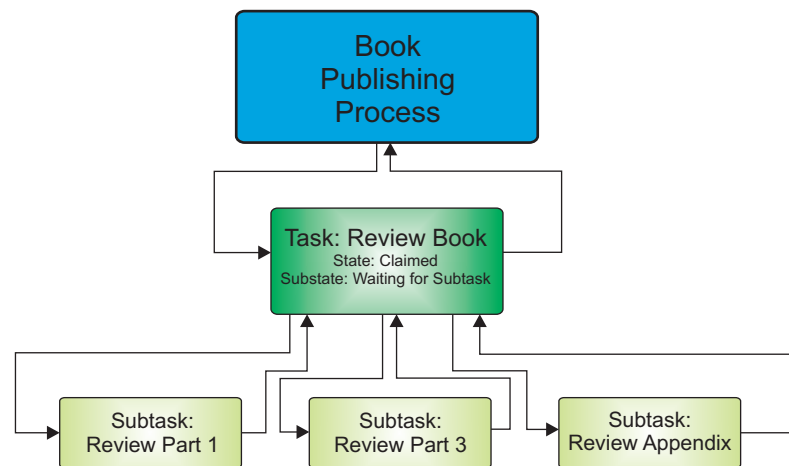


In a book publishing process, the "Review Book" task is claimed by Linda. She realizes that the book is too large for her to review alone, and specialized knowledge is required for some parts of it. She decides to deviate from the standard publishing process, and assigns parts of her task to some of her colleagues. She creates three additional tasks from the "Review book section" template: "Review Part 1", "Review Part 3", and "Review Appendix". She will review part 2 of the book herself.

She includes the complete book as input to the subtasks so that her colleagues have enough context information, but adds a note to the task description to tell her colleagues to review only the parts of the book that are assigned to them. She assigns the tasks to her colleagues: John to review part 1, Cindy part 3, and Mary the appendix. Then she starts the three tasks as subtasks of her own "Review Book" task. Her task that was in the claimed state is put into the waiting-for-subtask substate until all three subtasks are complete.

Cindy, John, and Mary claim their subtasks and start reviewing their parts of the book. In the meantime, Linda reviews part 2 of the book. When she finishes her part of the review, she checks on the progress of her colleagues. Cindy and John have completed their review, but Mary is still reviewing the large appendix. Linda's task is still in the waiting-for-subtask substate. Although, Linda cannot complete her task, she starts consolidating the review comments based on the output of Cindy and John's subtasks.

In the meantime, Mary completes her subtask too, and Linda's "Review Book" task leaves the waiting-for-subtask substate. Now, Linda consolidates Mary's review comments with the rest of the book, and completes her task. The book publishing process continues. Because the "Review Book" task is an inline human task, it is deleted with its subtasks when the business process instance is deleted.

### Example: Interaction between a parent task and an invocation task

The interaction between a parent task and an invocation task is similar to that of a parent task and a collaboration task. The task owner creates a task from an existing invocation task template, and starts it as a subtask of her own task. The parent task enters the waiting-for-subtask substate and waits for the invocation subtask to return. When all of the subtasks are complete, the parent task leaves the waiting-for-subtask substate and it can be completed.

**Related concepts**

"Authorization roles for human tasks" on page 64
Actions that you can take on human tasks depend on your authorization role. This role can be a system-level J2EE role or an instance-based role.

"Life cycle of human tasks" on page 51
Human tasks support people when they interact with Web services or business processes. The interactions that can take place over the lifetime of a task depend on whether the task is a to-do task, a collaboration task, a invocation task, or an administration task. Certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the human task.

## Follow-on tasks

Follow-on tasks support people when they want to delegate parts of their assigned work to other people, and the control over the completion of the work.

Follow-on tasks can be created from stand-alone task templates that are stored in the Business Process Choreographer database, from task templates created at runtime, or by providing a new task model at runtime. A follow-on task can have follow-on tasks of its own resulting in a chain of tasks.

Follow-on tasks can be only collaboration tasks. You can start a collaboration task from a to-do task or a collaboration task that has the **supportsFollowOnTask** attribute set to `true`.

The input message type of a follow-on task can be different from its predecessor task. If the input message type of the follow-on task is the same as that of the predecessor task, the input message content of the predecessor task is passed automatically to the follow-on task. The message content can be overwritten when the follow-on task is created or started.

For a chain of follow-on tasks, the output and fault message types of each of the follow-on tasks must be identical to those of the top-level task in the chain, because the last follow-on task in the chain returns the message to the calling component or person (originator). The output or fault message content of the parent task is always copied to the output or fault message of the follow-on task. These messages can be modified in the follow-on task.

## Authorization considerations

Follow-on tasks inherit the authorization roles from the predecessor task:
- The readers, editors, originator, and owner of the predecessor task become readers of the follow-on task and its escalations
- Administrators of the predecessor task become administrators of the follow-on task
- Escalation receivers of the predecessor task become readers of the follow-on task

## Life cycle considerations

When the follow-on task is started, the predecessor task enters the forwarded state. Follow-on tasks are children of their predecessor task so some life cycle operations (state changes) of the predecessor task are propagated to its follow-on tasks. When the predecessor task is suspended, resumed, escalated, terminated, deleted, or it expires, all of its follow-on tasks are also suspended, resumed, escalated, terminated, deleted, or expire.

Some life cycle operations on a follow-on task can conflict with the life cycle operations of the predecessor task, and are therefore not allowed. These are mainly operations that influence the end of the life cycle of a follow-on task and need coordination with the predecessor task. The following operations can be performed on follow-on tasks:
- Life cycle operations that do not conflict with the parent task are always supported. These are operations, such as claim, cancel claim, complete, creation and start of subtasks or follow-on tasks.
- Because the chain of follow-on tasks behaves like a single task to the calling component or person (originator), follow-on tasks do not support a duration until expiration, but expire when the expiration timer ends for the top-level task in the chain.
- Follow-on tasks can be suspended and resumed because work on a follow-on task might need to be stopped although work on the parent task continues.
- Follow-on tasks can be terminated.
- Follow-on tasks can have their own escalations so that the owner of the predecessor task and the originator of the follow-on task can better control the progress of the follow-on task.

- The deletion of individual follow-on tasks using the Business Process Choreographer APIs is not supported.

### Example: Follow-on tasks

The following figure shows a publishing process with a follow-on task for the human task activity.
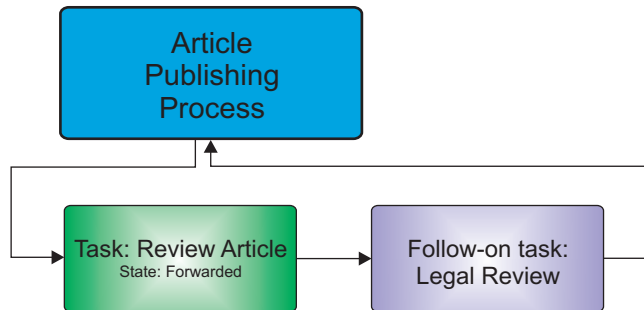


In an article publishing process, the "Review Article" task is claimed by John. He is empowered by the process to review and approve the legal aspects of articles as well. However, this article describes the collaboration with a competitor product, and is thus very sensitive from a legal point-of-view. He reviews the informational aspects of the article, and decides to pass the article on to Sarah from the legal department for additional review. He creates a "Legal Review" task, with a description that highlights his legal concerns. He includes the article as input to the task, and then assigns it to Sarah. He then starts the new task as a follow-on task of his own "Review Article" task. His task enters the forwarded state, and the work on it ends. The process waits for the response from the invoked "Review Article" task.

Sarah claims her "Legal Review" follow-on task and starts reviewing the legal aspects. She makes some comments, and completes her task. The output message of the follow-on task is passed to the business process. The article publishing process continues with the output that it associates with the "Review Article" task, but that actually comes from the "Legal Review" follow-on task. Because the "Review Article" task is an inline human task, it is deleted with the "Legal Review" task when the business process instance is deleted.

## Escalations

An escalation is an alert that is raised automatically when a human task is not actioned in the specified amount 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.

You can define escalations for any task by defining either an escalation template for the task template, or by defining them with an adhoc task at runtime.

Escalations are activated at a certain task state and escalate only if the expected task state (surveillance state) has not yet been reached when the time limit for the escalation expires. The time limit for the escalation timeout is specified as a string and it is interpreted by the calendar specified for the task. You can specify multiple escalations (or escalations chains) that have the same activation state.

You can define escalations that are activated when the task reaches the following task states:

**Ready**  For tasks in the ready state, you can define escalations for the following situations:

- Escalate when the task is not claimed in time using the expected task state of claimed.
- Escalate when the task is not completed in time using the expected task state of ended.

**Claimed**

For to-do tasks or collaboration tasks in the claimed state, you can define escalations for the following situations:

- Escalate when the task is not completed in time using the expected task state of ended.
- Escalate when the subtasks of this task are not completed in time using the expected task state of subtasks completed. Alternatively, you can use the waiting-for-subtasks activation state as the expected state so that you can track the progress of the subtask.

**Waiting for subtask**

In the waiting-for-subtasks substate of a to-do or collaboration task, you can escalate the task when its subtasks are not completed in time using the expected task state subtasks-completed.

**Running**

In the running state of an invocation task, you can escalate when the invoked service does not return in time using the expected task state of ended.

You can define repeating escalations. These escalations check the same expected task state at every timeout, and perform the defined escalation action until the expected task state is reached.

When an escalation is raised, the people affected by the escalation (the escalation receivers) receive work items. Depending on the definition of the escalation, the escalation receivers might also receive an e-mail notifying them that the task is escalated. The list of users to be notified is defined by a people query. This query must resolve to a set of user IDs for work item creation.

You can define the escalation to increase the priority of the escalated task using the increasePriority property. The priority can be increased automatically either for the first iteration only, or for every iteration of the escalation.

## Escalation life cycle

An escalation goes through the following states during its life cycle:

- After creation, an escalation remains inactive until the task reaches the activation state.
- When the task reaches the activation state for the escalation, the escalation is put into the waiting state. The timer is started and the escalation waits until it times out.
- When a timeout occurs, the atLeastExpectedState property of the task under surveillance is checked. If the task has reached or passed this state, the

escalation state is put into the superfluous state. If the expected state is not yet reached, the escalation is put into the escalated state, and the modeled escalation action is invoked.

After an escalation is created, it cannot be modified. The escalation action can be executed repeatedly. The repetition interval is defined by the autoRepeatDuration property of the escalation.

## Chained escalations

A chain of escalations is activated when the task reaches the start state for the first escalation in the chain. All of the escalations in a chain must have the same activation state. In a chain only one escalation is active at a time, except for repeated escalations because they remain active. Escalations that are defined as a sequence are processed sequentially: when the first escalation is raised, the next escalation in the chain is activated, and so on.

The wait duration of a chained escalation is calculated relative to the timeout of the previous escalation, and not to the time when the task reached the escalation activation state. Thus, if the wait duration of the first escalation in a chain is two hours and that of the second escalation in the chain is three hours, the first timeout occurs two hours after the task reaches the activation state and the second timeout occurs three hours later, so, five hours after the task reached the activation state. This behavior ensures that a later escalation in the chain does not time out before its predecessors.

## Dynamic durations for escalations

For some escalations, you might want to set the escalation period dynamically at runtime. You can do this by specifying a replacement expression instead of the fixed value when you define the escalation. The duration variable must be enclosed in percentage signs (%).

The variable can be any of the following:
- A task variable, such as *%htm:input.myEscalationDurationValue%*
- A custom property, such as *%htm:task.property.myEscalationDurationValue%*
- A process variable, such as *%wf:variable.myVariable\myPart\myEscalationDurationValue%*

You must make sure that the context data that you access is available when the escalation is evaluated.

The following table shows when escalation durations are evaluated:

| Duration for | Is evaluated when | Context date must be set before the task reaches the following state: |
|---|---|---|
| Escalation | The task reaches the activation state of the escalation | The task activation state of the escalation |
| Escalation repetition | The escalation is raised | Escalated |

**Related concepts**
People directories store user information that is used for people resolution.

**Related tasks**

"Creating notification event handlers" on page 541
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.

## E-mail notifications for escalations

When an escalation is raised, the people affected by the escalation receive a work item. They can also be sent an e-mail that the task has been escalated. Certain rules apply to the e-mail.

Each escalation can have a different e-mail. You can personalize the standard escalation e-mail, for example, so that it conforms to your organization's standards. To personalize the e-mail, edit the escalation details in the human task editor.

Both the e-mail subject line and body text can contain replacement expressions to make the e-mail more relevant to the task that it refers to, for example, to include the task name. These expressions must be set before the e-mail is sent otherwise the recipients of the e-mail see *%variable name%* in their e-mails. You can use any of the task and escalation expressions.

The following HTML snippet shows a sample e-mail that contains replacement expressions:

```
<html>
<head>
</head>
<body lang="EN-US"><div>
<p>The task '<b><span style="font-size:14.0pt">%htm:task.displayName%</span></b>'
with id '<b><span style="font-size:14.0pt">%htm:task.instanceID%</span></b>
' has been escalated because the </p>
<p>expected state '<b><span style="font-size:14.0pt">%htm:escalation.expectedTaskState%</span>
</b>'
 has not been reached in time.
</p>
<p> </p>
<p>The task has the following description: </p>
<p><span style="font-size:14.0pt;color:red">%htm:task.description%</span></p>
<p> </p>
<p><span style="font-size:14.0pt;color:green">The name of the Escalation is: %htm:escalation.displayName%
 and the escalation description reads: %htm:escalation.description%</span></p>
<p> </p>
<p><a href="%htm:task.URLPrefix%?id=%htm:task.instanceID%">Task details</a></p>
<p><a href="%htm:escalation.URLPrefix%?id=%htm:escalation.instanceID%">Escalation details</a></p>
</div>
</body>
</html>
```

## Life cycle of human tasks

Human tasks support people when they interact with Web services or business processes. The interactions that can take place over the lifetime of a task depend on whether the task is a to-do task, a collaboration task, a invocation task, or an administration task. Certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the human task.

## To-do and collaboration tasks

To-do tasks and collaboration tasks support people when they perform work as part of a business process (inline tasks), or implement a Web service that is publicly available (stand-alone task). To-do tasks and collaboration tasks differ in how they are started. To-do tasks are created automatically by a client application, or calling component. Collaboration tasks are created and started by a person.

The following diagram shows the state transitions that can occur during the life cycle of to-do tasks and collaboration tasks:
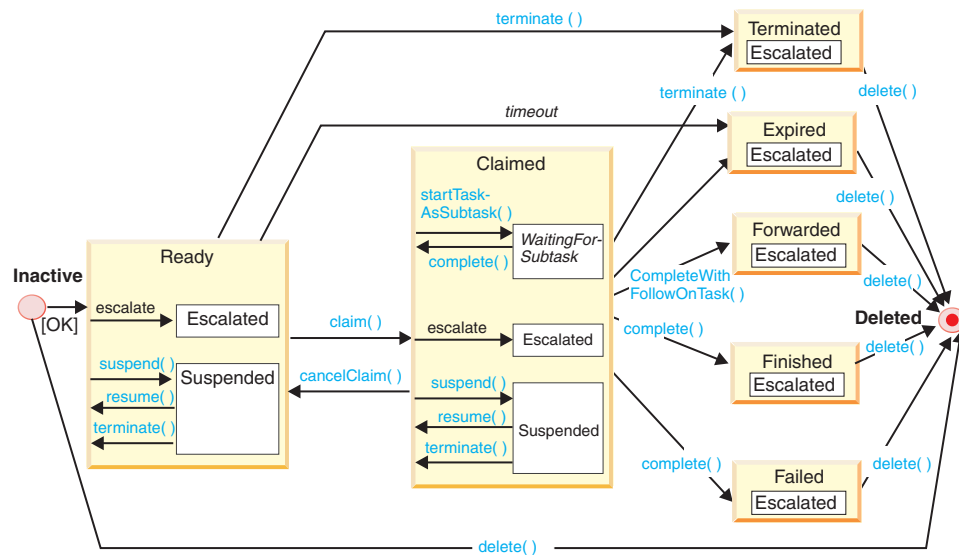


After the task is created, it is put into the inactive state. In this state, you can update task properties or set custom properties, but the task cannot be claimed. To work on a to-do task or collaboration task, it has to be started.

After the task is started, it is put into the ready state. In this state the task waits for one of the potential owners to claim it and perform the work associated with this task. In this state, the following exceptional events can occur:

- The task can be escalated because it is not claimed in time. It is put into the escalated substate, and it stays in this substate for the rest of the task life cycle.
- The task can be suspended manually. It is put into the suspended substate. Most actions on the task are blocked in this state. It can be resumed manually, or automatically by a timer that is set with the suspend action.
- The task can expire. This state change ends the task.
- The task can be terminated manually using the terminate action. This state change ends the task.

In the normal task flow, one of the potential owners claims the task, and becomes the owner. The task is put into the claimed state and the owner and the editors can work on it. When tasks are in the claimed state, task owners can take the following actions:

- If they need support for their work, they can delegate pieces of work using subtasks. These subtasks can be either collaboration tasks or invocation tasks. The parent task then enters the waiting-for-subtask substate and remains in this

state until all of its subtasks reach an end state. The parent task can be suspended while waiting for subtasks, but it cannot be completed and the claim cannot be canceled.

- If they want to delegate completion of the work to someone else, they can create a collaboration task and complete the task with the follow-on task. The parent task is put into the forwarded end state.
- If they want to delegate the overall responsibility for tasks, they can transfer owner work items to another potential owner, or an administrator.
- If they want to give up ownership of a task, they can cancel the claim of the task. The task is put into the ready state again, and it can be claimed by one of the potential owners.

In the claimed state, the following exceptional events can occur:

- The task can be escalated because it is not completed in time, or if it waits too long for subtasks. It is put into the escalated substate, and it stays in this state for the rest of the task life cycle.
- The task can be suspended manually. It is put into the suspended substate. Most actions on the task are blocked in this state. It can be resumed manually, or automatically by a timer that is set with the suspend action. Alternatively, when the timer expires, the claim on the task is cancelled and it is put into the ready state again.
- The task can expire. This is a state change that ends the task.
- The task can be terminated manually using the terminate action. This is a state change that ends the task.

When the owner finishes work on the task, they complete it. The task is then put into the finished state if it completes successfully, or the failed state if an error occurs.

The failed, terminated, finished, and expired states are end states in which work cannot be performed. If the task template specifies automatic deletion, the task is either deleted immediately, or after the deletion timer expires. Without automatic deletion, the task remains in its end state until it is explicitly deleted. When the parent task is deleted, its subtasks are also deleted. Automatic deletion does not apply if the task is in the forwarded end state. In this case, the parent task is deleted with its follow-on task. The deletion timer starts when the follow-on task reaches an end state.

Some additional rules apply to inline to-do tasks. Inline tasks are an integral part of the business process and thus their life cycle is controlled by the process life cycle:

- The task is created and started implicitly by the business process.
- The task is represented in the business process by a human task activity. Both the task and the activity have the same state, for example, when that task is in the ready state, the human task activity is in the ready state too. The human task activity does not reflect the forwarded state or the task substates.
- If the inline task has subtasks, the human task activity is not aware of them, and it waits in the claimed state until the parent task completes.
- If the inline task has follow-on tasks, the human task activity is not aware of them, and it waits in the claimed state until the follow-on task completes.
- Inline to-do tasks have no duration until expiration and cannot be terminated manually. Both expiration and termination are controlled by the human task activity or the business process.

- The tasks are deleted with the business process. They cannot be deleted manually, or have a duration until deletion.

## Invocation (originating) tasks

Invocation tasks support people when they invoke Web services. The person who creates and starts the invocation task becomes the task originator. When the task is started, it automatically invokes the service and waits for its result. When the service result is available, the invocation task stores it and the originator can retrieve it as long as the task exists.

The following diagram shows the state transitions that can occur during the life cycle of invocation tasks:
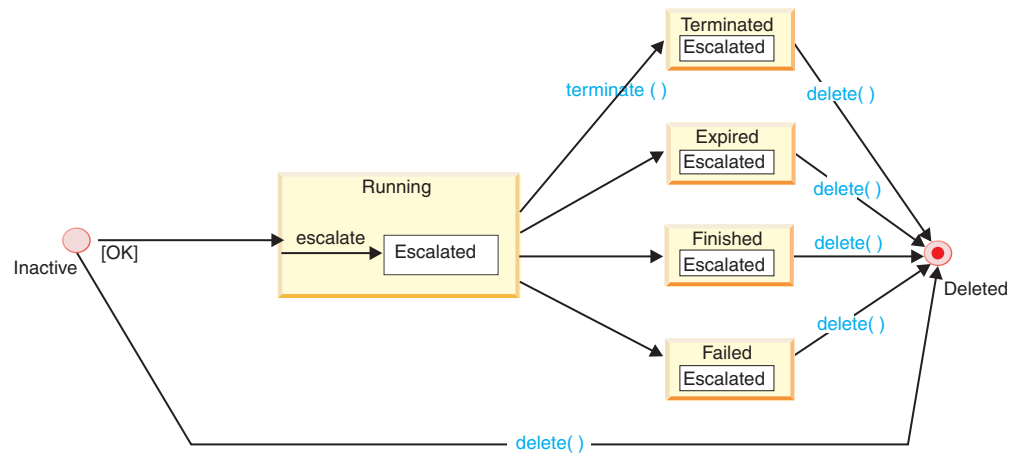


After creation, the task reaches the inactive state. In this state, you can update task properties, or set custom properties. To invoke the service, the task must be started. It can be started by the originator or one of the potential starters.

After the task starts, it is put into the running state. In this state the task waits for the invoked service to return. The following exceptional events can occur in this state:

- The task can escalate if the service does not return in time. It is put into the escalated substate, and it stays in this state for the rest of the task life cycle.
- The task can expire. This is a state change that ends the task.
- The task can be terminated manually using the terminate action. This is a state change that ends the task.

The normal task flow is that the service returns with an output or fault message. The task is then put into the finished state if an output message is returned, or the failed state if a fault message is returned. In both cases, the message is available to the task originator and starter.

The failed, terminated, finished, and expired states are end states. If the task template specifies automatic deletion, the task is either deleted after the deletion timer expires, or it is deleted manually. By default, invocation tasks are not automatically deleted so that the result of the invoked service can be accessed.

Some additional rules apply to inline invocation tasks. These tasks are an integral part of the business process, and thus the process can control their life cycle:

- If the business process is started using the Business Flow Manager API, or an SCA client, the task for the activity that creates the process instance is created and started implicitly by the business process. Invocation tasks can also be used by process instances that are already running. In this case they are created by the process and are associated with a receive or pick (receive choice) activity, or an on-event event handler.
- The task is represented in the business process as a receive or pick (receive choice) activity, or an on-event event handler. If an inline invocation task is defined for an activity, it also defines the authorization for this activity.
- Inline invocation tasks have no duration until expiration and cannot be terminated manually.
- If the task is started implicitly by the business process, it is also deleted implicitly with the business process.
- If the task is started by the Human Task Manager API, then it is not deleted with the process. If the task is modeled with automated deletion, it is deleted after the deletion timer expires. It can also be deleted manually.

## Administration tasks

Administration tasks support people in administering business processes and their activities. If an administration task template is not available, a default administration task is created at runtime whenever one is needed by the business process.

The following diagram shows the state transitions that can occur for administration tasks:
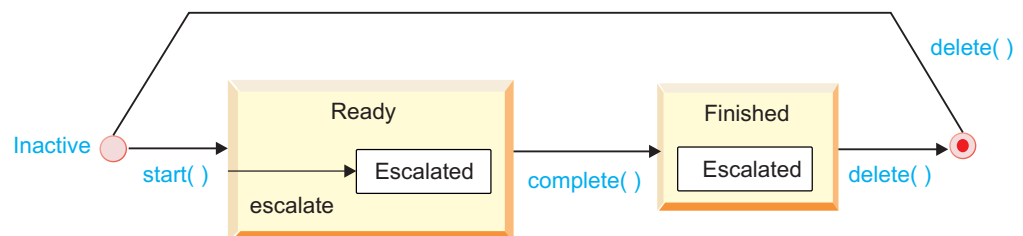


Business Flow Manager creates and starts an administration task implicitly in a single transaction. The inactive state is therefore not visible externally, and the task directly reaches the ready state.

The finished state is an end state. It does not, however, prohibit further administrative actions.

Administration tasks are always inline tasks and thus their life cycle is controlled by the business process. They are always deleted with the business process.

**Related concepts**

"Subtasks" on page 44
Subtasks support people when they need to delegate parts of their assigned work to other people, but want to keep control over the overall result. They can also be used to invoke supporting services to help people accomplish the tasks that they are working on.

"Stand-alone and inline tasks" on page 41
The Service Oriented Architecture (SOA) patterns recommend the realization of software solutions with a set of loosely coupled components. The human tasks

that follow the SOA patterns are called *stand-alone tasks*, while the human tasks that are defined as part of a business process are called *inline tasks*.

# Scenarios for invoking tasks

The various ways in which tasks can be invoked is described here.

## Invoking task components using the Human Task Manager API

Tasks can be instantiated by using the Human Task Manager API. Human Task Manager API clients use the API to create and start task instances, and query and manipulate task instances. For task invocation, the API provides methods to create and start the following kinds of tasks:

- Stand-alone and inline invocation tasks
- Stand-alone to-do tasks
- Collaboration tasks

Administration tasks cannot be invoked using the API because they are invoked in the context of a business process.

The API supports the following interaction styles for tasks:

- Synchronous invocation of the task and the associated service

  This interaction style uses the callTask method. For one-way operations, the invocation returns after triggering the execution of the task and the service component. For request-response operations, the invocation waits until the service and task are complete and the result of the invocation is returned.

  This style of interaction can be applied to invocation tasks only.

- Asynchronous invocation of the task and the associated service

  This interaction style uses the startTask method. For both one-way and request-response operations, the invocation returns after triggering the execution of the task and the service component. In addition, for request-response operations, the invocation returns a result asynchronously that is stored as an output or fault message in the context of the invocation task. The invoking API client must retrieve the result programmatically using the API methods. Alternatively, you can use a reply handler to ensure that the asynchronous response is returned to the client as soon as the response becomes available.

  This style of interaction can be applied to to-do, collaboration, and invocation tasks.

The Human Task Manager API is provided as an Enterprise JavaBeans (EJB) implementation and a Web service implementation. The API methods are similar for all implementations, but differ in their functional scope.

For more information on these API methods, see the Javadoc for the HumanTaskManager interface in the com.ibm.task.api package.

## Invoking to-do tasks as SCA service components

A stand-alone to-do task represents an SCA service component that can be invoked by an SCA client asynchronously. The mechanisms provided by SCA are available for connecting SCA clients and stand-alone to-do tasks. This includes the SCA means to define the following:

- Wires connecting an SCA client (reference) and the interface of a component representing a to-do task

- SCA qualifier settings for component references and interfaces that control aspects, such as interaction style, transaction behavior, and interaction reliability.

In addition, a stand-alone to-do task can be invoked by an SCA client that is implemented as a business process. In this case, the connection must be considered on both SCA and process levels. Viewed on the SCA level, the SCA client reference is connecting to the interface of an SCA service. Viewed on the process level, the partner link of an invoke activity is connected to a to-do task.

### Invoking inline to-do tasks

A to-do task can be specified in the context of a human task activity in a long-running business process. In this case, the task does not have a representation on the SCA level, instead it is part of the SCA component representing the business process. The task acts as a service provider to the human task activity. Whenever the activity is reached during process navigation, the to-do task is invoked asynchronously.

### Invoking an SCA service through an invocation task

A stand-alone invocation task serves as an access component to an associated SCA service. The association with the service is defined on the SCA level: the task represents an SCA client that is wired to an SCA service component. The invocation of an invocation task involves both Human Task Manager and SCA levels. The invocation task itself is invoked by the Human Task Manager API, either synchronously or asynchronously. The task (SCA client) then invokes the associated SCA service component either synchronously if the task was invoked synchronously, or asynchronously if the task was invoked asynchronously.

The modeling of the association between the task and the service is done on the SCA level. The concepts and mechanisms provided by SCA are therefore available for connecting stand-alone invocation tasks and SCA service components. This includes the SCA means to define the following:
- Wires that connect an SCA client reference and the interface of a service component
- SCA qualifier settings for component references and interfaces that control aspects, such as interaction style, transaction behavior, and interaction reliability

In addition, a stand-alone invocation task can be connected to an SCA component that is implemented by a business process.

### Invoking a business process through an inline invocation task

An inline invocation task can be specified in the context of a receive or pick activity, or an event handler in a business process. The task does not get a representation on the SCA level, instead it is part of the SCA component that represents the business process. Nonetheless, the task acts as a client to the business process. Whenever the task is invoked by the Human Task Manager API, the task in turn invokes the business process in the same way as it was invoked.

### Administration tasks created for administering business processes and activities

Administration tasks support people in administering business processes and their activities. Business Flow Manager creates and starts an administration task for

every business process or activity type that allows administration. If an administration task template was specified for the process or activity, this template is used. If a template is not available, a default administration task is created, whenever one is needed by the business process.

**Related concepts**

"Factors affecting the behavior of stand-alone invocation tasks and their service components"
You can use a stand-alone invocation task to run an Service Component Architecture (SCA) service component that is associated with the SCA component of the task. The association of the invocation task and service component is modeled at an SCA level by wiring the reference of the task component to the interface of the associated service component. A number of factors affect the behavior of the invocation task and its associated service component.

This scenario considers the asynchronous invocations of tasks and services only. It describes the Service Component Architecture (SCA) settings and the expected transactional and fault behavior for this type of invocation.

This scenario considers both the asynchronous and synchronous invocation of a task and its associated service. It describes the Service Component Architecture (SCA) settings and the expected transactional and fault behavior for these types of invocation.

**Related tasks**

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.

# Factors affecting the behavior of stand-alone invocation tasks and their service components

You can use a stand-alone invocation task to run an Service Component Architecture (SCA) service component that is associated with the SCA component of the task. The association of the invocation task and service component is modeled at an SCA level by wiring the reference of the task component to the interface of the associated service component. A number of factors affect the behavior of the invocation task and its associated service component.

**WSDL operation type**

SCA references and SCA interfaces are associated with a WSDL port type containing one or more operations. Each operation can be a one-way or a request-response operation:

- A one-way operation implies a service execution the completion of which is not made known to the invoking task. The task service execution ends with the successful invocation of the associated service.

- A request-response operation implies a service execution the completion of which is made known to the invoking task. The task execution ends when the result of the service execution is made available to the invoking task.

**API invocation method**

The Human Task Manager API supports the following interaction styles for tasks:

- Synchronous invocation of the task and its associated service using the callTask method
- Asynchronous invocation of the task and its associated service using the startTask method

**Execution duration of the service component**

The value that you set for the execution duration must account for the overhead that you expect due to other workload on your system. The execution duration also has to be considered in relation to the transaction time-out value set for the server that hosts Business Process Choreographer. Compare these values before you decide to make a service component with a request-response interface available for synchronous invocation. In such cases, the execution time of your service component must be below the transaction time-out value that is set for the server.

**SCA qualifier settings**

Only certain combinations of SCA qualifiers are allowed for the task component reference and service component interface.

**Related concepts**

"Scenarios for invoking tasks" on page 56
The various ways in which tasks can be invoked is described here.

# Scenario: stand-alone invocation tasks that support the asynchronous invocations of services

This scenario considers the asynchronous invocations of tasks and services only. It describes the Service Component Architecture (SCA) settings and the expected transactional and fault behavior for this type of invocation.

This scenario is applicable to Human Task Manager API clients, for example, Business Process Choreographer Explorer, that make use only of asynchronous invocations. It avoids the need for assessing the execution duration of the service associated with the task when you model the task.

## Task component settings

The task component can take the following settings. If you use WebSphere Integration Developer to define the task component, valid values for the attribute type are generated automatically.

| Qualifier type: attribute type | Value |
|---|---|
| Reference attribute: Multiplicity | 1:1 (mandatory) |
| Reference qualifier: DeliverAsyncAt | commit (mandatory) |
| Implementation qualifier*: Transaction | global (mandatory) |
| Reference qualifier**: SuspendTransaction | Not applicable |
| Implementation qualifier***: ActivitySession | true (mandatory) |
| Reference qualifier***: SuspendActivitySession | false (default) |
| Reference qualifier: Reliability | assured (mandatory) |
| Reference qualifier: RequestExpiration | any |
| Reference qualifier: ResponseExpiration | any |

| Qualifier type: attribute type | Value |
|---|---|
| **Note:** | |
| • *: use `global` if you use transactions settings, and `local` if you use activity session settings. <br> • **: if the transaction is set to `global`, only the transaction settings are used <br> • ***: if the transaction is set to `local`, only the settings for the activity sessions are used | |

## Service component settings

The service component can take the following settings. If you use WebSphere Integration Developer to define the task component, valid values for the attribute type are generated automatically.

| Qualifier type: attribute type | Value |
|---|---|
| Interface attribute: PreferredInteractionStyle | Ignored |
| Implementation qualifier*: Transaction | `local` (default) <br> `global` |
| Interface qualifier**: JoinTransaction | `false` (default) <br> `true` |
| Implementation qualifier***: ActivitySession | `any` (default) |
| Interface qualifier***: JoinActivitySession | `false` (default) |
| **Note:** | |
| • *: use `global` if you use transactions settings, and `local` if you use activity session settings. <br> • **: if the transaction is set to `global`, only the transaction settings are used <br> • ***: if the transaction is set to `local`, only the settings for the activity sessions are used | |

The following list gives the valid combinations of settings for the service **Transaction** and **JoinTransaction** qualifiers:

* The **Transaction** qualifier is set to `local` and the **JoinTransaction** is set to `false`. With these settings, the task and service invocation run in separate transactions.
* The **Transaction** qualifier is set to `global` and the **JoinTransaction** is set to `false`. With these settings, the task and service invocation run in separate transactions.
* The **Transaction** qualifier is set to `global` and the **JoinTransaction** is set to `true`. With these settings, the task and service invocation run in the same transaction.

## Transactional and fault behavior

In this asynchronous invocation scenario, the startTask method is used for API invocation only. Task and service invocations occur in different transactions. The following applies when a runtime exception occurs, which is not handled by the service implementation. This scenario has the following transactional behavior and exception handling.

| Operation type | When the SCA runtime exception occurs | Behavior of tasks and services |
|---|---|---|
| One-way operation | During service invocation but before the start of the service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| One-way operation | During service execution | The invocation task is not notified. The task moves to the finished state. A failed event is generated that can be handled by using the failed event manager. |
| Request-response operation | During service invocation but before the start of the service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| Request-response operation | During service execution | The task is notified of the SCA runtime exception and stores it in the task context in the database. If a reply handler is available, it is used to notify the client. The task is put into the failed state. |

The operation definition can include one or more fault messages that can be thrown by the service component during execution.

The task component is notified about a fault message as follows:
- The fault message is stored in the database in the context of the task
- The task is put into the failed state
- If the task was invoked synchronously and a reply handler was specified, the reply handler is invoked to return the fault occurrence to the client
- If the task was invoked asynchronously, the fault message is returned to the client as a FaultReplyException exception

Fault handling does not impact transactional behavior. The transactions are not rolled back.

**Related concepts**

"Scenarios for invoking tasks" on page 56
The various ways in which tasks can be invoked is described here.

## Scenario: stand-alone invocation tasks that support asynchronous and synchronous invocations of services

This scenario considers both the asynchronous and synchronous invocation of a task and its associated service. It describes the Service Component Architecture (SCA) settings and the expected transactional and fault behavior for these types of invocation.

In this scenario, Human Task Manager clients make use of both asynchronous and synchronous invocations. It implies that you have assessed whether the service execution time is lower than the expected value of the server transaction timeout. Typically, execution durations must be well below the value of the server transaction timeout.

## Task component settings

The task component can take the following settings. If you use WebSphere Integration Developer to define the task component, valid values for the attribute type are generated automatically.

| Qualifier type: attribute type | Value |
|---|---|
| Reference attribute: Multiplicity | 1:1 (mandatory) |
| Reference qualifier: DeliverAsyncAt | commit (mandatory) |
| Implementation qualifier*: Transaction | global (mandatory) |
| Reference qualifier**: SuspendTransaction | Not applicable |
| Implementation qualifier***: ActivitySession | true (mandatory) |
| Reference qualifier***: SuspendActivitySession | false (default) |
| Reference qualifier: Reliability | assured (mandatory) |
| Reference qualifier: RequestExpiration | any |
| Reference qualifier: ResponseExpiration | any |
| **Note:**<br>• \*: use global if you use transactions settings, and local if you use activity session settings.<br>• \*\*: if the transaction is set to global, only the transaction settings are used<br>• \*\*\*: if the transaction is set to local, only the settings for the activity sessions are used | |

## Service component settings

The service component can take the following settings. If you use WebSphere Integration Developer to define the task component, valid values for the attribute type are generated automatically.

| Qualifier type: attribute type | Value |
|---|---|
| Interface attribute: PreferredInteractionStyle | Ignored |
| Implementation qualifier*: Transaction | local (default)<br>global |
| Interface qualifier**: JoinTransaction | false (default)<br>true |
| Implementation qualifier***: ActivitySession | any (default) |
| Interface qualifier***: JoinActivitySession | false (default) |
| **Note:**<br>• \*: use global if you use transactions settings, and local if you use activity session settings.<br>• \*\*: if the transaction is set to global, only the transaction settings are used<br>• \*\*\*: if the transaction is set to local, only the settings for the activity sessions are used | |

The following list gives the valid combinations of settings for the service **Transaction** and **JoinTransaction** qualifiers:

- The **Transaction** qualifier is set to local and the **JoinTransaction** is set to false. With these settings, the task and service invocation run in separate transactions.
- The **Transaction** qualifier is set to global and the **JoinTransaction** is set to false. With these settings, the task and service invocation run in separate transactions.

- The **Transaction** qualifier is set to `global` and the **JoinTransaction** is set to `true`. With these settings, the task and service invocation run in the same transaction.

### Transactional and fault behavior

This scenario has the following transactional behavior and exception handling.

| API invocation style | Operation type | When the SCA runtime exception occurs | Behavior of tasks and services |
|---|---|---|---|
| callTask | One-way operation | During service invocation but before the start of the service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| callTask | One-way operation | During service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| callTask | Request-response operation | During service invocation but before the start of the service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| callTask | Request-response operation | During service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| startTask | One-way operation | During service invocation but before the start of the service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| startTask | One-way operation | During service execution | The invocation task is not notified. The task moves to the finished state. A failed event is generated that can be handled by using the failed event manager. |
| startTask | Request-response operation | During service invocation but before the start of the service execution | The task receives an SCA runtime exception. The Human Task Manager API method throws a CoreOTaskServiceRuntimeExceptionReceivedException exception. The task transaction is rolled back and the task stays in the inactive state. |
| startTask | Request-response operation | During service execution | The task is notified of the SCA runtime exception and stores it in the task context in the database. If a reply handler is available, it is used to notify the client. The task moves to the failed state. |

The operation definition can include one or more fault messages which can be thrown by the service component during execution.

The task component is notified about a fault message as follows:
- The fault message is stored in the database in the context of the task
- The task is put into the failed state

- If the task was invoked asynchronously and a reply handler was specified, the reply handler is invoked to return the fault occurrence to the client
- If the task was invoked synchronously, the fault message is returned to the client as a FaultReplyException exception

Fault handling does not impact transactional behavior. The transactions are not rolled back.

**Related concepts**

"Scenarios for invoking tasks" on page 56
The various ways in which tasks can be invoked is described here.

# Authorization and people assignment

Authorization is the mechanism by which certain people are enabled to perform selected actions on task templates, task instances, and escalations. Authorization roles are used to define sets of actions available to specific roles. People can be assigned to system-level roles using J2EE mechanisms, or to task instance roles using people assignment criteria.

## Authorization roles for human tasks

Actions that you can take on human tasks depend on your authorization role. This role can be a system-level J2EE role or an instance-based role.

System-level Java 2 Platform, Enterprise Edition (J2EE) roles are set up when Human Task Manager is configured. The authority level implied by these roles is valid for all tasks and escalations. Instance-based roles are valid for individual task and escalation instances, or the templates that are used to create task or escalation instances. Role-based authorization requires that administration and application security is enabled for the application server.

### J2EE roles

The following J2EE roles are supported:
- TaskSystemAdministrator. Users assigned to this role have all privileges. This role is also referred to as the system administrator for human tasks.
- 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.

### Instance-based roles

A task instance or an escalation instance is not assigned directly to a person, instead it is associated with predefined roles to which people are assigned. Anyone 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 either by people assignment, or as the result of task actions.

People are assigned to the following roles at runtime by people assignment based on the user and user group information that is stored in a people directory: potential creator, potential starter, potential owner, reader, editor, administrator,

and escalation receiver. The following roles are associated with only one user and are assigned as the result of a task action: originator, starter, owner.

These roles are authorized to perform the following actions:

| Role | Authorized actions |
|------|--------------------|
| Potential creator | Members of this role can create an instance of the task. If a potential instance creator is not defined for the task template, then all users are considered to be a member of this role. |
| Originator | The person with this role has 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 | The person with this role has 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 a potential owner is not defined for the task template, 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 | The person with this role works on and completes a task. |
| Reader | Members of this role can 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. |

**Related concepts**

"Authorization roles for business processes" on page 32
A role is a set of people who share the same level of authorization. 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.

"Subtasks" on page 44
Subtasks support people when they need to delegate parts of their assigned work to other people, but want to keep control over the overall result. They can also be used to invoke supporting services to help people accomplish the tasks that they are working on.

"Default people assignments" on page 84
Default people assignments are performed if you do not define people assignment criteria for certain task roles, or if people resolution fails or does not return a result. The default assignments differ for inline tasks and stand-alone tasks.

### Instance-based authorization roles for task kinds

Instance-based authorization roles are associated with human tasks and escalations when the task is modeled. The task kind determines whether a specific authorization role is available.

| Role | To-do tasks | Invocation tasks | Collaboration tasks | Administration tasks | Comments |
|---|---|---|---|---|---|
| Potential instance creator | X | X | X | | People who are allowed to create task instances |
| Originator | X | X | X | | The person who created the task |
| Potential owner | X | | X | | People who can claim and work with tasks |
| Owner | X | | X | | The person who claimed the task |
| Potential starter | | X | | | People who are allowed to start the task |
| Starter | | X | | | The person who started the task |
| Administrator | X | X | X | $X^1$ | People who are allowed to administer a task |
| Editor | X | | X | | People who are allowed to edit task data |
| Reader | X | X | X | $X^2$ | People who are allowed to see task data |
| Escalation receiver | $X^3$ | $X^3$ | $X^3$ | $X^3$ | People who receive an escalation |

**Notes:**

1. This role also has authorization for administrative actions on the administered process or activity
2. This role also has authorization for read operations on the administered process or activity
3. This role is authorized to perform actions on the escalations generated from these task types but not on the tasks themselves

## Authorization and work items

Every task role enables users to carry out an exact set of actions on the associated task. A person's authorization is managed using work items. A work item represents the relationship of the assigned person to the task actions implied by the task role.

A work item has the following aspects:

- The identity of a user or user group
- The identity of the task upon which actions can be performed
- The task role that the users are associated with

The people associated with a work item can be specified in one of the following ways:

- As exactly one user ID. This leads to a user work item.
- As exactly one user group ID. This leads to a group work item.
- For every user by using the **Everybody** people assignment criteria. This leads to an Everybody work item.

The authorization mechanisms of Business Process Choreographer ensure that a user can perform the actions associated with a work item if one of the following conditions holds:

- The user logs in with a user ID that matches the specified user ID for the user work item
- The logged-on user is a member of the group that corresponds to the specified group ID for the group work item
- The work item is a work item that is assigned to everybody

The Human Task Manager API provides methods for querying human tasks, escalations, and other objects. When a query is run, a user's authorization to see the queried data is ensured by returning only the data for which the user has a work item. You can also use the API to manage instance-based authorization. This is done by creating and deleting work items, and by transferring work items between people. For more information on these API methods, see the Javadoc for the HumanTaskManager interface in the com.ibm.task.api package.

## People assignment criteria

People assignment criteria are constructs that are used in the task model to identify sets of people that can be assigned to an instance-based authorization role. At runtime, the people resolution uses the people assignment criteria to retrieve the user IDs and other user information, for example, for composing e-mails. People assignment criteria are also used during runtime when task models or are created programmatically.

You can use people assignment criteria definitions (previously known as staff verbs) in WebSphere Integration Developer to model people assignments for task roles. A definition comprises a query name and a set of query parameters. When the task is deployed, the assignment criteria are transformed into queries that are specific to the people directory, for example, virtual member manager. When the task runs, these queries retrieve the set of people who are assigned to a role, such as potential owner.

The following example illustrates the steps that are involved in implementing a people assignment criteria definition for a task role:

1. In WebSphere Integration Developer, a modeler associates a new task with the people directory configuration, for example, for virtual member manager, bpe/staff/samplevmmconfiguration.

   This step determines which people assignment criteria are available for people assignment.

2. In WebSphere Integration Developer, the modeler associates a task role with a people assignment criteria definition.

   For example, the potential owner role is associated with the people assignment criteria **Group Members**, including the parameters:

   - **GroupName** set to the value `cn=group1, dc=mycomp, dc=com`
   - **IncludeSubgroups** set to the value `true`

3. When the task is deployed, the people assignment service establishes which people directory provider to use. It transforms the people assignment criteria into a query for the people directory provider, which is stored internally.

Depending on the people directory that is used, different subsets of the predefined people assignment criteria are available when the task is modeled:

- The LDAP and virtual member manager people directory providers support all of the predefined definitions
- The user registry people directory provider supports only those definitions that are based on user and group names. Support is not provided for definitions based on manager, or e-mail attributes.
- The system people directory provider is for testing purposes only. Support is limited to specifying a set of hard-coded user IDs so that access to a people directory is not required.

## Predefined people assignment criteria

Predefined people assignment criteria are provided for retrieving sets of users from people directories.

You can use people assignment criteria (previously known as staff verbs) in WebSphere Integration Developer to specify people assignments in a human task. These criteria are transformed during modeling and deployment into a set of queries that can be run on a people directory. The parameters for the following predefined people assignment criteria are listed here:
- Department Members
- Everybody
- Group
- 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
- User Records by user ID
- User Records by user ID without Named Users
- Users
- Users by user ID
- Users by user ID without Named Users

Consider the following when you assign people assignment criteria:
- If you are working with large groups of people, the Group people assignment criteria works best because it handles the members of a group as a unit. This allows you to transfer a human task from one group to another group easily. A person's group membership is resolved when the person logs in and accesses a human task.
- To individually assign people that belong to a group to a human task, the Group Members people assignment criteria provides an alternative to the group assignment. The Group Members people assignment criteria creates an assignment for each person individually. This assignment can then be transferred to another person. Substitution can occur, that is, a person that is absent can be replaced by another person. A variant of this people assignment criteria, Group Members without Named Users supports the separation-of-duties assignment pattern.

    **Note:** Assigning people to a group individually can affect performance at runtime, especially when assigning more than a few people to the group.

- To assign a few people to a human task that do not all belong to the same group, consider using the User Records by user ID people assignment criteria definition. You can also use this definition when the people assignment is not statically defined during modeling, but includes replacement expressions. Replacement expressions can refer to custom properties or the input message of a human task. The Users by user ID people assignment criteria definition is similar to the User Records by user ID definition. Although the Users by user ID definition performs better than the User Records by user ID definition at runtime, it provides less functionality:
  - It does not check if the user IDs are entered correctly
  - It does not retrieve, for example, the e-mail addresses for the user IDs specified, which makes it less suitable for assigning people to e-mail escalations
- The Everybody people assignment criteria definition is also worth considering. It indicates that all authenticated users are assigned to the human task. While there are cases where all people in an organization can do a certain job, this definition is particularly useful during development, or when rapidly prototyping an application.

### Department Members

Use this criteria to retrieve the members of a department. It is supported by the Lightweight Directory Access Protocol (LDAP), and the virtual member manager people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| DepartmentName | Mandatory | string | Department name of the users to retrieve. The department name must correspond to one of the following values:<br><br>• For virtual member manager, a unique name of a virtual member manager group<br><br>• For LDAP, a distinguished name (DN) of an LDAP group |
| IncludeNestedDepartments | Mandatory | boolean | Specifies whether nested departments are considered in the query. |
| AlternativeDepartmentName1 | Optional | string | An additional department to which the users can belong. |
| AlternativeDepartmentName2 | Optional | string | An additional department to which the users can belong. |

### Everybody

Use this criteria to assign every user that is authenticated by WebSphere Process Server to a task role. This criteria has no parameters.

This criteria is supported by all of the people directory providers.

### Group

Use this criteria to assign a group to a task role. This assignment creates a group work item instead of creating user work items for every assigned user.

This criteria is supported by all of the people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| GroupId | Mandatory | string | Group name of the users to retrieve. This parameter supports replacement expressions. The group ID must correspond to one of the following values:<br>• For virtual member manager, a unique name of a group entry<br>• For LDAP, a DN of a group entry<br>• For the user registry provider, the name format you use depends on which user repository is set for the application server where the task is deployed:<br>  – For the local operating system, use a group name that is supported by the local operating system<br>  – For a stand-alone custom registry, use a group name supported by the custom implementation<br>  – For a stand-alone LDAP registry, use a DN of a group entry |

## Group Members

Use this criteria to retrieve the members of a group. It is supported by the LDAP, virtual member manager, and the user registry people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| GroupName | Mandatory | string | Group name of the users to retrieve. This parameter supports replacement expressions. The group ID must correspond to one of the following values:<br>• For virtual member manager, a unique name of a group entry<br>• For LDAP, a DN of a group entry<br>• For the user registry provider, the name format you use depends on which user repository is set for the application server where the task is deployed:<br>  – For the local operating system, use a group name that is supported by the local operating system<br>  – For a stand-alone custom registry, use a group name supported by the custom implementation<br>  – For a stand-alone LDAP registry, use a DN of a group entry |
| IncludeSubgroups | Mandatory | boolean | Specifies whether nested subgroups are considered in the query. |
| 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 criteria to retrieve all of the members of a group, except for the explicitly named users. It is supported by the LDAP, virtual member manager, and the user registry people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| GroupName | Mandatory | string | Group name of the users to retrieve. This parameter supports replacement expressions. The group ID must correspond to one of the following values:<br>• For virtual member manager, a unique name of a group entry<br>• For LDAP, a DN of a group entry<br>• For the user registry provider, the name format you use depends on which user repository is set for the application server where the task is deployed:<br> – For the local operating system, use a group name that is supported by the local operating system<br> – For a stand-alone custom registry, use a group name supported by the custom implementation<br> – For a stand-alone LDAP registry, use a DN of a group entry |
| 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. This parameter supports replacement expression. |

### Group Members without Filtered Users

Use this criteria to retrieve all of the members of a group except for a set of users that is defined by a search filter. It is supported by the LDAP, and virtual member manager people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| GroupName | Mandatory | string | Group name of the users to retrieve. This parameter supports replacement expressions. The group ID must correspond to one of the following values:<br>• For virtual member manager, a unique name of a group entry<br>• For LDAP, a DN of a group entry |
| IncludeSubgroups | Mandatory | boolean | Specifies whether nested subgroups are considered in the query. |
| FilterAttribute | Mandatory | string | Name of the attribute to use in the search filter. |
| FilterValue | Mandatory | string | Filter value to use in the search filter. You can use the wildcard character, asterisk (*), in the filter. |

### Group Search

Use this criteria to search for a group based on attribute matches, and to assign the members of the group. It is supported by the LDAP, and virtual member manager people directory providers.

| 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. |
| 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. |

For virtual member manager, the Group entity has properties that are equivalent to the following Group Search criteria parameters:

- GS_GroupID: cn
- GS_DisplayName: displayName
- GS_BusinessCategory: businessCategory

### Manager of Employee

Use this criteria to retrieve the manager of a person using the person's name. It is supported by the LDAP and virtual member manager people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| EmployeeName | Mandatory | string | The name of the employee whose manager is retrieved. The employee name must correspond to one of the following values: <br>• For virtual member manager, the unique name of a person entry<br>• For LDAP, a DN of a person entry |

### Manager of Employee by user ID

Use this criteria to retrieve the manager of a person using the person's user ID. It is supported by the LDAP and virtual member manager people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| EmployeeUserID | Mandatory | string | The login user ID of the employee whose manager is retrieved. This parameter supports replacement expressions. |

### Native Query

Use this criteria to define a native query based on directory-specific parameters. It is supported by the LDAP and virtual member manager people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| QueryTemplate | Mandatory | string | The query template to use. This must be one of the following values: search, user, and usersOfGroup. |
| Query | Mandatory | string | Specifies the query. This parameter supports replacement expressions. The type of query depends on the query template.<br>• search template: search filter<br> – For virtual member manager, a valid search expression<br> – For LDAP, a valid LDAP filter<br>• user template: user DN<br> – For virtual member manager, a unique name of a user entry<br> – For LDAP, a DN of a user entry<br>• usersOfGroup: group DN<br> – For virtual member manager, a unique name of a group<br> – For LDAP, a DN of a group |
| AdditionalParameter1 | Optional | string | Specifies the query. This parameter supports replacement expressions. The type of parameter depends on the query template.<br>• search template. Used to specify whether recursive search is done. Supported values: `yes` and `no`<br>• user template. Not supported<br>• usersOfGroup. Used to specify whether recursive search is done. Supported values: `yes` and `no` |
| AdditionalParameter2 | Optional | string | Use this criteria to specify a base entry for searching.<br>• For virtual member manager, a unique name of a base entry, for example, `dc=mycomp, dc=com`<br>• For LDAP, a DN of a base entry |
| AdditionalParameter3 | Optional | string | Use this criteria to specify an additional parameter.<br><br>If you use the default mapping XSLT files, this parameter is not supported. |
| AdditionalParameter4 | Optional | string | Use this criteria to specify an additional parameter.<br><br>If you use the default mapping XSLT files, this parameter is not supported. |

| Parameter | Use | Type | Description |
|---|---|---|---|
| AdditionalParameter5 | Optional | string | Use this criteria to specify an additional parameter.<br><br>If you use the default mapping XSLT files, this parameter is not supported. |

### Nobody

Use this criteria to deny users access to a task role. Only authorization inheritance and people resolution defaults apply with this criteria. This criteria has no parameters.

### Person Search

Use this criteria to search for people based on attribute matches. It is supported by the LDAP, virtual member manager, and the user registry people directory providers.

| 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. |
| 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. |

For virtual member manager, the PersonAccount entity has properties that are equivalent to the following People Search criteria parameters:

- PS_UserID: uid
- PS_LastName: sn
- PS_FirstName: givenName
- PS_MiddleName: initials
- PS_Email: mail
- PS_DisplayName: displayName
- PS_Secretary: secretary

- PS_Manager: manager
- PS_Department: departmentNumber
- PS_Phone: telephoneNumber
- PS_PreferredLanguage: preferredLanguage

### Role Members

Use this criteria to retrieve the users associated with a role. It is supported by the LDAP and virtual member manager people directory providers.

| 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. |
| AlternativeRoleName1 | Optional | string | An additional role name for the user. |
| AlternativeRoleName2 | Optional | string | An additional role name for the user. |

### User Records by User ID

Use this criteria to define a query for a user whose user ID is known. It is supported by the LDAP, virtual member manager people directory providers. This criteria returns the user IDs, the e-mail information, and the preferred locale, if set, for these users.

| Parameter | Use | Type | Description |
| --- | --- | --- | --- |
| UserID | Mandatory | string | The user ID of the user to retrieve. This parameter supports replacement expressions. |
| 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 Records by User ID without Named Users

Use this criteria to define a query for users whose user ID is known, while excluding an explicitly named user ID. It is supported by the LDAP, virtual member manager, and user registry people directory providers. This criteria returns the user IDs and the e-mail information for these users.

| Parameter | Use | Type | Description |
| --- | --- | --- | --- |
| UserID | Mandatory | string | The user ID of the user to retrieve. This parameter supports replacement expressions. |
| 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. This parameter supports replacement expressions. |

## Users

Use this criteria to define a query for a user who is known by name. It is supported by all of the people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| Name | Mandatory | string | The name of the user to retrieve. <br> • For virtual member manager, the unique name of a person entry <br> • For LDAP, a DN of a person entry <br> • For the user registry provider, the name format you use depends on which user repository is set for the application server where the task is deployed: <br>   – For the local operating system, use the user ID of the user to assign <br>   – For a stand-alone custom registry, use a person name supported by the custom implementation <br>   – For a stand-alone LDAP registry, use a DN of a person entry |
| 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 criteria to define a query for a user whose user ID is known. Use short names to specify values, for example, wpsadmin. This criteria does not require access to a people directory. It is supported by all of the people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| UserID | Mandatory | string | The user ID of the user to retrieve. This parameter supports replacement expressions. |
| 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 criteria to define a query for users whose user ID is known, while excluding an explicitly named user ID. Use short names to specify values, for example, wpsadmin. This criteria does not require access to a people repository. It is supported by all of the people directory providers.

| Parameter | Use | Type | Description |
|---|---|---|---|
| UserID | Mandatory | string | The user ID of the user to retrieve. This parameter supports replacement expressions. |

| 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. |
| NamedUsers | Mandatory | string | The user IDs of the users to exclude from the user ID list. This parameter supports replacement expressions. |

**Related concepts**

"Replacement expressions in people assignment criteria definitions"
You can use replacement expressions as parameter values in some people assignment criteria definitions. The people resolution can resolve the assignment criteria at runtime, based on information supplied by the contexts.

## Replacement expressions in people assignment criteria definitions

You can use replacement expressions as parameter values in some people assignment criteria definitions. The people resolution can resolve the assignment criteria at runtime, based on information supplied by the contexts.

For example, the following people assignment criteria definition specifies the `htm:input.\name` replacement expression as a parameter:

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

This variable denotes the "name" element of the task input message value that is received by the task when it is initiated. People resolution dynamically replaces the expression with the actual task input message value.

**Related information**

"Predefined people assignment criteria" on page 68
Predefined people assignment criteria are provided for retrieving sets of users from people directories.

# People resolution

People resolution retrieves user information from people directories based on a set of parameterized query expressions, known as people assignment criteria.

**Related concepts**

"Authorization roles for business processes" on page 32
A role is a set of people who share the same level of authorization. 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.

## People directories

People directories store user information that is used for people resolution.

To support people resolution, the people directory must support the following attributes:

- The name that identifies a user profile and the login ID of a user

- To exploit information that is related to the manager of a user, the people directory should offer a corresponding attribute, by default the manager attribute
- To exploit the e-mail notification feature for escalations, the people directory should offer user e-mail addresses

Business Process Choreographer supports the following people directories for people resolution. If you want to exploit the full set of features offered by Business Process Choreographer for people assignment, use virtual member manager as the people directory.
- Federated repositories (also referred to as virtual member manager)

  This is the default people directory that is supported by WebSphere Application Server. It provides access to a variety of directory types, including Lightweight Directory Access Protocol (LDAP) directories, database and file-based repositories, and custom repositories. It also supports the federation of the repositories.

  Both person and group information can be retrieved. The supported person schema (PersonAccount entity type) includes properties for the name, login identity, manager identity, and e-mail address of a user. To be available for people resolution, federated repositories must be configured as the active security realm definition in WebSphere Application Server.
- An LDAP directory

  Business Process Choreographer can directly access an LDAP directory for people resolution without using WebSphere Application Server security. To ensure consistency across people resolution (implemented by Business Process Choreographer) and user authentication (implemented by WebSphere Application Server security), WebSphere Application Server security must be configured to access the same LDAP directory server as the one specified for people resolution in Business Process Choreographer.

  Depending on the LDAP person schema that you use, the person-related information includes the user name, identity, manager name, and e-mail address. To be available for people resolution, a Business Process Choreographer people directory provider configuration is required.
- WebSphere Application Server user registry

  The user registry is a subsystem of the application server for retrieving user information. Business Process Choreographer can use this user registry as a people directory. Business Process Choreographer uses its own user registry people directory provider to access the WebSphere Application Server user registry.

  **Related concepts**

  "Escalations" on page 48
  An escalation is an alert that is raised automatically when a human task is not actioned in the specified amount 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.

## People directory providers and configurations

Business Process Choreographer uses people directory providers as adapters for accessing people directories. You can configure virtual member manager, LDAP, the user registry, and the system people directory providers to retrieve user information.

The decision on which people directory provider to use depends on the support that you need from people resolution. To exploit all of the people assignment features offered by Business Process Choreographer, use virtual member manager.

All people directory providers are made available at the node level.

**Virtual member manager people directory provider**
> The virtual member manager people directory provider is used to access WebSphere Application Server federated repositories. You can use this provider to exploit the following aspects of people resolution:
>
> - Federated repository features, including the use of various repositories, such as file and database repositories, LDAP directories, the property extension repository, and the federation of repositories
> - E-mail notification for escalations
> - Substitution for absentees
> - All of the predefined people assignment criteria

**Lightweight Directory Access Protocol (LDAP) people directory provider**
> The LDAP people directory provider is used to access an LDAP directory directly without using WebSphere Application Server. In most cases, the WebSphere Application Server security realm is set to `Stand-alone LDAP registry`, and configured to point to the same LDAP directory as the one referenced by the LDAP people directory provider. You can use this provider to exploit the following aspects of people resolution:
>
> - E-mail notification for escalations
> - All of the predefined people assignment criteria

**User registry people directory provider**
> You can use the user registry people directory provider to access the following people directories with WebSphere Application Server: the local operating system, a stand-alone LDAP registry, or a stand-alone custom registry. The people directory that is used depends on the configuration of the application server security realm. You can use this provider to exploit the following aspects of people resolution:
>
> - Minimum configuration of the people directory provider for Business Process Choreographer because the repository is determined by the security realm for the application server
> - A limited set of predefined people assignment criteria. The user registry people directory provider can resolve users and groups, but not employee to manager relationships, user properties, or e-mail addresses.

**System people directory provider**
> The system people directory provider has limited people resolution support. Because the system provider supports only hard-coded queries, it is suitable only for test purposes.

All of the people directory configurations require that WebSphere Application Server administrative and application security are enabled.

Each of the people directory providers can be associated with one or more people directory provider configurations. All of the configurations, except the LDAP people directory provider, are ready to use. For the virtual member manager people directory provider, the federated repositories functionality must be configured in WebSphere Application Server. For the LDAP provider configuration, the required connection parameters must be set. In addition, the transformation file for the LDAP provider configuration must be customized.

Each of the configurations is uniquely identified by its Java Naming Directory (JNDI) name. The JNDI names are the link between a task template definition and the people directory configuration that is to be used for resolving the people assignments to task roles. Use WebSphere Integration Developer to specify the configuration name for a task template. If you are defining tasks at runtime using the task creation API, you can specify the configuration name directly in the API. Different task templates can reference different people directory configurations.

After a task template is deployed, the people directory configuration name is fixed for the lifetime of the deployed template. If you need to change the people directory that is associated with the template, use WebSphere Integration Developer to change the JNDI name of the people directory configuration that is defined for the task template definition, and deploy the template again.

> **Related tasks**
>
> "Configuring the LDAP people directory provider" on page 194
> Use this task to configure the Lightweight Directory Access Protocol (LDAP) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task.
>
> "Configuring the Virtual Member Manager people directory provider" on page 193
> Configure the Virtual Member Manager (VMM) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task. The default people directory provider is ready to use, and only needs to be configured if you introduce custom people assignment criteria.

## Mapping people assignment criteria to people queries

When an application is deployed, people assignment criteria definitions are transformed into sets of queries that are specific to a people directory configuration. The resulting people queries are stored with the task template in the Business Process Choreographer database.

If you use virtual member manager as the people directory, you need to change the predefined mappings in the transformation XSL file only if you define custom people assignment criteria.

A transformation (XSLT) file contains the instructions for translating the people assignment criteria. Each people directory configuration is associated with a transformation file. The following transformation files are provided for the default people directory configurations:

- LDAPTransformation.xsl for the LDAP people directory provider
- VMMTransformation.xsl for the virtual member manager people directory provider
- UserRegistryTransformation.xsl for the user registry people directory provider
- SystemTransformation.xsl and EverybodyTransformation.xsl for the system people directory provider

On Windows® platforms, these files are in the `install_root\ ProcessChoreographer\Staff` directory. On Linux®, UNIX®, and i5/OS® platforms, these files are in the `install_root/ProcessChoreographer/Staff` directory.

## People queries for a specific people directory provider

An XSL transformation file that is associated with a people directory configuration is used to generate people queries that are specific to a particular repository. Each query can be executed by the respective people directory provider to obtain a list of user IDs. The predefined queries that are available to a people directory provider correspond to the calls that can be executed by the provider, and are therefore fixed.

The set of repository-specific queries provided by a people directory provider correspond to the methods it can use to retrieve user information from the corresponding people directory. You can uses this set queries to form more complex queries as shown in the following examples:

- Combine query results so that the user IDs returned by the individual queries are added to the current result list of user IDs. For example, the LDAP people directory provider allows the following predefined queries:
  - The list of user IDs for the group members of a specified group:

    ```
    <sldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
    ...
    </sldap:usersOfGroup>
    ```

  - The distinguished name (DN) of the specified user:

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

  - A complex query can be constructed for a list of user IDs for the members of a specified group, and the DN of a specified user:

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

- Remove query results from the current result list. For example, the following snippet shows how to remove "user1" from the list of IDs retrieved for the specified group members:

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

- Use the query results obtained from one query to influence the behavior of 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". This variable is then used to look up the manager's LDAP entry and retrieve the associated user ID.

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

People queries that are constructed according to these combination rules can be executed by the people directory providers.

## Customizing the transformation of people assignment criteria

You might need to customize the transformation of the people assignment criteria in the following situations: to adapt the transformation to the LDAP person and group schemas, or to define custom people assignment criteria.

To make use of a customized transformation, you must create a transformation file. Do not change the default transformation files and do not reuse the names of these files for your transformation files. For new transformation files for the LDAP or virtual member manager people directory providers, always include the **User Records by User ID** people assignment criteria definition in the file. This definition is required by Business Process Choreographer even if it is not explicitly specified for the people assignment.

To use the new transformation file, define a new people directory configuration that points to the file.

> **Related tasks**
>
> "Configuring the LDAP people directory provider" on page 194
> Use this task to configure the Lightweight Directory Access Protocol (LDAP) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task.
>
> "Configuring the Virtual Member Manager people directory provider" on page 193
> Configure the Virtual Member Manager (VMM) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task. The default people directory provider is ready to use, and only needs to be configured if you introduce custom people assignment criteria.

**Defining custom people assignment criteria:**

You might need to extend the set of predefined people assignment criteria with criteria of your own.

When you create a custom people assignment criteria, you need to include it in the following files:

* In the VerbSet.xml file
* In the transformation file

To make use of a customized transformation, you must create a transformation file. Do not change the default transformation files and do not reuse the names of these files for your transformation files.

For example, you created a new criteria Mentor of Employee. This criteria is similar to the Manager of Employee criteria, but it retrieves the user ID of the employee's mentor instead of the employee's manager.

1. Add the following XML snippet to the VerbSet.xml file:

```
   <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>
```

2. In the transformation file, define a new XSL variable to contain the LDAP attribute for the person's mentor:

```
<xsl:variable name="DefaultMentorAttribute">mentor</xsl:variable>
```

3. Add the following XML snippet to the transformation file:

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

    <sldap:intermediateResult>
      <xsl:attribute name="name">mentorvar</xsl:attribute>
      <sldap:user>
        <xsl:attribute name="dn">
          <xsl:value-of select="staff:parameter[@id='EmployeeName']"/>
        </xsl:attribute>

    <sldap:resultObject>
      <xsl:attribute name="objectclass">
        <xsl:value-of select="$DefaultPersonClass"/>
      </xsl:attribute>
      <xsl:attribute name="usage">simple</xsl:attribute>

      <sldap:resultAttribute>
        <xsl:attribute name="name">
          <xsl:value-of select="$DefaultMentorAttribute"/>
        </xsl:attribute>
        <xsl:attribute name="destination">intermediate</xsl:attribute>
      </sldap:resultAttribute>
    </sldap:resultObject>



    <sldap:user>
      <xsl:attribute name="dn">%mentorvar%</xsl:attribute>
        <xsl:call-template name="ResultObjectSpecForUserData"/>
    </sldap:user>
  </sldap:staffQueries>
</xsl:template>
<!-- End template ManagerOfEmployee -->
```

## Substitution for absentees

The substitution feature allows you to specify absence settings either for yourself, or for members of the group that you administer. A substitution policy defines how to deal with tasks and escalations that are assigned to absent users.

The substitution policy is defined when the task template is modeled. The same policy is applied for all of the task roles that are associated with a task template. After the task template is deployed, you cannot change the policy.

If a user is absent, the substitution policy is applied to the results of the people resolution to determine who receives the work items instead of the absent user. It

is applied only to task roles that have people assignment criteria. This means that task originators, starters, or owners are not subject to substitution. Similarly, substitution is refreshed if the people assignment criteria get refreshed.

Depending on the specific substitution policy, the following actions are applied:

**No substitution (default)**
> The set of users remains unchanged

**Substitute if the user is absent**
- For every user that is present, the user itself is used.
- For every user that is absent, the first substitute that is present is used.
- If none of the users and none of their substitutes are present, then the default people assignment rules apply.

**Select user if present**
- For every user that is present, the user is used.
- Substitutes are not taken into account.
- If none of the users is present, then the original set of users is used, that is, the fact that they are absent is disregarded.

The substitution feature requires virtual member manager as the people directory. To make virtual member manager available for substitution, the federated repositories must be configured as the active security realm in WebSphere Application Server. Ensure that you enable substitution for Human Task Manager in the administrative console. If you deploy a task template with a non-default substitution policy to a people directory provider other than virtual member manager, the deployment fails.

> **Related tasks**
>
> "Configuring people substitution" on page 199
> Create and activate a Virtual Member Manager (VMM) property extension repository for Business Process Choreographer to support user substitution.
>
> "Specifying absence settings" on page 351
> If you intend to be away from the office for a certain time, specify a substitute for your tasks.
>
> "Specifying absence settings for users" on page 352
> If users are prevented from working on their tasks, for example, if they are on sick leave, specify a substitute for the user's tasks and a date range for which the delegation is applicable.

# Default people assignments

Default people assignments are performed if you do not define people assignment criteria for certain task roles, or if people resolution fails or does not return a result. The default assignments differ for inline tasks and stand-alone tasks.

## Inline tasks

The following table shows the default people assignments for inline tasks.

| Roles for inline human tasks and their escalations | If the role is not defined in the task model ... | If people assignment fails... |
|---|---|---|
| Task administrator | Only inheritance applies | Only inheritance applies |
| Task potential instance creator | Everybody becomes potential instance creator | Everybody becomes potential instance creator |

| Roles for inline human tasks and their escalations | If the role is not defined in the task model ... | If people assignment fails... |
|---|---|---|
| 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 to 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.

## Stand-alone tasks

The following table shows the default people assignments for stand-alone tasks.

| Roles for stand-alone human tasks and their escalations | If the role is not defined in the task model ... | If people assignment fails... |
|---|---|---|
| Task administrator | Originator becomes administrator | 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 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 to 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.

When a method is invoked using the Business Flow Manager API, members of the BPESystemAdministrator role have administrator authorization, and members of the BPESystemMonitor role have reader authorization. When a method is invoked via the Human Task Manager API, members of the TaskSystemAdministrator role have administrator authorization, and members of the TaskSystemMonitor role have reader authorization.

**Related concepts**

"Authorization roles for human tasks" on page 64
Actions that you can take on human tasks depend on your authorization role. This role can be a system-level J2EE role or an instance-based role.

## Managing people assignment criteria and people resolution results

A people assignment criteria that is associated with a task authorization role is valid throughout the lifetime of a deployed task template or task instance.

If you need to change the people assignment criteria, you must change the task definition in WebSphere Integration Developer, and deploy the task template again.

The people query that is derived from the people assignment criteria is stored as part of the deployed task template, or task instance. During the execution of a task, the authorization roles require the resolution of the associated people queries.

The result of a people query depends on the content of the people directory, which might change over time. For example, new members might be added to a people group. To reflect changes in the people directory, a people query must be refreshed in one of the following ways:

- Explicitly by an administrator

  An administrator can use either the administrative console or the administrative commands to refresh people query results. Commands exist for the following actions:
  - Refreshing all of the people query results at once
  - Refreshing all of the people query results that are associated with a task template
  - Refreshing people query results that contain a specific user ID in the current result.
- Triggered by a scheduled refresh of expired people queries

  This approach is based on the following parameters:
  - A timeout value for people query results ($T_{out}$).
  - A refresh schedule for the people query. Use the WebSphere Application Server CRON-syntax for defining the schedule, for example, every Monday at 1 pm, or every work-day at midnight.

  The following parameters determine how people queries are automatically refreshed:
  - When a query is run for the first time or it is refreshed, the query result gets an expiration timestamp ($t_{exp} = t_{current} + T_{out}$)

– When the query refresh daemon is invoked, all of the people queries with results that have expired are run again

You can set the timeout value to be above the schedule refresh interval. For example, you can set the timeout value to be 24h, and the refresh interval to 1h. In this way, you can spread the updates to the people queries throughout the day and avoid the overhead of refreshing all of the people query results at once.

**Related tasks**

"Refreshing people query results, using the administrative console" on page 287 The results of a people query are static. Use the administrative console to refresh people queries.

"Refreshing people query results, using administrative commands" on page 306 The results of a people query are static. Use the administrative commands to refresh people queries.

"Refreshing people query results, using the refresh daemon" on page 289 Use this method if you want to set up a regular and automatic refresh of all expired people query results.

# Sharing people assignments

For a specific task role, the same people assignment criteria are used in all instances of a task template. This is because all of the task instances are instantiated from the same task template. To avoid rerunning people queries, the result of a query is shared across task instances of a task template.

The sharing of results applies only if a people assignment criteria definition contains fixed parameter values. Such values, for example, the name of a group: `cn=group1, cn=groups`, imply that the corresponding people query result is the same, regardless of the task instance context in which the people query is resolved.

If the people assignment criteria definition contains replacement variables, the sharing scope is reduced to people assignments that have the same replacement variable values. For example, a parameter value can depend on parts of the input message of a task. Because different task instances can have different input messages, the parameter values for the people queries differ, too.

If you post process people query results, sharing does not apply to these results by default. To enable the sharing of post-processed results, complete the following steps in the administrative console:

1. If Business Process Choreographer is configured on a server, click **Servers** → **Application Servers** → *server_name*.
2. If Business Process Choreographer is configured on a cluster, **Servers** → **Clusters** → *cluster_name*.
3. Under **Business Integration**, click **Business Process Choreographer** → **Human Task Manager** → **[Additional Properties] Custom Properties**.
4. Change the value of the **Staff.PostProcessorPlugin.EnableResultSharing** custom property to `true`, and save your changes
5. Restart the server or cluster to make the changes effective.

**Related tasks**

"Creating plug-ins to post-process people query results" on page 543 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 people queries returned by people 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.

# Part 2. Planning and configuring Business Process Choreographer

# Chapter 3. Planning to configure Business Process Choreographer

Plan your Business Process Choreographer setup and configuration parameters.

**Procedure**

1. Perform "Planning the topology, setup, and configuration path."
2. Depending on your chosen configuration path, perform one of the following:
   - For "Basic sample", perform "Planning to create a basic sample Business Process Choreographer configuration" on page 95.
   - For "Sample with organization", perform "Planning to create a sample Business Process Choreographer configuration including a sample organization" on page 96.
   - For "Non-production deployment environment", perform "Planning a non-production deployment environment configuration" on page 96.
   - For "Production deployment environment", perform "Planning to use the administrative console's deployment environment wizard" on page 97.
   - For "Flexible custom configuration", perform "Planning for a custom Business Process Choreographer configuration" on page 100.

**Results**

You have planned everything you need to be able to perform Chapter 4, "Configuring Business Process Choreographer," on page 141.

**Related concepts**

"About Business Process Choreographer" on page 125
Describes the facilities provided by the Business Flow Manager and the Human Task Manager.

## Planning the topology, setup, and configuration path

Your choice of topology and setup affects which Business Process Choreographer configuration paths you can use.

**About this task**

The different configuration paths vary in complexity, flexibility, and their support for different topologies and databases.

**Procedure**

1. Be aware that you must choose between five different configuration paths.
   - "Basic sample"
   - "Sample with organization"
   - "Non-production deployment environment"
   - "Production deployment environment"
   - "Flexible custom configuration"

   For most configuration paths, you have a choice of configuration tools.

2. Be aware of the different configuration tools that you can use to configure Business Process Choreographer.

**Installer or Profile Management Tool**
>    Provide the easiest ways to create a non-production system, and they require the least planning.
>    - The "Basic sample" configuration includes the following Business Process Choreographer components:
>        – Business Process Choreographer
>        – Explorer
>        – Observer and event collector
>    - The "Sample with organization" configuration also includes a people directory that is preconfigured with 15 users in a sample organization, and has substitution and group work items enabled.
>    - The "Non-production deployment environment" configuration provides an easy way to configure Business Process Choreographer on a cluster, but Business Process Choreographer cannot have its own database, instead, it uses the common WPRCSDB database.

**Administrative console's deployment environment wizard**
>    Can be used to create a "Production deployment environment" Business Process Choreographer configuration, based on a deployment environment pattern.

**Administrative console's Business Process Choreographer configuration page**
>    You can use this administrative console page to configure a "Flexible custom configuration" Business Process Choreographer production system on a server or cluster. It provides the opportunity to set many configuration parameters, which need detailed planning. This page does not configure the Business Process Choreographer Explorer or Business Process Choreographer Observer; they have their own configuration pages, or can be configured by running scripts. This configuration path is most suitable for creating production systems.

**bpeconfig.jacl configuration script**
>    You can use this script to configure a "Flexible custom configuration" Business Process Choreographer production system and all the necessary resources on a given server or cluster. You can run the script interactively, or if you provide all the necessary parameters, it can run in batch mode for repeatable automation. It can create a local database, the necessary messaging resources, and can optionally configure the Business Process Choreographer Explorer and Business Process Choreographer Observer. For some database systems, it can also create a remote database. This configuration path is most suitable for creating production systems.

3. Be aware that some of the configuration paths have restrictions that limit their suitability for production systems: For example:
   - After experimenting with one of the sample configurations, it must be removed before you can create a configuration that is suitable for a production system.
   - If you create a configuration that uses a Derby Embedded database, or the common WPRCSDB database, it will not be suitable for a high-performance system. You must remove the configuration before you can create a new configuration that uses a separate high-performance database.

- If you use a FILESTORE or Derby Embedded message store, then you cannot federate the profile into a network deployment environment. To be able to federate the profile, you would have to completely remove your Business Process Choreographer configuration and create a new configuration that uses a remotely accessible database for the message store.

4. Identify the main criteria for deciding which configuration path to use. Use the following table to identify choices and constraints:

*Table 4. Criteria for selecting a configuration path*

| Suitable for a production system? | Deploy-ment target | Business Process Choreographer configuration | Can have a separate BPEDB database? | Message stores supported for the messaging engine | Suitable configuration name, tools, and options |
|---|---|---|---|---|---|
| No | Stand-alone server | Basic sample (without sample organization, people assignment, and substitution) | Yes, but only Derby Embedded | Only Derby Embedded | **"Basic sample"** using one of: <br> • Installer <br> • Profile Management Tool <br><br> Select the options: <br> • **Stand-alone server profile** <br> • **Typical** <br> • **Enable administrative security** |
| | | Sample including a 15 person organization, people assignment, and substitution enabled. <br><br> This sample is identical to the sample that is available in WebSphere Integration Developer. | | Derby Embedded, File Store, or WPRCSDB | **"Sample with organization"** using: <br> • Profile Management Tool <br><br> Select the options: <br> • **Stand-alone server profile** <br> • **Advanced** <br> • **Create server from development template** <br> • **Enable administrative security** |
| | Cluster | Choice of deployment environment patterns: <br> • Remote Messaging and Remote Support <br> • Remote Messaging <br> • Single Cluster | No, it shares WPRCSDB, which can be any database except Derby Embedded | Shares WPRCSDB, which can be any supported database except File Store and Derby Embedded | **"Non-production deployment environment"** using one of: <br> • Installer <br> • Profile Management Tool <br><br> Select: **Deployment environment** |

*Table 4. Criteria for selecting a configuration path  (continued)*

| Suitable for a production system? | Deploy- ment target | Business Process Choreographer configuration | Can have a separate BPEDB database? | Message stores supported for the messaging engine | Suitable configuration name, tools, and options |
|---|---|---|---|---|---|
| Yes | Cluster | Choice of deployment environment patterns:<br>• Remote Messaging and Remote Support<br>• Remote Messaging<br>• Single Cluster<br>• Custom | Yes, any supported database except Derby Embedded | Any supported database except File Store and Derby Embedded | **"Production deployment environment"** using:<br>• Administrative console<br><br>Select: **Deployment environment** |
| | | Flexible custom configuration | Yes, any supported database | Any supported database except File Store and Derby Embedded | **"Flexible custom configuration"** using one of:<br>• bpeconfig.jacl script<br>• Administrative console Business Process Choreographer configuration page |
| | Stand- alone server | | | Any supported database, or File Store | |

**Note:**  It is also possible to use any of the configuration paths that are recommended for creating a production system to create a configuration that is not suitable for a production system.

Consider the following options:

a. Decide whether you are configuring a production system. Typically a production system requires high-performance, scalability, and security. For Business Process Choreographer, a production system should have its own non-Derby BPEDB database.

b. Decide whether the deployment target for the Business Process choreographer will be a stand-alone server or a cluster.

c. If you do not want to create a production system, decide whether a sample configuration on a stand-alone server will meet your needs. If so, decide whether you want the sample to include a sample people directory for people assignment and substitution enabled.

   **Note:** The sample people directory uses the file registry for the federated repository, and includes all sample people with the same password "wid". The WebSphere administration user ID is also added to the directory, using the password that was specified during profile creation.

d. If you want to configure Business Process Choreographer on a cluster, depending on your performance requirements, decide whether the messaging engines and supporting applications, (such as the Business Process Choreographer Explorer, Observer, and Common Event Infrastructure) will have their own cluster, or share one. The standard deployment environment patterns are:

   **Remote Messaging and Remote Support**
   Three clusters are used. One each for the applications, messaging engines, and support applications.

> **Remote Messaging**
>> One cluster is used for the applications and support functions. A second cluster is used for the messaging engines.
>
> **Single cluster**
>> Only one cluster is used for applications, messaging engines, and support applications.
>
> **Custom**
>> More flexible setup.

    e. Decide whether you want a dedicated BPEDB database for Business Process Choreographer.

    f. Decide whether the Business Process Choreographer messaging engine will share the WPRCSDB database or have its own message store, and whether it will use File Store or a database system.

    g. If you want to use the Business Process Choreographer Observer, you can either configure it at the same time as you create a Business Process Choreographer configuration, or you can create it later. Decide whether Business Process Choreographer Observer will also use the BPEDB database, or whether it will have its own, OBSRVRDB, database. Also plan the topology for the Business Process Choreographer Observer components. For more information, see "Planning for the Business Process Choreographer Observer" on page 122.

5. If you want WebSphere Portal to access Business Process Choreographer, plan to configure a Business Process Choreographer client on the portal server. Similarly, you can configure a Business Process Choreographer client to enable any custom WebSphere Process Server client to access Business Process Choreographer.

**Results**

You have planned the topology and know which configuration path and configuration tool you will use.

## Planning to create a basic sample Business Process Choreographer configuration

This basic sample, for a stand-alone server, does not include people assignment.

**Before you begin**

You have performed "Planning the topology, setup, and configuration path" on page 91, and have selected the "Basic sample" configuration path.

**Procedure**

1. Decide whether you will create the sample using the Installer or Profile Management Tool. The sample is identical in both cases, the only difference is which tool you use.

2. If you want the Human Task Manager to be able to send escalation e-mails, plan the following:
   - If there will not be a local Simple Mail Transfer Protocol (SMTP) mail server available, plan to change the mail session later to point to a suitable mail server.

- Plan to change the sender address for the e-mails. Otherwise, it will use a dummy sender address.
3. Be aware that this sample configuration uses the WebSphere administrator user ID and password for the various Business Process Choreographer user IDs.

# Planning to create a sample Business Process Choreographer configuration including a sample organization

This sample includes a 15 person sample organization, which is suitable for experimenting with people assignment and substitution on a stand-alone server. This sample is identical to the sample that is available in WebSphere Integration Developer.

**Before you begin**

You have performed "Planning the topology, setup, and configuration path" on page 91, and have selected the "Sample with organization" configuration path.

**About this task**

This sample Business Process Choreographer configuration requires minimal planning.

**Procedure**

1. Be aware that this sample can only be created using the Profile Management Tool. To get this sample, you must select the following options:
   - **Stand-alone server profile**
   - **Advanced**
   - **Create server from development template**
   - **Enable administrative security**

   If for example, you do not enable administrative security, the sample Business Choreographer configuration will not be created.
2. Decide whether the Business Process Choreographer messaging engine will use file store, an embedded Derby database, or the common, WPRCSDB, database.
3. If you want the Human Task Manager to be able to send escalation e-mails, plan the following:
   - If there will not be a local Simple Mail Transfer Protocol (SMTP) mail server available, plan to change the mail session later to point to a suitable mail server.
   - Plan to change the sender address for the e-mails. Otherwise, it will use a dummy sender address.
4. Be aware that this sample configuration uses the WebSphere administrator user ID and password for the various Business Process Choreographer user IDs.

# Planning a non-production deployment environment configuration

Planning to use the Installer or Profile Management Tool to create a Business Process Choreographer configuration that is based on a deployment environment pattern.

**Before you begin**

You have performed "Planning the topology, setup, and configuration path" on page 91, and have selected the "Non-production deployment environment" configuration path.

**About this task**

When using the deployment environment wizard, you must select the deployment environment pattern, then you have the opportunity to change the default database parameters and authentication aliases for the WBI_BPC component, and enter other parameters for Business Process Choreographer.

**Procedure**

1. Decide which deployment environment pattern you will use:
   - **Remote Messaging and Remote Support**
   - **Remote Messaging**
   - **Single Cluster**
2. Plan the user name for the Business Process Choreographer JMS authentication alias that you will enter during the Security step.
3. Plan the context roots for the Business Process Choreographer step:

   **Business Process Choreographer Explorer context root**
   > This defines part of the URL that browsers must use to reach the Business Process Choreographer Explorer.

   **Business Process Choreographer Observer context root**
   > This defines part of the URL that browsers must use to reach the Business Process Choreographer Observer.

4. Plan the security parameters for the Business Process Choreographer step. These user IDs and groups will be used for the Business Flow Manager and Human Task Manager:

   **Administrator User and Group**
   > Plan a list of user IDs or a list or groups, or both, onto which the business administrator role is mapped.

   **Monitor User and Group**
   > Plan a list of user IDs or a list or groups, or both, onto which the business monitor role is mapped.

   **JMS API Authentication User and Password**
   > The run-as user ID for the Business Flow Manager message driven bean.

   **Escalation User Authentication User and Password**
   > The run-as user ID for the Human Task Manager message driven bean.

5. If you want to use people assignment, perform "Planning for the people directory provider" on page 119.

# Planning to use the administrative console's deployment environment wizard

For a production system plan all the configuration parameters for Business Process Choreographer, including a separate database. For a non-production system you can use a shared database.

**Before you begin**

You have performed "Planning the topology, setup, and configuration path" on page 91, and have selected the "Production deployment environment" configuration path.

**About this task**

When using the deployment environment wizard, you must select the deployment environment pattern, then you have the opportunity to change the default database parameters and authentication aliases for the WBI_BPC component, and enter other parameters for Business Process Choreographer.

**Procedure**

1. If you do not have enough information or authority to create the whole configuration on your own, consult and plan with the people who are responsible for other parts of the system. For example:
   - You might need to request information about your organization's LDAP server, if it uses authentication you will need to request a user ID, and authorization.
   - If you are not authorized to create the database, your database administrator must be included in planning the databases, and will need a copy of the database scripts to customize and run.
2. Perform "Planning security, user IDs, and authorizations" on page 101.
3. Decide which deployment environment pattern you will use:
   - **Remote Messaging and Remote Support**
   - **Remote Messaging**
   - **Single Cluster**
   - **Custom**
4. If you chose the **Custom** deployment environment pattern:
   a. Decide if you want to install the Business Process Choreographer Explorer. If so, plan where you will deploy it.
   b. Decide if you want to install the Business Process Choreographer event collector. If so, plan where you will deploy it.
   c. Decide if you want to install the Business Process Choreographer Observer. If so, plan where you will deploy it.
   d. Plan the context root for the SCA bindings.
   e. Plan whether you want to enable or disable the state observers and audit logging.
5. If you are planning to have dedicated databases for the following:
   - The BPEDB database for Business Process Choreographer – component WBI_BPC.
   - The OBSRVRDB database for the Business Process Choreographer Observer – component WBI_BPCEventCollector.
   - The BPEMEDB database for the Business Process Choreographer messaging engine – component WBI_BPC_ME.

   Plan the following parameters for each database, to enter on the wizard's database page:

   **Database Instance**
   >  The name of the data source, for example, BPEDB, OBSRVRDB, or BPEMEDB

instead of the default value, WPRCSDB, which results in sharing the common database. The default value is only suitable for lower performance setups.

**Schema**
The database schema qualifier to be used.

**Create Tables**
If selected, the tables will be created automatically the first time that the database is accessed. For this option to work, the database must already exist, and the user name provided for creating the data source must have the authority to create tables and indexes in the database. If not selected, the tables will not be created automatically, and you must create the tables manually by running scripts. For a production system, clear this option, and plan to use the provided SQL scripts to setup the database.

**User Name and Password**
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 option to create the tables automatically can be used, and when necessary, the database schema will be updated automatically after applying a service or fix pack.

**Server** The address of the database server. Specify either the host name or the IP address.

**Provider**
The JDBC provider.

6. Plan the user name for the Business Process Choreographer JMS authentication alias that you will enter during the Security step.

7. Plan the context roots for the Business Process Choreographer step:

**Business Process Choreographer Explorer context root**
This defines part of the URL that browsers must use to reach the Business Process Choreographer Explorer.

**Business Process Choreographer Observer context root**
This defines part of the URL that browsers must use to reach the Business Process Choreographer Observer.

8. Plan the security parameters for the Business Process Choreographer step. These user IDs and groups will be used for the Business Flow Manager and Human Task Manager:

**Administrator User and Group**
Plan a list of user IDs or a list or groups, or both, onto which the business administrator role is mapped.

**Monitor User and Group**
Plan a list of user IDs or a list or groups, or both, onto which the business monitor role is mapped.

**JMS API Authentication User and Password**
The run-as user ID for the Business Flow Manager message driven bean.

**Escalation User Authentication User and Password**
The run-as user ID for the Human Task Manager message driven bean.

9. If you want to configure an e-mail session for the Human Task Manager escalations, plan the following parameters for the Business Process Choreographer step:

   **Mail transport host**
   > The hostname or IP address where the Simple Mail Transfer Protocol (SMTP) e-mail service is located.

   **Mail transport user and Mail transport password**
   > If the mail server does not require authentication, you can leave these fields empty.

   **Business Process Choreographer Explorer URL**
   > This URL is used to provide a link in generated e-mails so that a business administrator who receives an e-mail notification can click on the link to view the related business process or human task in their Web browser.

10. If you want to use people assignment, perform "Planning for the people directory provider" on page 119.

# Planning for a custom Business Process Choreographer configuration

Plan the configuration parameters and options for creating a custom configuration, using either the Administrative console's Business Process Choreographer configuration page or the bpeconfig.jacl configuration script.

**Before you begin**

You have performed "Planning the topology, setup, and configuration path" on page 91, and have selected the "Flexible custom configuration" configuration path.

**Procedure**
1. Know which of the following you will use to configure Business Process Choreographer:
   - Administrative console's Business Process Choreographer configuration page
   - The bpeconfig.jacl configuration script
2. If you do not have enough information or authority to create the whole configuration on your own, consult and plan with the people who are responsible for other parts of the system. For example:
   - You might need to request information about your organization's LDAP server, if it uses authentication you will need to request a user ID, and authorization.
   - If you are not authorized to create the database, your database administrator must be included in planning the databases, and will need a copy of the database scripts to customize and run.
3. "Planning security, user IDs, and authorizations" on page 101
4. "Planning the databases for Business Process Choreographer" on page 107
5. "Planning for the Business Flow Manager and Human Task Manager" on page 118
6. "Planning for the people directory provider" on page 119
7. "Planning for the Business Process Choreographer Explorer" on page 121
8. "Planning for the Business Process Choreographer Observer" on page 122

9. If you will use the Administrative console's Business Process Choreographer configuration page, make sure that you have planned all the values that you will enter on the configuration page, which is described in "Business Process Choreographer configuration" on page 130.

10. If you will use the bpeconfig.jacl configuration script:

    a. Make sure that you have planned all the options and parameter values that you must specify on the command-line, or in a batch file. The options and parameters are summarized in "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157, and are described in detail in "bpeconfig.jacl script file" on page 163.

    b. If you will use a batch file to run the bpeconfig.jacl configuration script, create the batch file or shell script.

**Results**

You have planned everything you need to be able to perform Chapter 4, "Configuring Business Process Choreographer," on page 141.

## Planning security, user IDs, and authorizations

Plan the user IDs and authorizations for configuring Business Process Choreographer.

**About this task**

During configuration, you need to use various user IDs and you must specify other user IDs that will be used at runtime. Make sure that you plan and create all user IDs before you start configuring Business Process Choreographer.

**For a sample Business Process Choreographer configuration:**
> You only need the authority to create a new profile. In the Profile Management Tool, using the option to create a typical profile, when you enable administrative security, the Business Process Choreographer sample will also be configured. No other planning or user IDs are required, and you can skip this task.

**For a high security configuration:**
> You must plan all user IDs in detail as described in this task.

**For a low security configuration:**
> If you do not require full security, for example for a non-production system, you can reduce the number of user IDs that are used. You must plan all user IDs in detail, but you can use certain user IDs for multiple purposes. For example, the database user ID used to create the database schema can also be used as the data source user name to connect to the database at runtime.

**If you will use the bpeconfig.jacl script to configure Business Process Choreographer:**

> The user ID used to run the bpeconfig.jacl script must have, or specify user IDs as parameters, that have the necessary rights for the configuration actions that the script will perform. In this case you must plan all user IDs in detail. For user IDs that can be specified as parameters to the bpeconfig.jacl script, the parameter names are included in the table.The profile must already exist. If WebSphere administrative security is enabled, you need a WebSphere administrator user ID in the `configurator` role that you can use invoke the wsadmin tool.

**Procedure**

1. Print a hardcopy of this page so that you can write your planned values in the last column. Keep it for reference when you are configuring Business Process Choreographer, and keep it in your records for future reference.

2. Plan the user ID you will use on the WebSphere Process Server to configure Business Process Choreographer.

*Table 5. Planning user IDs for WebSphere Process Server*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| "Configurator" or "Administrator" role | Configuring | Logging onto the administrative console and running administrative scripts. | WebSphere administrator or configurator role, if WebSphere administrative security is enabled. | |
| | | If you are going to run the bpeconfig.jacl script to configure the Business Process Choreographer. | When running the script, you must also provide any user IDs that are necessary for the options that you select. For more information see "bpeconfig.jacl script file" on page 163. | |

3. Plan which people need access to subdirectories of *install_root*. If your security policy does not allow these people to be granted this access, they will need to be given copies of the files in the directories.

*Table 6. Planning access to the subdirectories of install_root*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| Database administrator | Configuring | Running the scripts to setup the following databases:<br><br>**BPEDB**: The database for Business Process Choreographer.<br><br>**OBSRVDB**: The database for the Business Process Choreographer Observer. | If you use the bpeconfig.jacl script to configure Business Process Choreographer:<br><br>Read access to (or a copy of) the createSchema.sql script that bpeconfig.jacl generates in a subdirectory of the directory:<br>• On Windows platforms: *profile_root*\ dbscripts\ProcessChoreographer\<br>• On Linux, UNIX, i5/OS, and in the UNIX System Services (USS) on z/OS platforms:*profile_root*/dbscripts/ ProcessChoreographer/ | |
| | | | If you do **not** use the bpeconfig.jacl script to configure Business Process Choreographer:<br><br>Read access to (or a copy of the files in) the database scripts provided in the directory:<br>• On Windows platforms: *install_root*\dbscripts\ ProcessChoreographer\*database_type*<br>• On Linux, UNIX, i5/OS, and in the UNIX System Services (USS) on z/OS platforms: *install_root*/dbscripts/ProcessChoreographer/ *database_type*<br><br>Where *database_type* is one of the following:<br>• DB2<br>• DB2zOSV7<br>• DB2zOSV8<br>• Db2iSeries<br>• Derby<br>• Informix<br>• Oracle<br>• SQLServer | |
| Integration developer | Customizing | To use people assignment with a Lightweight Directory Access Protocol (LDAP) or Virtual Member Manager (VMM) people directory provider, you will have to customize a copy of the sample XSL transformation file. | Read access to (or a copy of the files in) the directory:<br>• On Windows platforms: *install_root*\ ProcessChoreographer\Staff<br>• On Linux, UNIX, and i5/OS platforms: *install_root*/ProcessChoreographer/Staff<br><br>The integration developer will also need write access to a suitable directory to make the customized XSL transformation file available to the server. | |

4. Plan the user IDs that will be used to create, configure, and access the database that is used by Business Process Choreographer.

*Table 7. Planning user IDs for the BPEDB database*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| Database administrator | Before configuring | To create the BPEDB database instance. For Oracle: To create the BPEDB database. | Create the database. | |
| Database administrator or an administrator who will run the bpeconfig.jacl script | Configuring | You or your database administrator must run Business Process Choreographer database scripts, unless you are using the embedded Derby database. | For the BPEDB database: Alter tables, connect, insert tables, and create indexes, schemas, tables, table spaces, and views. | |
| Data source user name<br><br>If you use the bpeconfig.jacl script, this is the -dbUser parameter. | Configuring | If you select the **Create Tables** option, this user ID is used to create the database tables. | To use the **Create Tables** configuration option, this user ID must also be authorized to perform the following actions on the BPEDB database: Alter tables, connect, insert tables, and create indexes, tables, and views. | |
| | Runtime | The Business Flow Manager and Human Task Manager use this user ID to connect to the BPEDB database. | This user ID must be authorized to perform the following actions on the BPEDB database: Connect, delete tables, insert tables, select tables and views, and update tables. | |
| | After applying service or a fix pack | When necessary, the database schema is updated automatically after applying service. This only works if this user ID has the necessary database rights, otherwise schema updates must be performed manually. | This user ID must be authorized to perform the following actions on the BPEDB database: Alter, create, insert and select tables. Connect to the database. Create and drop indexes and views. | |

5. If you will configure the Business Process Choreographer Observer, plan the user IDs to use to create, configure, and access the database by the event collector will be used for the Business Process Choreographer Observer.

*Table 8. Planning user IDs for the OBSRVRDB database*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| Database administrator | Before configuring | To create the OBSRVRDB database instance. For Oracle, to create the OBSVRDB database. | Create the database. | |
| Database administrator or an administrator | Configuring | Running the setupEventCollector tool, or SQL scripts to create the schema. | For the OBSRVRDB database: Alter tables, connect, create function, insert tables, and create indexes schemas, tables, table spaces, and views.<br><br>If you are going to use the Java implementation of the user-defined functions, the user ID must also be authorized to install the JAR file. | |

*Table 8. Planning user IDs for the OBSRVRDB database  (continued)*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| Event collector data source user name | Runtime | Connecting to the observer database. If you are using the Business Process Choreographer Observer and it uses the BPEDB database, use the same user name as for the Business Process Choreographer data source. | Connect to the database, | |

6. If you will have a separate database for the Business Process Choreographer's messaging engine message store (not Derby embedded nor file store), plan the user ID that will be used to access the database.

*Table 9. Planning user ID for the preconfigured BPEME messaging engine database*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| Bus data source user name<br><br>If you use the bpeconfig.jacl script, this is the -medbUser parameter. | Configuring and runtime | This user name is used to connect to the data source for the BPEME messaging bus, and to create the necessary tables and index. | This user ID must be authorized to perform the following actions on the BPEME database: Connect, delete tables, insert tables, select tables and views, and update tables. | |

7. Plan user IDs for the Java Message Service (JMS) provider.

*Table 10. Planning user IDs for the JMS provider*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| JMS authentication user | Runtime | The authentication alias for the system integration bus. You must specify it when configuring Business Process Choreographer. | It must be a valid user name. It is automatically added to the Bus Connector role for the Business Process Choreographer bus. | |
| JMS API authentication user | Runtime | Any Business Flow Manager JMS API requests will be processed on using this user ID. | These must be valid IDs.<br><br>If you use the bpeconfig.jacl script, these are the parameters: | |
| Escalation user authentication user | Runtime | Any Human Task Manager JMS API requests will be processed on using this user ID. | • -jmsBFMRunAsUser<br>• -jmsHTMRunAsUser | |

8. Plan which groups or user IDs, the J2EE roles for the Business Flow Manager and Human Task Manager will be mapped onto.

*Table 11. Planning the security roles for the Business Flow Manager and Human Task Manager*

| User ID or role | When the user ID is used | What the user ID is used for | Planned list of user IDs, groups, or both |
|---|---|---|---|
| Administrator user | Runtime | The system administrator and monitor security roles for both the Business Flow Manager and Human Task Manager are each mapped to a list of user IDs, groups, or both. The values defined here create the mapping that gives users in this role the access rights that they need.<br><br>If you use the bpeconfig.jacl script, these users and groups correspond to the following parameters:<br>• -adminBFMUsers and -adminHTMUsers<br>• -adminBFMGroups and -adminHTMGroups<br>• -monitorBFMUsers and -monitorHTMUsers<br>• -monitorBFMGroups and -monitorHTMGroups | |
| Administrator group | Runtime | | |
| Monitor user | Runtime | | |
| Monitor group | Runtime | | |

9. If you want human task escalations to send notification e-mails for specific business events, and your Simple Mail Transfer Protocol (SMTP) server requires authentication, decide which user ID will be used to connect to the e-mail server.

*Table 12. Planning the user ID for the e-mail server*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| Mail transport user | Runtime | The Human Task Manager uses this user ID to authenticate against the configured mail server to send escalation e-mails.<br><br>If you use the bpeconfig.jacl script, this is the -mailUser parameter. The password is the -mailPwd parameter. | Send e-mails. | |

10. If you will use people assignment for human tasks, and you will use a Lightweight Directory Access Protocol (LDAP) people directory provider that uses simple authentication, plan the user ID that will be used to log on to the LDAP server.

*Table 13. Planning the user ID for the LDAP server*

| User ID or role | When the user ID is used | What the user ID is used for | Which rights the user ID must have | Planned user ID |
|---|---|---|---|---|
| LDAP plug-in property: Authentication Alias | Runtime | When configuring a Lightweight Directory Access Protocol (LDAP) people directory provider that uses simple authentication to connect to LDAP, for example, `mycomputer/My LDAP Alias`. You specify this user ID when customizing the properties for the LDAP plug-in. | If the LDAP server uses simple authentication, this user ID must be able to connect to the LDAP server. If the LDAP server uses anonymous authentication this user ID is not required. | |

11. Create the user IDs that you have planned with the necessary authorizations. If you do not have the authority to create them all yourself, submit a request to the appropriate administrators, and enter the names of the user IDs that they create for you in this table.

**Results**

You know which user IDs will be required when configuring Business Process Choreographer.

# Planning the databases for Business Process Choreographer

Plan the databases for Business Process Choreographer. Depending on your setup, you might need to plan to create up to three databases, or none.

**About this task**

Business Process Choreographer can share a database with other process server components. The BPEDB database is used by the Business Flow Manager and the Human Task Manager. For a production system plan to have a dedicated database for each deployment target where Business Process Choreographer is configured. If you are using the Business Process Choreographer Observer, it can use the same BPEDB database, but using an additional database, OBSRVDB, gives better performance. The Business Process Choreographer messaging engines can either share the database used by the SCA messaging engines, or have their own BPEMEDB database. For more information about which databases are supported for your selected configuration path, see Table 4 on page 93.

**Procedure**

1. For a non-production system where simplicity of setup is more important than performance, your options depend on the configuration path that you have chosen:
   - If you will use the Installer or Profile Management Tool to create the "Basic sample" Business Process Choreographer configuration, a separate Derby BPEDB database is created, which is also used by the Business Process Choreographer Observer. For the Business Process Choreographer messaging engine, the default is to have a separate Derby database (BPEME), but if you use the "Sample with organization" configuration, you can also select to use a File Store or to share the WPRCSDB database.
   - If you will use the Installer or Profile Management Tool to create a deployment environment that includes a Business Process Choreographer configuration, Business Process Choreographer, Business Process choreographer Observer, and the Business Process Choreographer messaging engine will all use the WPRCSDB database. Therefore, you do not need to do any database planning for Business Process Choreographer.

2. For a production system:
   a. If performance is important, plan to use a separate database for Business Process Choreographer, as described in "Planning the BPEDB database" on page 108, otherwise, plan to use the WPRCSDB common database.
   b. If you will use the Business Process Choreographer Observer:
      - If you want to minimize the impact that its queries have on the performance of your business processes, plan to use a separate database as described in "Planning the OBSRVRDB database" on page 113.
      - Otherwise, plan to configure it to use the BPEDB database.
   c. If you want to improve performance by using a separate database for the Business Process Choreographer messaging engine, perform "Planning the messaging engine database" on page 117, otherwise plan to use the default database that is used by the Service Component Architecture (SCA).

**Results**

You have planned all the database for your Business Process Choreographer configuration.

## Planning the BPEDB database

Plan the database for Business Process Choreographer.

**About this task**

Business Process Choreographer requires a database. SQL scripts are provided for all supported database systems to create and administer the database schema. Once a database is in place, JDBC access to the database has to be configured for Business Process Choreographer. Depending on the database system, your topology, the purpose of the installation, and the administrative tool you choose to use, some or all of the tasks to create the database and to configure JDBC access can be automated. For a productions system, Business Process Choreographer should have its own database, but if performance is not important, you can also configure Business Process Choreographer to share a database with other WebSphere Process Server components.

**Procedure**

1. Make sure that your choice of BPEDB database and configuration path are compatible: The following databases are supported:
   - DB2 UDB for Linux, UNIX, and Windows
   - DB2 for iSeries™
   - DB2 for z/OS®
   - Derby
   - Informix Dynamic Server
   - Microsoft® SQL Server
   - Oracle

   If you have already decided how you are going to configure Business Process Choreographer, your choice of configuration path has implications on how you can create the database. If you have not yet decided which configuration path to use to configure Business Process Choreographer, identifying your database requirements will help eliminate the configuration paths that do not support your needs. For details about which databases are supported by each configuration path, see Table 4 on page 93.

2. If you want a **simple** database setup with minimum planning and the following characteristics:
   - It does not offer the performance, scalability, nor security that is normally required for a production system.
   - All database objects are created in a single table space, for example the default tablespace.
   - The database server is local to the WebSphere Process Server machine.
   - The user ID used to access the database also has administration rights.

   The options that you need to plan depends on your choice of configuration path:
   - If you use the **Installer** or **Profile Management Tool** to get a sample Business Process Choreographer configuration, a separate Derby BPEDB database is created for Business Process Choreographer, which requires no further planning.

- If you use the administrative console's **Deployment Environment wizard** to configure Business Process Choreographer, plan to use a copy of the provided SQL script to create the BPEDB database, which will create the default schema in a single table space.
- The **bpeconfig.jacl** tool can configure Business Process Choreographer and the JDBC access for any database.
  - If you will run the bpeconfig.jacl script in interactive mode, you can select to create the tables in an existing database.
  - If you have a user ID with the authority to create the database objects, you can use the -createDB yes option, which causes the bpeconfig.jacl script to generate and run an SQL file to create the database objects in the default table space. In this case also plan to stop the server and use the `-conntype NONE` option for the wsadmin utility. If you are using an Oracle or DB2 for z/OS database, the database instance must already exist. For other database types, bepconfig.jacl will try to create the database instance. If any error occurs while creating the database or objects, you can use the generated SQL scripts as if you used the -createDB no option.
  - If you do not have a user ID with the authority to create the database objects, you must use the -createDB no option, which causes the bpeconfig.jacl script to generate an SQL file to create the database objects in the default table space, but it does not run the script. In this case, plan to ask your database administrator to customize and run the script for you.

  For more information about the tool and other database parameters, see "bpeconfig.jacl script file" on page 163.
- Using the administrative console's **Business Process Choreographer configuration page**:
  - To have the Business Process Choreographer database objects created in the common database, plan to use the default database instance as the target for the Business Process Choreographer data source.
  - To reuse an existing database, plan to specify the existing database instance as the target for the Business Process Choreographer data source.
  - If you select the `Create tables` option, Business Process Choreographer will create the database objects that it needs in the database, the first time that it uses the database. This option cannot be used for a DB2 on z/OS database.
  - To create the database using scripts, plan not to use the `Create tables` option.
3. If you want a **high performance** database setup for Business Process Choreographer with the following characteristics:
   - The database is only used by Business Process Choreographer.
   - Ideally, the database server is on a dedicated machine, however it can also be local to the WebSphere Process Server machine.
   - You can customize the allocation of tables space to disks for better performance.
   - You can use a different user ID to access the database to the one used to administer it.

   You must plan all the following steps:

4. If you have not already planned the user IDs for the database, perform Table 7 on page 104.

5. Plan the allocation of disks and table spaces. Ideally, the database host should have a fast storage subsystem, such as network-attached storage or a storage area network. For a production system, take into account the results of your experiences during development and system testing. The size of your database depends on many factors. Processes that run as microflows use very little space, yet each process template can require tens or hundreds of kilobytes.

   If you will use individual disks, and your database system supports allocating database tables to different disks, plan how many disks you will use and how you will allocate them. Hardware-assisted disk arrays usually offer better performance than single disks.If you use one of the following:
   - DB2
   - DB2zOSV7
   - DB2zOSV8
   - Informix (table spaces are known as named dbspaces)
   - Oracle

   Plan where to locate the following BPEDB database table spaces:
   - AUDITLOG
   - COMP
   - INDEXTS
   - INSTANCE
   - LOBTS
   - STAFFQRY
   - TEMPLATE
   - WORKITEM
   - SCHEDTS

   They can be all on one high-performance RAID-array, but each table space should be in a different file to allow parallel access. Keep in mind that for a given number of disks, using a RAID configuration will have better performance than allocating table spaces to separate disks. For example, for a DB2 database that is running on a dedicated server with N processors, consider using the following guidelines:
   - For the table spaces, use a RAID-1 array with 2*N primary disks, 2*N mirror disks, and a stripe size of 256 kb.
   - For the database transaction log, use a RAID-1 array with 2*N primary disks, 2*N mirror disks, and a stripe size of 64 kb.

   If you are using a DB2 database and will use 15 disk drives on a RAID controller, consider using the following allocations:
   - One disk for the operating system and paging (known as the page file on Windows, paging space on AIX® and HP-UX, and swap space on Solaris).
   - Use eight disks in a RAID-1 configuration (four primary disks and four mirrors) as one logical disk for the database control files and table spaces, using a stripe size of 256 kB.
   - Use six disks in a RAID-1 configuration (three primary disks and three mirrors) as one logical disk for the database transaction log, using a stripe size of 64 kB.

   If you are using an Oracle database, consider the following guidelines:

- Stripe and mirror everything (SAME) for all files, across all disks, using a one megabyte stripe width.
- Mirror data for high availability.
- Create a partition (for the tables space) that is on the outside half of the disk drives.
- Subset data by partition, not by disk.
- Use the Automatic Storage Management (ASM) file system.
- Do not separate redo logs from other data files.

6. Plan that you or your database administrator will customize the SQL scripts that create the database objects before running them.
   - If you use the **bpeconfig.jacl** tool to configure Business Process Choreographer, use the -createDB no option. This prevents the tool from running the SQL script that it generates. The generated SQL files are based on the standard SQL files that are provided for your database, but with all configuration parameters that are provided to the bpeconfig.jacl tool pre-filled in the SQL file, which minimizes the customization necessary. For more information about the tool and other database parameters, see "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157.
   - If you use the administrative console's **Business Process Choreographer configuration page** or **Deployment Environment wizard** to configure Business Process Choreographer, plan to clear the `Create tables` option, to make sure that you do not get the default schema. Plan to customize a copy of the standard SQL files that are delivered for your database system.

   If you want to preview the SQL files for your database, so that you can plan what customizations you will make, locate and view the SQL createSchema.sql script for your database, but do not modify it. The SQL files are located in:
   - On Windows platforms: *install_root*\dbscripts\ProcessChoreographer\ *database_type*
   - On Linux, UNIX, i5/OS, and in the UNIX System Services (USS) on z/OS platforms: *install_root*/dbscripts/ProcessChoreographer/*database_type*

   Where *database_type* is one of the following:
   - DB2
   - DB2zOSV7
   - DB2zOSV8
   - Db2iSeries
   - Derby
   - Informix
   - Oracle
   - SQLServer

7. If the database server is remote to the process server, plan to install either a Java Database Connectivity (JDBC) driver or a database client on the process server machine:
   - For a Type 2 JDBC driver: Decide which database client to install, and where to install it.
   - For a Type 4 JDBC driver: Locate the JAR file for the driver, and decide where to install it.

8. If the database server is local to the process server, the JDBC JAR files required to access the database are installed with the database system. Find and note the location of these JAR files.

9. If you use DB2 for z/OS, decide on the subsystem to use. Plan the values that you will substitute for storage group name, database name (not the subsystem name), and schema qualifier in the script files createTablespace.sql and createSchema.sql.

10. Decide which server will 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.

11. Decide the values for the following configuration parameters that you will need to specify for the database:

    • The Java Database Connectivity (JDBC) provider can be type-2 or type-4. For Oracle, decide between using the oci or thin driver.

    • Database instance (for Oracle, the database name, for DB2 on z/OS: the subsystem name).

    • Schema qualifier. The default is to use the connection user ID as the implicit schema qualifier.

    • User name to create the schema.

    • If you are using a type-4 JDBC driver: the name or IP address of the database server.

    • The port number used by the database server. This is only required if you are using a type-4 JDBC driver.

    • The user ID and password for the authentication alias. This is the user ID that the jdbc/BPEDB data source uses to access the database at runtime. These are the -dbUser and -dbPwd parameters for bpeconfig.jacl.

    For more information about these parameters, including default values, refer to their descriptions in "Business Process Choreographer configuration" on page 130 or "bpeconfig.jacl script file" on page 163.

12. Plan to support enough parallel JDBC connections:

    a. Estimate the maximum number of parallel JDBC connection required to the Business Process Choreographer BPEDB database. This will depend on the nature of your business processes and the number of users. A good estimate is the maximum number of clients that can concurrently connect through the Business Process Choreographer API plus the number of concurrent endpoints defined in the BPEInternalActivationSpec and HTMInternalActivationSpec JMS activation specifications, plus a 10% safety margin to allow for overload situations.

    b. Make sure that your database system can support the necessary number of parallel JDBC connections.

    c. Plan suitable settings according to the best-practices for your database system to support the expected number of parallel JDBC connections.

13. For a production system, make plans for the following administration tasks:

    • Tune your database after it is populated with typical production data.

    • Regularly delete completed process instances and task instances from the database.

**Results**

You have planned the database for Business Process Choreographer.

**Related tasks**

"Balancing the hardware resources" on page 594
You can improve the performance of long-running business processes by balancing the hardware resources.

## Planning the OBSRVRDB database

Plan the database for the Business Process Choreographer Observer.

**About this task**

Business Process Choreographer Observer can use the same database, but using an additional database, OBSRVRDB, gives better performance. If you will not reuse the BPEDB database, perform the following:

**Procedure**

1. If you plan to have multiple event collector instances, and they are going to use the same database, plan unique schema names for each event collector. For better performance, plan a database for each event collector.
2. Decide which database system to use for the database:
   - Derby
   - DB2 UDB for Linux, UNIX, and Windows
   - DB2 for iSeries
   - DB2 for z/OS
   - Oracle

   **Restriction:** The Business Process Choreographer Observer does not support using an Informix or SQL Server database.
3. Decide which server will host the database.
4. If you have not already planned the user IDs for the database, perform Table 8 on page 104.
5. If you will not use the bpeconfig.jacl script to configure the observer and event collector to use the BPEDB database, decide how you will create the OBSRVRDB database.

   **Using the menu-driven administration tool, setupEventCollector**
   > You can use this tool to create the database in an interactive mode, with your input validated against the runtime environment. If you use this tool, decide whether you want the tool to create an SQL file, but not run it – use this option if you want to customize the SQL before running it or to give to your database administrator to customize and run. For more information about this tool, see "setupEventCollector tool" on page 261.
   >
   > The tool allows you to create either Java-based user-defined functions (UDFs) or SQL-based UDFs, and to switch between these two options, and also to install and remove the JAR file that is required to support the UDFs. For a non-Derby database, the tool supports creating the database using either Java-based user defined functions (UDFs) or SQL-based UDFs.For a DB2 on z/OS database, the tool supports creating the database using either Java-based UDFs or SQL-based UDFs. For a Derby database, only Java-based UDFs are used to create the database.

   **Running SQL scripts**
   > You might need to use the SQL scripts if you are not allowed to use a tool to access the database. For a non-Derby database, all SQL scripts create the UDFs for the Business Process Choreographer Observer using the SQL implementation, which simplifies the task when configuring using the SQL based UDFs. . For a Derby database, only Java-based UDFs are used to create the database.

**Automatically create tables on first use**

Selecting the **Create tables** option on the administrative console's Business Process Choreographer configuration page is an easy way to get a default database schema. This option is not suitable for high performance systems. For a non-Derby database, SQL-based UDFs are used, which simplifies the configuration task. This option cannot be used for a DB2 on z/OS database. For a Derby database, only Java-based UDFs are used to create the database.

**Note:** If you use a DB2 for z/OS database, and would prefer that the database is created using Java-based UDFs, rather than SQL-based UDFs, then you have no choice but to use the menu-driven administration tool, setupEventCollector. If you use a Derby database, Java-based UDFs will be used because the embedded Derby database does not support SQL UDFs. For more information about UDFs, see "User-defined functions for Business Process Choreographer Observer" on page 244.

6. If you use a DB2 for Linux, UNIX, or Windows database, plan the following:
   - The database name. The default value is BPEDB.
   - The user ID to use to connect to the database. You must also know the password for this user ID.
   - The database schema name, under which, the database objects are created. The default value is the connection user ID.
   - Plan the fully qualified location for the table space OBSVRTS.
   - Decide whether you want to use the SQL-based user-defined function (UDFs) rather than the default, Java-based ones.
   - If you will use the setupEventCollector tool to setup the database, also plan the following:
     - Decide which type of JDBC driver to use:
       - Type 2, connecting using a native database client. This is the default.
       - Type 4, connecting directly via JDBC. In this case, also make sure that you know the following:
         - The hostname or IP address of the database server. The default is localhost.
         - The port number used for the database. The default is 50000.
     - Locate the directory where the DB2 JDBC driver files, db2jcc.jar and db2jcc_license_cu.jar are installed.

7. If you use a DB2 for i5/OS database, plan the following:
   - The database name. If you configure the database in the native i/Series environment, for example, in qshell, use *LOCAL. Otherwise, use *SYSBAS.
   - The user ID to use to connect to the database. You must also know the password for this user ID.
   - The database schema name, under which, the database objects are created. The default value is the connection user ID.
   - Decide whether you want to use the SQL-based user-defined function (UDFs) rather than the default, Java-based ones.
   - If you will use the setupEventCollector tool to setup the database, also plan the following:
     - The hostname of the database server. Typically this is always localhost. The port number is always 446.
     - The directory for the JDBC driver:

- If the database is in a native i/Series environment, for example, in qshell, this is the path where the db2_classes.jar file is located, which is normally /QIBM/ProdData/Java400/ext.
- If the database is remote, this is the path where the jt400.jar file is located.

8. If you use a DB2 for z/OS database, plan the following:
   - Location name (network name) of the subsystem.
   - The storage group name.
   - The database name as known by the subsystem. The default value is OBSRVRDB
   - The user ID to use to connect to the database. You must also know the password for this user ID.
   - The database schema name (SQLID), under which, the database objects are created.
   - Depending on the DB2 version you use, plan in which storage group the table spaces will be created:
     - For DB2 for z/OS Version 7 use:
       - Regular table space for OBSVRTS.
       - Large object (LOB) table space for OS26201, OS26202, OS26203, and OS26204.
     - For DB2 for z/OS Version 8 use:
       - Regular table space for OBSVR01, OBSVR02, OBSVR03, OBSVR04, OBSVR05, OBSVR06, OBSVR07, and OBSVR08.
       - LOB table space for OS26201, OS26202, OS26203, and OS26204.
   - If you want to use the Java-based user-defined function (UDFs) rather than the default SQL ones, decide on the name of the WLM environment that you will use to run the functions in.
   - If you will use the setupEventCollector tool to setup the database, also plan the following:
     - Decide which type of JDBC driver to use:
       - Type 4, connecting directly via JDBC. In this case, also make sure that you know the following:
         - The hostname or IP address of the database server. The default is `localhost`.
         - The port number used for the database. The default is 446.
         - The directory for the JDBC driver JAR files, db2jcc.jar and db2jcc_license_cisuz.jar.
       - Type 2, connecting using a native database client. In this case, also plan what the database alias will be in the local catalog.

9. If you use a Derby database, plan the following:
   - The database name. This must be the fully qualified path on the server's file system. The default value is *install_root*/databases/BPEDB.
   - The database schema name, under which, the database objects are created. The default value is `APP`.
   - If you will use the setupEventCollector tool to setup the database, also plan the following:
     - If you use the Derby Network JDBC driver, plan the user ID to use to connect to the database. You must also know the password for this user ID.

– Decide which type of JDBC driver to use:
  - Embedded JDBC driver. In this case, also plan the directory for the JDBC driver JAR file derby.jar. The default location is *install_root*/derby/lib.
  - Network JDBC driver. In this case, also make sure that you know the following:
    • The directory for the JDBC driver JAR file derbyclient.jar. The default location is *install_root*/derby/lib.
    • If using a Derby Network server, decide on the location of the UDF JAR file bpcodbutil.jar on the Derby network server. The default location is *install_root*/derby/lib.
    • The hostname of the database server. The default is `localhost`.
    • The port number used for the database. The default is `1527`.
10. If you use an Oracle database, plan the following:
    • The SID name. The default value is `BPEDB`.
    • Decide which Oracle user ID to use to connect to the database. It must have the roles CONNECT and RESOURCE. The default user ID is `system`. You must also know the password for this user ID.
    • The database schema name, under which, the database objects are created. The default value is the user ID used to connect to the database.
    • Plan the fully qualified locations for each of the following table spaces:
      – OBSVRIDX
      – OBSVRLOB
      – OBSVRTS
    • Decide whether you want to use the SQL-based user-defined function (UDFs) rather than the default, Java-based ones.
    • If you will use the setupEventCollector tool to setup the database, also plan the following:
      – The location of the JDBC driver file ojdbc14.jar.
      – The hostname of the database server. The default is `localhost`.
      – The port number used for the database. The default is `1521`.
11. If you use the **bpeconfig.jacl** tool in batch mode with the -createEventCollector yes option, plan one of the following:
    • The -createDB yes option causes the tool to run the SQL script that bpeconfig.jacl generates. You can use the -dbSchema parameter to specify a schema qualifier for the BPEDB database, but the Business Process Choreographer Observer will use the same schema in the same database. That is why using the -createDB yes option is not suitable for a high-performance system.
    • The -createDB no option prevents the tool from running the SQL script that it generates. The generated SQL files are based on the standard SQL files that are provided for your database, but with all configuration parameters that are provided to the bpeconfig.jacl tool pre-filled in the SQL file, which minimizes the customization necessary. Plan that you or your database administrator will customize the generated SQL script that creates the database objects before running it. For more information about the tool and other database parameters, see "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157.

**Restriction:** You cannot configure the Business Process Choreographer Observer using the bpeconfig.jacl script in interactive mode.

12. If you will use administrative console's **Business Process Choreographer event collector page** to create the database tables, plan one of the following:

    - For all database types except for DB2 on z/OS, you can use the Create tables option to causes the tool to creates the default schema in the specified database the first time that Business Process Choreographer accesses the database.

    - If you want to run an SQL script to prepare the database tables, do not use the Create tables option. Plan that you or your database administrator will customize a copy of the SQL script that creates the database objects before running it. This option is most suitable for a production system.

13. If you want to preview the SQL script for your database, so that you can plan what customizations you will make: Locate and view the createSchema_Observer.sql file for your database, but do not modify it. The SQL files are located in:

    - On Windows platforms: *install_root*\dbscripts\ProcessChoreographer\ *database_type*
    - On Linux, UNIX, i5/OS, and in the UNIX System Services (USS) on z/OS platforms: *install_root*/dbscripts/ProcessChoreographer/*database_type*

    Where *database_type* is one of the following:
    - DB2
    - DB2zOSV7
    - DB2zOSV8
    - Db2iSeries
    - Derby
    - Oracle

    **Note:** If you use the bpeconfig.jacl tool to configure Business Process Choreographer, plan to use the SQL script that the tool generates, which does not need to be edited to substitute values for placeholders for configuration parameters. The generated scripts are only available after running the tool, but they are based on the scripts in the locations listed above. You will still have to edit the generated script file if you want to customize the table space allocations.

**Results**

You have planned the database for the Business Process Choreographer Observer.

## Planning the messaging engine database

You can improve performance by using a separate database for the messaging engine for the Business Process Choreographer bus.

**About this task**

You can use one messaging database for all buses that are created by WebSphere Process Server, that is, for the Service Component Architecture (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 performance is important, plan to use a dedicated database for the Business Process Choreographer messaging engine, rather than using the default MEDB that is used for the SCA bus and applications.

**Procedure**

1. If you use the **Installer** or **Profile Management Tool** to get one of the sample Business Process Choreographer configurations, decide whether the Business Process Choreographer messaging engine will use Derby embedded, File Store, or the WPRCSDB database.

2. The Java Database Connectivity (JDBC) provider. Note that the File Store and embedded Derby database are not available in a network deployment environment.

3. If you want to use WebSphere MQ, you must use the bpeconfig.jacl configuration script to configure Business Process Choreographer. Using WebSphere MQ is deprecated.

4. If you use the bpeconfig.jacl configuration script to configure Business Process Choreographer on a stand-alone server, decide what you will use for the message store:

   • A separate schema in the messaging engine database for SCA.

   • A separate Derby database.

   • Any user-defined data source that has been created on the deployment target before running the bpeconfig.jacl script.

   • A File Store.

5. If you use the administrative console's Business Process Choreographer configuration page, plan the following configuration parameters. For more information about these parameters, see the section about the "Bus" on page 136 on the Business Process Choreographer configuration page.

   • Local or remote bus member location.

   • The name of the database. The default is BPEME.

   • The schema name. The default is MEDBPM00.

6. If you are using file store or the embedded Derby JDBC provider, the message stores will be created automatically.

7. If you are not using file store or the embedded Derby JDBC provider, plan the following configuration parameters.

   a. Plan that the database will already exist before Business Process Choreographer is started.

   b. The host name or IP address of the database server, and the port number that it uses.

   c. The user name used to connect to the database and to create the schema. This is the user ID that you planned in Table 9 on page 105.

**Results**

You have planned the database for the Business Process Choreographer messaging engine.

## Planning for the Business Flow Manager and Human Task Manager

The core of a Business Process Choreographer configuration consists of the Business Flow Manager and the Human Task Manager. You must plan their configuration parameters.

**Procedure**

1. Make sure you know the Java Message Service (JMS) provider user ID that will be used as the run-as user ID for the Business Flow Manager message driven bean. In the administrative console, and in Table 10 on page 105, it is known as the **JMS API Authentication User**.

2. Make sure you know the Java Message Service (JMS) provider user ID that will be used as the run-as user ID for the Human Task Manager message driven bean. In the administrative console, and in Table 10 on page 105, it is known as the **Escalation User Authentication User**.

3. Make sure you know which groups or user IDs the security roles for administrator and monitor will map onto. For details, see Table 11 on page 106.

4. If you want the Human Task Manager to send e-mail notifications of escalation events , identify the host name or IP address where the Simple Mail Transfer Protocol (SMTP) e-mail server is located. Plan what the sender address should be for email notifications. If the e-mail service requires authentication, make sure you know the user ID and password to use to connect to the service.

5. Decide on the context root for the Web service binding of the API.
   - When configured on a server:
     – The default for the Business Flow Manager is /BFMIF_*nodeName_serverName*.
     – The default for the Human Task Manager is /HTMIF_*nodeName_serverName*
   - When configured on a cluster:
     – The default for the Business Flow Manager is /BFMIF_*clusterName*
     – The default for the Human Task Manager is /HTMIF_*clusterName*

6. Decide whether you want to initially enable audit logging for the Business Flow Manager, or Human Task Manager, or both.

7. If you are going to use the Business Process Choreographer Observer, decide whether you want the Business Flow Manager to be initially configured to generate Common Event Infrastructure logging events.

**Results**

You have planned all the initial configuration parameters for the Business Flow Manager and Human Task Manager. You can change any of these setting anytime later using the administrative console.

# Planning for the people directory provider

Plan the people directory provider, people substitution, virtual member manager, and Lightweight Directory Access Protocol (LDAP) settings for Business Process Choreographer.

**Procedure**

1. If you are going to use human tasks, decide which people directory providers you will use:

   **Virtual member manager (VMM) people directory provider**
   The VMM people directory provider is ready to-use federated repositories (also known as virtual member manager) as is preconfigured for WebSphere security – using a file repository. If you want to use a different user repository with federated repositories, you will need to reconfigure federated repositories. The VMM people directory provider supports all Business Process Choreographer people assignment features including substitution. It relies on the features

provided by federated repositories, such as support for different repository types, such as LDAP, database, file based, and property extension repository.

To use the VMM people directory provider requires that you have configured a federated repository for WebSphere Application Server security. You can associate a federated repository with other repositories, based on LDAP or a database. For more information about this, see Managing the realm in a federated repository configuration. For more information about using a federated repository, see IBM WebSphere Developer Technical Journal.

**Lightweight Directory Access Protocol (LDAP) people directory provider**
This people directory provider must be configured before you can use it. Perform the planning in step 2.

**System people directory provider**
This people directory provider can be used without configuring it. Do not use this provider for a production system, it is only intended for application development testing.

**User registry people directory provider**
This people directory provider can be used without configuring it.

2. If you are going to use the Lightweight Directory Access Protocol (LDAP), plan the following.

a. You might need to customize your own version of the LDAPTransformation.xsl file. For the location of that file and a list of properties that you might need to customize, see "Configuring the LDAP people directory provider" on page 194.

b. Plan the following LDAP custom properties:

| LDAP plug-in property | Required or optional | Description |
|---|---|---|
| AuthenticationAlias | Optional | The authentication alias used to connect to LDAP, for example, `mycomputer/My LDAP Alias`. You must define this alias in the administrative console by clicking **Security** → **Secure administration, applications, and infrastructure** → **Java Authentication and Authorization Service** → **J2C Authentication Data**. If this alias is not set, anonymous logon to the LDAP server is used. |
| AuthenticationType | Optional | If this property is set to `simple`, for simple authentication, then the AuthenticationAlias parameter is required. Otherwise, if it is not set, anonymous authentication is used. |
| BaseDN | Required | The base distinguished name (DN) for all LDAP search operations, for example, `"o=mycompany, c=us"`. To specify the directory root, specify an empty string, ''. |
| Casesentiveness ForObjectclasses | 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, `com.sun.jndi.ldap.LdapCtxFactory` |
| ProviderURL | Required | This Web address must point to the LDAP JNDI directory server and port. The format must be in normal JNDI syntax, for example, `ldap://localhost:389`. For SSL connections, use the LDAP's URL. |
| 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` |

| LDAP plug-in property | Required or optional | Description |
| --- | --- | --- |
| additionalParameter Name1-5 and additionalParameter Value1-5 | Optional | Use these name-value pairs to set up to five arbitrary JNDI properties for the connection to the LDAP server. |

3. If you are going to use the virtual member manager, plan the following.

   a. You might need to customize your own version of the VMMTransformation.xsl file. For the location of that file and a list of properties that you might need to customize, refer to "Configuring the Virtual Member Manager people directory provider" on page 193.

4. If you are going to use people substitution. Plan to use the VMM Property Extension Repository to store the substitution information. This requires a database. The Property Extension Repository and, implicitly, the selected database must be unique and accessible from within the whole WebSphere cell. As the BPEDB database is not necessarily unique within a cell, BPEDB cannot be used. You can use the common database, WPSRCDB, to host the Property Extension Repository, however, for a production environment, it is recommended to use a database that is independent of other WebSphere Process Server databases.

**Results**

You have planned the people directory provider and people assignment options.

## Planning for the Business Process Choreographer Explorer

Plan the configuration parameters for the Business Process Choreographer Explorer.

**About this task**

If you will use the Business Process Choreographer Explorer. you can either plan and configure it at the same time as you configure Business Process Choreographer, or you can do it later.

**Procedure**

1. Decide how many Business Process Choreographer Explorer instances you want to configure. You can easily create the first instance while configuring Business Process Choreographer. Possible reasons for having more than one instance, include:

   • Because each Business Process Choreographer Explorer instance can only manage one Business Process Choreographer configuration, if you have more than one Business Process Choreographer configuration in your environment, it makes sense to set up a Business Process Choreographer Explorer instance for each configuration.

   • If you are going to customize your Business Process Choreographer Explorer, you might want to have two or more different customized versions of the Business Process Choreographer Explorer manage the same Business Process Choreographer configuration.

   If you want extra instances, you will have to create them individually, either using the administrative console, as described in "Business Process Choreographer Explorer settings" on page 138, or using the configuration

script, as described in "Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer" on page 208.

2. For each Business Process Choreographer Explorer instance that you want, plan the following:
   - The context root for the Business Process Choreographer Explorer. It must be unique within the cell. The default is /bpc.
   - The URL for the Business Process Choreographer Explorer that will be inserted in escalation e-mails.

   If you have Business Process Choreographer Explorer managing the same Business Process Choreographer configuration, the Human Task Manager for the Business Process Choreographer configuration can only link to one of the Business Process Choreographer Explorer instances. In this case, decide which You must decide which Business Process Choreographer Explorer URL the Human Task Manager will use.
   - The maximum number of results to be returned for a query - the default is 1000.
   - The deployment target (server or cluster) of the Business Process Choreographer instance that this Business Process Choreographer Explorer will manage.

   For more information about these configuration parameters, refer to "Business Process Choreographer Explorer settings" on page 138.

**Results**

You have planned the configuration options for the Business Process Choreographer Explorer.

# Planning for the Business Process Choreographer Observer

Plan to configure the Business Process Choreographer Observer and event collector.

**About this task**

If you will use the Business Process Choreographer Observer, you can either plan and configure it at the same time as you configure Business Process Choreographer, or you can do it later.

**Procedure**

1. Understand the purpose and relationships between the different Business Process Choreographer Observer topology elements.

   **The observer application.**
   > You can configure multiple instances on a server or cluster. The instances do not need to be on a deployment target where Business Process Choreographer has been configured. Each instance connects via a data source to exactly one database schema.

   **The event collector application.**
   > This application must be deployed on a server or cluster where the Common Event Infrastructure (CEI) server is configured. It does not need to be deployed where Business Process Choreographer has been configured. It receives business process events from CEI, transforms them, and writes them to the OBSRVRDB database.

**The OBSRVRDB database.**

> The event collector and observer communicate by using the same database. For non-production systems, the database can be shared with other components.

Your choices are independent of the topology you have for your Business Process Choreographer setup. For more insight into the possibilities, see "About Business Process Choreographer Observer" on page 127.

2. Identify the purpose of your setup, your machine requirements, and the topology implications.

   **Simple setup**

   > For simpler configuration and administration, but lower performance, deploy the observer and event collector applications on the same deployment target as you have Business Process Choreographer and CEI configured on, and use a local database system.

   **High load production system: network deployment**

   > Use a cell of multiple nodes, with multiple clusters. Install instances of the observer application on any deployment targets in the cell. Install the event collector application on the cluster where you have configured the Common Event Infrastructure (CEI). Use a separate database server.

3. If you have not already planned the database for the observer, perform "Planning the OBSRVRDB database" on page 113.

4. For each event collector instance that you want to configure, plan the following:

   a. Decide where you will install it. You can only install one event collector instance per deployment target, and the deployment target must have CEI configured on it.

   b. Decide how you will configure this event collector instance:

      • Using the administrative console page. For more information about this option, see "Using the administrative console to configure a Business Process Choreographer event collector" on page 248.

      • Using the interactive setupEventCollector tool. For more information about this option, see "Using the setupEventCollector tool to configure a Business Process Choreographer event collector" on page 246.

      • At the same time as creating a Business Process Choreographer configuration, using the bpeconfig.jacl script. The `-createEventCollector` option has the default value `yes`.

        **Note:** Do not use bpeconfig.jacl to configure the Business Process Choreographer Observer for a high-performance system, because bpeconfig.jacl will configure the event collector and observer applications on the same deployment target as the Business Process Choreographer configuration, and configure them to share the BPEDB database. For more information about this option, see "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157. You cannot use bpeconfig.jacl to configure the Business Process Choreographer Observer in interactive mode.

   c. Plan the data source:

      • If security is important:

– If the Business Process Choreographer Observer shares the same physical database as Business Process Choreographer, plan to use a separate data source for the observer database, and plan its JNDI name.

– Plan an authentication alias that will be used for the database.

- If security is unimportant:

– If the Business Process Choreographer Observer shares the same physical database as Business Process Choreographer (BPEDB), you can use the same data source, and therefore the same JNDI name.

– If you use the same data source, you can also use the same authentication alias.

- Plan to create the data source with a cell scope.

d. Plan the configuration parameters required when configuring the event collector:

- The JNDI data source name for the database.

- The schema to be used for the database objects. The default is the user ID that is used to connect to the database.

- The user ID to use to connect to the database. The default is db2admin.

- The password for the user ID.

- If you are using a type 4 JDBC connection, also collect the host name or IP address of the database server and the port number that it uses

- Decide where the event collector will be deployed. The deployment target must have CEI configured on it, so if you have a separate cluster for CEI, plan to deploy the event collector on the same cluster.

- If you will deploy the event collector in a network deployment environment, know on which deployment target the messaging engine for the CEI bus is configured.

- If the CEI bus has security enabled, plan the JMS user ID that will be used to authenticate with the CEI bus.

- Decide whether you want to enable CEI event logging business events when configuring the event collector, or whether you will enable it later using the administrative console or by running a script.

e. Plan the runtime configuration values, which you might need to customize to suit your needs after configuring the event collector:

- BpcEventTransformerEventCount

- BpcEventTransformerMaxWaitTime

- BpcEventTransformerToleranceTime

- ObserverCreateTables

- If the authentication alias user ID will not own the database schema, plan the ObserverSchemaName.

For more information about these values, see "Changing configuration parameters for the Business Process Choreographer Observer" on page 256.

5. For each Business Process Choreographer Observer instance that you configure, plan the following:

- Decide how you will configure this instance:

– Using the administrative console page. For more information about this option, see "Using the administrative console to configure a Business Process Choreographer Observer" on page 253.

- Using the interactive setupObserver tool. For more information about this option, see "Using the setupObserver tool to configure a Business Process Choreographer Observer" on page 252.
- At the same time as creating a Business Process Choreographer configuration, using the bpeconfig.jacl script. The `-createObserver` option has the default value `yes`.

   **Note:** Do not use bpeconfig.jacl to configure the Business Process Choreographer Observer for a high-performance system, because bpeconfig.jacl will configure the event collector and observer applications on the same deployment target as the Business Process Choreographer configuration, and configure them to share the BPEDB database. For more information about this option, see "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157.
- The deployment target for this instance.
- The JNDI name for the data source that is used by the event collector that this instance will monitor.
- If you will use the administrative console to configure this instance, also plan the following:
  - The context root for this instance. This specifies part of the URL that is used in a browser to reach this Business Process Choreographer Observer instance. The default is `/bpcobserver`.
  - The deployment target of the event collector instance whose data will be visualized by this Business Process Choreographer Observer instance.

   For more information about these configuration parameters, refer to "Business Process Choreographer Observer settings" on page 139.
6. If you will use the bpeconfig.jacl script to configure Business Process Choreographer:
- When the script is run in batch mode, the default is that it will also configure the event collector and observer applications, and that they will be configured on the same deployment target as the Business Process Choreographer configuration.
- If you do not want bpeconfig.jacl to configure one or both of the event collector and observer applications, plan to use one or both of the bpeconfig.jacl options `-createEventCollector no` and `-createObserver no`, which prevent bpeconfig.jacl from configuring them.

**Results**

You have planned the configuration options for the Business Process Choreographer Observer and event collector.

# About Business Process Choreographer

Describes the facilities provided by the Business Flow Manager and the Human Task Manager.

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 orchestrate services as well as integrate activities that involve people in business processes. 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 activities that involve people. Both business processes and human tasks are exposed as services in a Service Oriented Architecture (SOA) or Service Component Architecture (SCA); they also support 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 Explorer."
- Business Process Choreographer Observer. This Web applications allows you to observe the states of running business processes. For more information refer to "About Business Process Choreographer Observer" on page 127.

   **Related tasks**

   Planning to configure Business Process Choreographer
   Plan your Business Process Choreographer setup and configuration parameters.

## 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 configure one or more Business Process Choreographer Explorer instances on a server or cluster. It is sufficient to have a WebSphere Process Server installation with a WebSphere Process Server profile, or a WebSphere Process Server client installation – it is not necessary to have Business Process Choreographer configured on the server or cluster. The client installation is only the infrastructure that you need to connect a client to a WebSphere Process Server, but it does not contain the Business Process Choreographer Explorer. However if you have a deployment manager, then the Business Process Choreographer Explorer can be installed on the servers in the WebSphere Process Server client installation as well.

A single Business Process Choreographer Explorer can only connect to one Business Process Choreographer configuration, though it does not have to connect to a local configuration. However, you can configure multiple instances of the Business Process Choreographer Explorer on the same server or cluster, and each instance can connect to different Business Process Choreographer configurations.

Application server

Process application

Process container

Web client application

Web client

http://app_server_host:port/bpc

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 a business process administrator, you are responsible for the smooth operation of deployed business processes. 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.

## About Business Process Choreographer Observer

About Business Process Choreographer Observer.

You can use Business Process Choreographer Observer to create reports on processes that have been completed. You can also use it to view the status of running processes. This describes the architecture and possible configuration paths.

Business Process Choreographer Observer uses the Common Event Infrastructure (CEI) to collect events that are emitted by WebSphere Process Server. You can either use a number of predefined reports or define your own reports to get an overview of the number of processes, activities, or other aggregate data. You can also get information about specific processes or activities.

Business Process Choreographer Observer is based on two J2EE enterprise applications, which are shown in the following figure:
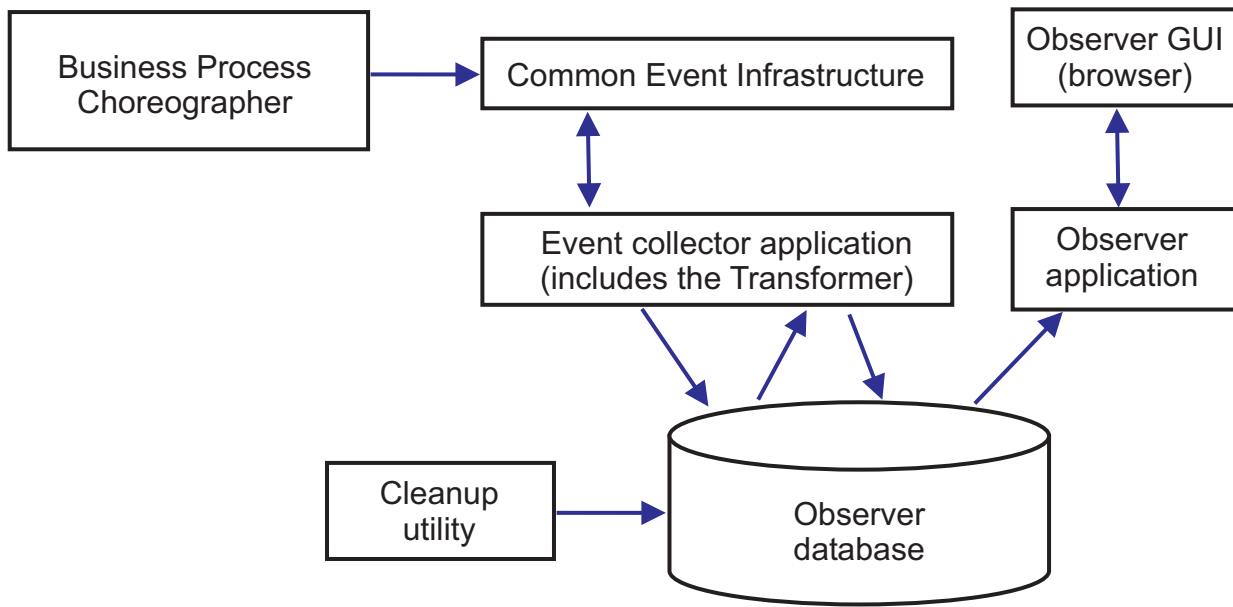
*Figure 1. Business Process Choreographer Observer architecture*

- The event collector application reads event information from the CEI bus and stores it in the event collector table in the Business Process Choreographer Observer database.
- The observer database is a set of database tables that store the event data.
- Periodically the event transformer J2EE enterprise application is triggered, which transforms the raw event data into a format that is suitable for queries from the Business Process Choreographer Observer.
- The observer application generates the reports and performs other actions that the user can initiate using the observer graphical user interface (GUI).
- You can use the GUI to generate your reports. You can also store and retrieve reports that you have defined.
- A cleanup utility can be used to remove records from the observer database, which can help to improve the performance.

## Simple configurations

A simple configuration, where performance is not an important consideration is illustrated in the following figure.
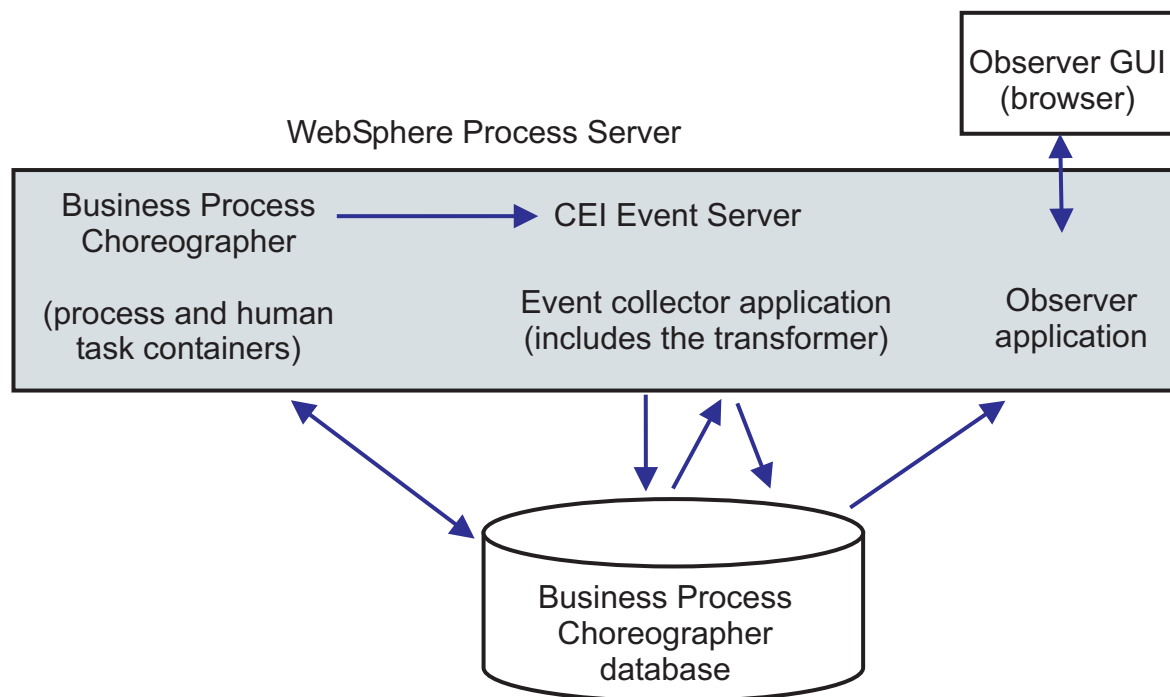
*Figure 2. Business Process Choreographer Observer standalone setup*

Everything is installed on a single machine, and Business Process Choreographer and Business Process Choreographer Observer use the same database.

This kind of simple configuration is created if you create a sample Business Process Choreographer configuration. Also the bpeconfig.jacl tool defaults to configuring the observer and event collector applications on the same deployment target as the Business Process Choreographer configuration. Common Event Infrastructure (CEI) logging will be enabled and the necessary database schema is created in the Business Process Choreographer Derby database BPEDB. This configuration path can be ideal if performance is not an important consideration.

### High-performance configurations

Interactive configuration tools are provided which give you the freedom to exploit the full potential of the Business Process Choreographer Observer architecture. For example, in an ideal configuration for performance in a production system, the three Business Process Choreographer elements run on separate machines, and the Business Process Choreographer Observer has its own database.
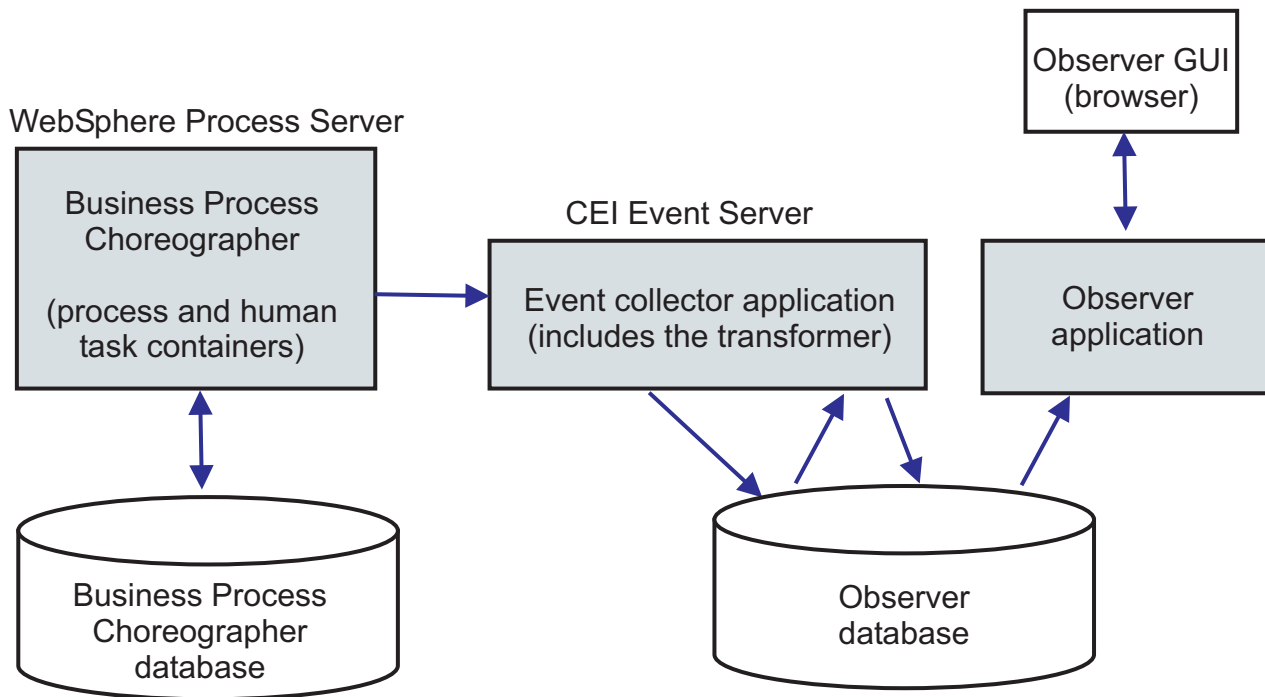
*Figure 3. Business Process Choreographer Observer setup for production performance*

If you want to use a separate database for the Business Process Choreographer Observer, or to add the Business Process Choreographer Observer to an existing Business Process Choreographer configuration, in a clustered setup, or using more sophisticated database options, perform "Configuring the Business Process Choreographer Observer" on page 211.

### In a network deployment environment

The following constraints apply if you want to configure Business Process Choreographer Observer in a network deployment environment.

- CEI must be configured in your cell.
- As illustrated in the previous figure, the Business Process Choreographer event collector must be configured on a deployment target where the CEI Event server is configured. If the CEI Event server is configured on a different cluster than Business Process Choreographer, you must configure the Business Process Choreographer event collector on a deployment target where the CEI Event server is configured. The Business Process Choreographer Observer application does not need to be installed on the same machine as the event collector.

## Business Process Choreographer configuration

Use this panel to install and configure Business Process Choreographer.

To view this administrative console page, click **Servers** ‣ **Clusters** ‣ *cluster_name* or **Servers** ‣ **Application servers** ‣ *server_name*, then under **Containers**, click **Business Process Choreographer Containers**.

This page is divided into sections. For detailed information about the fields in each section, refer to:

- "Data Source" on page 131

## Data Source

In this section you specify the data source for Business Process Choreographer.

## Edit

Click this to edit the data source.

## Test Connection

Tests the connection to the data source.

## Database Instance

The name of the database used by the Business Flow Manager and Human Task Manager.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | WPRCSDB |

## Schema Name

The name of the schema to be used.

You only need to change the schema name if you want to use your own schema instead of the default schema.

| Property | Value |
| --- | --- |
| Data type | String |

## Create Tables

If selected, the tables will be created automatically the first time that the database is accessed. For this option to work, the database must already exist, and the user name provided must have the authority to create tables and indexes in the database.

For a production system, it is not recommended to use this option. If you do not select this option, the tables will not be created automatically, and you must create the tables manually by running scripts.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

## 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 option to create the tables automatically can be used, and when necessary, the database schema will be updated automatically after applying a service or fix pack.

| Property | Value |
| --- | --- |
| Data type | String |

## Password

The password for the data source user ID.

| Property | Value |
| --- | --- |
| Data type | String |

## Server

The address of the database server.

Specify either the host name or the IP address, and the port number.

| Property | Value |
| --- | --- |
| Data type | String |
| Example | localhost:50000 |

## Provider

The JDBC provider for Business Process Choreographer.

| Property | Value |
| --- | --- |
| Data type | Drop-down list |

## Human Task Manager Mail Session

In this section you specify the parameters for escalation e-mails.

## Enable e-mail service

You must enable a mail session if the Human Task Manager will send escalation e-mail notifications.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

## Mail transport host

The hostname or IP address where the Simple Mail Transfer Protocol (SMTP) e-mail service is located.

| Property | Value |
| --- | --- |
| Data type | String |

## Mail transport user

The user ID for the e-mail service.

If the mail server does not require authentication, you can leave this field empty.

| Property | Value |
|---|---|
| Data type | String |

## Mail transport password

Password for the mail transport user ID.

If the mail server does not require authentication, you can leave this field empty.

| Property | Value |
|---|---|
| Data type | String |

## Business Process Choreographer Explorer URL

Specifies the URL used for e-mail links to the Business Process Choreographer Explorer.

This URL is used to provide a link in generated e-mails so that a business administrator who receives an e-mail notification can click on the link to view the related business process or human task in their Web browser.

| Property | Value |
|---|---|
| Data type | String |
| Example | http://www.ibm.com:9080/bpc |

## Security

In this section you specify the mapping of functional roles to user IDs and groups, and the authentication credentials necessary for Business Process Choreographer.

## Administrator User

The administrator security role is mapped to the specified user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

## Administrator Group

The administrator security role is mapped to the specified group.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

## Monitor User

The system monitor security role is mapped to the specified user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

## Monitor Group

The system monitor security role is mapped to the specified group.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

## JMS Authentication User

The authentication alias for the system integration bus.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

## JMS Authentication Password and Confirm Password

The password for the JMS authentication user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

## JMS API Authentication User

The run-as user ID for the business flow manager message driven bean.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

## JMS API Authentication Password and Confirm Password

The password for the JMS API authentication user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

## Escalation User Authentication User

The run-as user ID for the human task manager message driven bean.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

## Escalation User Authentication Password and Confirm Password

The password for the escalation user authentication user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

## State Observers

In this section you can enable audit logging and Common Event Infrastructure (CEI) logging for the Business Flow Manager and Human Task Manager.

## Audit Logging for the Business Flow Manager

When selected, audit logging for the Business Flow Manager is enabled.

| Property | Value |
|---|---|
| Data type | Check box |
| Default | Selected |

## Audit Logging for the Human Task Manager

When selected, audit logging for the Human Task Manager is enabled.

| Property | Value |
|---|---|
| Data type | Check box |
| Default | Selected |

## Common Event Infrastructure Logging for the Business Flow Manager

When selected, Common Event Infrastructure logging for the Business Flow Manager is enabled.

| Property | Value |
|---|---|
| Data type | Check box |
| Default | Selected |

## Common Event Infrastructure Logging for the Human Task Manager

When selected, Common Event Infrastructure logging for the Human Task Manager is enabled.

| Property | Value |
|---|---|
| Data type | Check box |
| Default | Selected |

## SCA Bindings

For the Service Component Architecture (SCA) bindings, you can set the context root for the Web services API.

## Host

This read-only field illustrates the contextual prefix of the hosts for the Business Flow Manager and the Human Task Manager bindings to which the context roots are appended.

## Context Root for the Business Flow Manager

The context root for the Business Flow Manager Web service.

| Property | Value |
| --- | --- |
| Data type | String |
| Default when configured on a server | /BFMIF_*nodeName_serverName* |
| Default when configured on a cluster | /BFMIF_*clusterName* |

## Context Root for the Human Task Manager

The context root for the Human Task Manager Web service.

| Property | Value |
| --- | --- |
| Data type | String |
| Default when configured on a server | /HTMIF_*nodeName_serverName* |
| Default when configured on a cluster | /HTMIF_*clusterName* |

## Relative Path

This read-only field shows the relative paths for the SCA bindings for the Business Flow Manager and the Human Task Manager.

| Property | Value |
| --- | --- |
| Data type | Read-only string |
| Relative path for the Business Flow Manager | /sca/com/ibm/bpe/spi/sca/BFMWS |
| Relative path for the Human Task Manager | /sca/com/ibm/task/spi/sca/HTMWS |

## Bus

If you want to use a different data source for the Business Process Choreographer messaging engine than you configured for the Service Component Architecture (SCA), expand this section to change the settings.

## Use the default configuration

If selected, the current configuration settings of the SCA messaging engine will be used.

If you want to use different settings, clear the check box to enable the other fields in this section.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

## Bus Member Location

Determines whether the messaging engine's data will be stored locally or remotely.

Select between **Local** and **Remote**. If you select **Remote**, the remote destination location selector and the **New** button are enabled.

| Property | Value |
| --- | --- |
| Data type | Radio buttons |
| Default | Local |

## Remote destination location

Specifies the deployment target for the remote messaging engine store.

If the list is empty, or does not contain the location that you want to select, click **New**.

| Property | Value |
| --- | --- |
| Data type | Drop-down list |
| Default | None |

## New

This button opens the Browse deployment target page.

After selecting a deployment target, the target is added to the list of remote destination locations.

## Edit

Click this to edit the data source.

## Test Connection

Tests the connection to the data source.

## Database Instance

The name of the database.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | ${USER_INSTALL_ROOT}\databases\ BPEME |

## Schema Name

The name of the schema to be used.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | MEDBM00 |

## Create Tables

If selected, the tables will be created automatically the first time that the database is accessed. For this option to work, the database must already exist, and the user name provided must have the authority to create tables and indexes in the database.

If not selected, the tables will not be created automatically, and you must create the tables manually by running scripts. For a production system, you might prefer not to have the default tables that are created by this option.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

## 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 option to create the tables automatically can be used, and when necessary, the database schema will be updated automatically after applying a service or fix pack.

| Property | Value |
|---|---|
| Data type | String |

## Password

The password for the data source user ID.

| Property | Value |
|---|---|
| Data type | String |

## Server

The address of the database server.

Specify either the host name or the IP address, and the port number.

| Property | Value |
|---|---|
| Data type | String |
| Example | localhost:50000 |

## Provider

The JDBC provider for the Business Process Choreographer messaging engine.

If you configured SCA to use a file store, this field is set to `File Store` and the fields for the database parameters are not available. When a database provider is selected, the database parameters are available.

| Property | Value |
|---|---|
| Data type | Drop-down list |
| Default | The provider that you configured for SCA. |

# Business Process Choreographer Explorer settings

General properties can be set here.

To view this administrative console page, click **Servers** → **Clusters** → *cluster_name* or **Servers** → **Application servers** → *server_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Explorer Configuration**. To create a new explorer configuration, click **Add**. To view an existing configuration, click an instance name, or select an instance and click **Edit**.

## Context Root

The context root for the Business Process Choreographer Explorer.

This defines part of the URL that browsers must use to reach the Business Process Choreographer Explorer.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | /bpc |

## Explorer search result limit

The maximum number of results to be returned for a query.

| Property | Value |
| --- | --- |
| Data type | Integer |
| Units | Search results |
| Default | 10000 |

## Managed Business Process Choreographer container

The scope of the Business Process Choreographer container.

This specifies the Business Process Choreographer container that this instance connects to. This can be a many-to-one relationship. Each Business Process Choreographer Explorer instance can connect to exactly one Business Process Choreographer container. However, each Business Process Choreographer container can be connected to by zero or more Business Process Choreographer Explorer instances.

| Property | Value |
| --- | --- |
| Data type | Drop-down list of deployment targets |
| Default | The deployment target (server or cluster) where this Business Process Choreographer Explorer instance is configured. |

# Business Process Choreographer Observer settings

Displays the context root and monitoring target for a particular instance of the Business Process Choreographer Observer.

To view this administrative console page, click **Servers** → **Clusters** → *cluster_name* or **Servers** → **Application servers** → *server_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Observer Configuration**. To create a new observer configuration, click **Add**. To view an existing configuration, click an instance name, or select an instance and click **Edit**.

## Context Root

The context root for the Business Process Choreographer Observer.

This defines part of the URL that browsers must use to reach the Business Process Choreographer Observer.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | /bpcobserver |

## Visualize monitoring data from this Business Process Choreographer event collector

Select the event collector instance whose data is to be visualized by this Business Process Choreographer Observer instance.

| Property | Value |
|---|---|
| Data type | Drop-down list |
| Default | The deployment target (server or cluster) where this Business Process Choreographer observer instance is configured. |

# Chapter 4. Configuring Business Process Choreographer

Business Process Choreographer must be configured before you install any enterprise applications that contain business processes or human tasks.

**Before you begin**

You have completed Chapter 3, "Planning to configure Business Process Choreographer," on page 91 and you have chosen a configuration path from Table 4 on page 93.

**About this task**

Depending on your chosen configuration path, perform one of the following:
- For any of the following non-production configuration paths:
  - "Basic sample"
  - "Sample with organization"
  - "Non-production deployment environment"

  Perform "Using the Installer or Profile Management Tool to configure Business Process Choreographer."
- For the "Production deployment environment" configuration path, perform "Using the administrative console's deployment environment wizard to configure Business Process Choreographer" on page 144.
- For the "Flexible custom configuration" configuration path, depending on the tool you want to use, perform one of the following:
  - "Using the administrative console's Business Process Choreographer configuration page" on page 146
  - "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157

**Results**

Business Process Choreographer is configured.

**What to do next**

You can start to customize your setup.

## Using the Installer or Profile Management Tool to configure Business Process Choreographer

There are five easy ways to create a non-production Business Process Choreographer configuration.

**Before you begin**

You have completed "Planning to create a basic sample Business Process Choreographer configuration" on page 95 and you have decided which flavor of non-production system you want, as summarized in Table 4 on page 93.

**Procedure**

1. Depending on the configuration path you selected, perform one of steps 1a - 1c.

   a. If you want the "Basic sample" Business Process Choreographer configuration which does not include a sample organization for people assignment and substitution:

      1) Start the Installer or Profile Management Tool.
         - For the Installer:
            – Make sure that you select the **Typical Installation** option.
            – Make sure that you select the **Stand-alone server** option.
            – Make sure that you enable **Administrative security**.
         - For the Profile Management Tool:
            – Make sure that you create a **WebSphere Process Server** profile.
            – Make sure that you select the **Stand-alone server profile** option.
            – Make sure that you select the **Typical** profile creation option.
            – Make sure that you select **Enable administrative security**.

   b. If you want the "Sample with organization" Business Process Choreographer configuration which includes a sample 15 person organization for people assignment and substitution:

      1) Start the Profile Management Tool.
      2) Make sure that you create a **WebSphere Process Server** profile.
      3) Make sure that you select the **Stand-alone server profile** option.
      4) Make sure that you select the **Advanced** option.
      5) Make sure that you select the **Create server from development template** option.
      6) Make sure that you select **Enable administrative security**.
      7) Make sure that you select **Configure a sample Business Process Choreographer**.

   c. If you want a "Non-production deployment environment" Business Process Choreographer configuration based on a deployment environment pattern:

      1) Start the Installer or Profile Management Tool.
         - For the Installer:
            – Make sure that you select the **Deployment environment installation** option.
            – Make sure that you create a deployment manager.
            – You can base the Business Process Choreographer configuration on any of the following patterns:
               - Remote Messaging and Remote Support
               - Remote Messaging
               - Single Cluster
            – Make sure that you enable **Administrative security**, otherwise you will not get a Business Process Choreographer configuration.
         - For the Profile Management Tool:
            – Make sure that you create a **WebSphere Process Server** profile.
            – Make sure that you select the **Deployment manager profile** option.
            – You can base the Business Process Choreographer configuration on any of the following patterns:

- Remote Messaging and Remote Support
- Remote Messaging
- Single Cluster

– Make sure that you enable **Administrative security**, otherwise you will not get the Business Process Choreographer sample.

   2) Create and federate the custom profiles.

2. Optional: Perform "Verifying that Business Process Choreographer works" on page 267.

3. Optional: If you want to change the JMS authentication user IDs, the run-as user IDs, or the mappings of roles onto users and groups, click **Security** → **Business Integration security** to change the security settings.

4. Optional: Change settings for the Human Task Manager:
   - If you want to change any of the Human Task Manager settings for the escalation e-mails, such as the sender address, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and make your changes.
   - If you want to change the e-mail server address or port number, the user ID or password for the e-mail server, or the URL prefix for the Business Process Choreographer Explorer, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, under **Container Settings**, expand **Business Process Choreographer Container Settings**, click **Business Process Choreographer Containers**, and make your changes.

5. Depending on the type of people directory provider that you use for people assignment, you might need to configure it:
   - The system and user registry people directory providers can be used without configuring them.
   - If you are using Lightweight Directory Access Protocol (LDAP), perform "Configuring the LDAP people directory provider" on page 194.
   - If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 193.

6. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 199.

7. Optional: If you want to use group work items, use the administrative console to enable them. Click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and select **Enable group work items**.

**Results**

Business Process Choreographer is configured.

# Using the administrative console's deployment environment wizard to configure Business Process Choreographer

Using the administrative console's deployment environment wizard, you can create a pattern-based configuration that includes Business Process Choreographer. If the Business Process Choreographer configuration has its own database, the configuration can be suitable for a production system.

**Before you begin**

You have performed "Planning to use the administrative console's deployment environment wizard" on page 97.

**Procedure**

1. Start the deployment environment wizard. In the administrative console click **Servers** → **Deployment Environments** → **New**. As you enter other configuration parameters, make sure that you enter the values that you planned in "Planning to use the administrative console's deployment environment wizard" on page 97:

   a. You can base the Business Process Choreographer configuration on any of the following patterns:
      - Remote Messaging and Remote Support
      - Remote Messaging
      - Single Cluster
      - Custom

   b. On the security page, you can set the user name and password that will be used as the authentication alias for Business Process Choreographer, which is identified as component WBI_BPC.

   c. On the database page, if you want to use separate databases for Business Process Choreographer, the Business Process Choreographer Explorer, or the Business Process Choreographer messaging engine, change the default data sources from the default values to the values that you planned.

   d. On the Business Process Choreographer page, specify the context roots, security parameters, and mail session parameters that you planned for this configuration.

2. If you specified a separate database for Business Process Choreographer, perform "Using SQL scripts to create the database for Business Process Choreographer" on page 181. Otherwise, for a non-Derby database, make sure that the empty database exists so that Business Process Choreographer can create the default schema in the database the first time that it accesses the database.

3. If you specified a separate database for Business Process Choreographer Observer, perform "Preparing a database for the Business Process Choreographer Observer" on page 213. Otherwise, for a non-Derby database, make sure that the empty database exists so that Business Process Choreographer can create the default schema in the database the first time that it accesses the database.

4. If you specified a separate database for Business Process Choreographer messaging engine, make sure that the database exists.

- If you want to use the **Create tables** option to have the messaging engine create the default schema the first time that it uses the database, grant the database user ID the rights to create tables and views in the schema that you planned to use.
- Otherwise, if you will **not** use the **Create tables** option, 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.

5. For each node where Business Process Choreographer is configured, make sure that the environment variables for the JDBC drivers are set. On a cluster, you must perform this for every node that hosts a cluster member.

   a. Click **Environment** → **WebSphere variables**, for **Scope**, select the node where Business Process Choreographer is configured.

   b. Select the environment variable for your JDBC provider:
      - For Derby, you do not need to set any environment variable.
      - For DB2 on Linux, UNIX, Windows, or z/OS, using the CLI driver, select DB2_JDBC_DRIVER_PATH.
      - For DB2 on Linux, UNIX, Windows, or z/OS, using the Universal driver, select DB2UNIVERSAL_JDBC_DRIVER_PATH.
      - For DB2 on i5/OS, using the native driver, select OS400_NATIVE_JDBC_DRIVER_PATH.
      - For DB2 on i5/OS, using the toolbox driver, select OS400_TOOLBOX_JDBC_DRIVER_PATH.
      - For Oracle, select ORACLE_JDBC_DRIVER_PATH.
      - For Informix, select INFORMIX_JDBC_DRIVER_PATH.
      - For SQL Server using the WebSphere embedded ConnectJDBC driver you do not need to set any environment variable.
      - For SQL Server using the DataDirect ConnectJDBC type-4 driver, select CONNECTJDBC_JDBC_DRIVER_PATH.

   c. Set the environment variable to point to the location of the JDBC driver's JAR file, or files.

6. Activate Business Process Choreographer: Perform "Activating Business Process Choreographer" on page 266.

7. Optional: Verify that the basic Business Process Choreographer configuration works: Perform "Verifying that Business Process Choreographer works" on page 267.

8. Optional: Change settings for the Human Task Manager:
   - If you want to change any of the Human Task Manager settings for the escalation e-mails, such as the sender address, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and make your changes.
   - If you want to change the e-mail server address or port number, the user ID or password for the e-mail server, or the URL prefix for the Business Process Choreographer Explorer, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, under **Container Settings**, expand **Business Process Choreographer Container Settings**, click **Business Process Choreographer Containers**, and make your changes.

9. Depending on the type of people directory provider that you use for people assignment, you might need to configure it:
   - The system and user registry people directory providers can be used without configuring them.
   - If you are using Lightweight Directory Access Protocol (LDAP), perform "Configuring the LDAP people directory provider" on page 194.
   - If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 193.
10. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 199.
11. Optional: If you want to use group work items, use the administrative console to enable them. Click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and select **Enable group work items**.
12. Optional: Perform "Verifying that Business Process Choreographer works" on page 267.

**Results**

Business Process Choreographer is configured for the deployment environment that you selected.

# Using the administrative console's Business Process Choreographer configuration page

Describes how to use the administrative console's Business Process Choreographer configuration page to create a configuration on a given server or cluster.

**About this task**

You must configure the necessary resources and install the Business Process Choreographer applications before you can run applications that contain business processes or human tasks.

**Procedure**

1. If you selected the Business Process Choreographer sample configuration option when you created a default profile, the Business Flow Manager, Human Task Manager, Business Process Choreographer Explorer, and Business Process Choreographer Observer are already configured.

   You can check if they are configured, by looking in the administrative console for enterprise applications with names that start with:
   - BPCObserver
   - BPCECollector
   - BPEContainer
   - BPCExplorer
   - TaskContainer

   The sample configuration uses a Derby database, and is not suitable for a production system. Because you can only have one Business Process Choreographer configuration on a deployment target, you must remove the

sample configuration, as described in Chapter 5, "Removing the Business Process Choreographer configuration," on page 271 before you can continue configuring Business Process Choreographer.

2. If you have a network deployment environment, make sure that the Service Component Architecture (SCA) is configured:

   a. If you want to configure Business Process Choreographer on a server, click **Servers** → **Application servers** → *serverName*, then in the **Business Integration** section, click **Service Component Architecture**.

   b. If you want to configure Business Process Choreographer on a cluster, click **Servers** → **Clusters** → *clusterName*, then in the **Business Integration** section, click **Service Component Architecture**.

   c. If it is not enabled, select **Support the Service Component Architecture components**.

3. Create the database for the data store for the Business Process Choreographer messaging engine:

   • If you want to use the **Create tables** option on the Business Process Choreographer configuration page, to have the messaging engine create the default schema the first time that it uses the database, perform the following:

      a. If the database does not already exist, create it.

      b. Grant the database user ID the rights to create tables and views in the schema that you planned to use.

   • Otherwise, if you will **not** use the **Create tables** option, 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.

4. If you are using the WebSphere MQ Java Message Service (JMS) provider, you must create the queue manager and queues: Perform "Creating the queue manager and queues for Business Process Choreographer" on page 202. If you are using the default messaging JMS provider do not perform this step.

5. Creating the database for Business Process Choreographer:

   • If you want to use the **Create tables** option on the Business Process Choreographer configuration page, to have Business Process Choreographer create the default schema the first time that it uses the database, with all tables created in the user's default table space, perform the following:

      a. If the database does not already exist, create it.

      b. Grant the database user ID the rights to create tables and views in the schema that you planned to use.

   • Otherwise, if you will **not** use the **Create tables** option, perform "Using SQL scripts to create the database for Business Process Choreographer" on page 181.

6. For each node where Business Process Choreographer is configured, make sure that the environment variables for the JDBC drivers are set. On a cluster, you must perform this for every node that hosts a cluster member.

   a. Click **Environment** → **WebSphere variables**, for **Scope**, select the node where Business Process Choreographer is configured.

   b. Select the environment variable for your JDBC provider:

      • For Derby, you do not need to set any environment variable.

      • For DB2 on Linux, UNIX, Windows, or z/OS, using the CLI driver, select DB2_JDBC_DRIVER_PATH.

- For DB2 on Linux, UNIX, Windows, or z/OS, using the Universal driver, select DB2UNIVERSAL_JDBC_DRIVER_PATH.
- For DB2 on i5/OS, using the native driver, select OS400_NATIVE_JDBC_DRIVER_PATH.
- For DB2 on i5/OS, using the toolbox driver, select OS400_TOOLBOX_JDBC_DRIVER_PATH.
- For Oracle, select ORACLE_JDBC_DRIVER_PATH.
- For Informix, select INFORMIX_JDBC_DRIVER_PATH.
- For SQL Server using the WebSphere embedded ConnectJDBC driver you do not need to set any environment variable.
- For SQL Server using the DataDirect ConnectJDBC type-4 driver, select CONNECTJDBC_JDBC_DRIVER_PATH.

  c. Set the environment variable to point to the location of the JDBC driver's JAR file, or files.

7. In the administrative console, select the server or cluster where you want to configure Business Process Choreographer. Click one of the following:
   - **Servers** → **Application Servers** → *serverName*
   - **Servers** → **Clusters** → *clusterName*

   Where *serverName* or *clusterName* is the name of the server or cluster.

8. Go to the Business Process Choreographer configuration page: In the **Container Settings** section, expand **Business Process Choreographer Container Settings** and click **Business Process Choreographer Containers**.

9. Verify that Business Process Choreographer is not configured. There should be a message indicating that the Business Process Choreographer containers (Business Flow Manager and Human Task Manager) are not currently installed.

   If the Business Flow Manager and Human Task Manager are already installed, perform Chapter 5, "Removing the Business Process Choreographer configuration," on page 271 before continuing with the next step.

10. Enter the values and select the options that you planned for the Business Process Choreographer configuration on this server or cluster. For more information see "Business Process Choreographer configuration" on page 130, which is divided into the following sections:
    a. "Data Source" on page 131
    b. "Human Task Manager Mail Session" on page 132
    c. "Security" on page 133
    d. "State Observers" on page 135
    e. "SCA Bindings" on page 135
    f. "Bus" on page 136

11. Click **Apply**. Information is displayed reporting the progress deploying and configuring Business Process Choreographer.

12. If the installation succeeded, click **Save Master Configuration**, then click **Save**. Otherwise, discard the changes, check the administrative console and the SystemOut.log file on the Deployment Manager or server for any error messages that can help you correct the problem, then try again.

13. Activate Business Process Choreographer: Perform "Activating Business Process Choreographer" on page 266.

14. Optional: Verify that the basic Business Process Choreographer configuration works: Perform "Verifying that Business Process Choreographer works" on page 267.

15. Optional: Change settings for the Human Task Manager:

- If you want to change any of the Human Task Manager settings for the escalation e-mails, such as the sender address, click **Servers** ⟶ **Application servers** ⟶ *server_name* or **Servers** ⟶ **Clusters** ⟶ *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and make your changes.
- If you want to change the e-mail server address or port number, the user ID or password for the e-mail server, or the URL prefix for the Business Process Choreographer Explorer, click **Servers** ⟶ **Application servers** ⟶ *server_name* or **Servers** ⟶ **Clusters** ⟶ *cluster_name* if Business Process choreographer is configured on a cluster, under **Container Settings**, expand **Business Process Choreographer Container Settings**, click **Business Process Choreographer Containers**, and make your changes.

16. Depending on the type of people directory provider that you use for people assignment, you might need to configure it:
    - The system and user registry people directory providers can be used without configuring them.
    - If you are using Lightweight Directory Access Protocol (LDAP), perform "Configuring the LDAP people directory provider" on page 194.
    - If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 193.

17. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 199.

18. Optional: If you want to use group work items, use the administrative console to enable them. Click **Servers** ⟶ **Application servers** ⟶ *server_name* or **Servers** ⟶ **Clusters** ⟶ *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and select **Enable group work items**.

19. Optional: If you have not yet installed and configured Business Process Choreographer Explorer, you can configure it now. Perform "Configuring Business Process Choreographer Explorer" on page 207.

20. Optional: If you have not yet installed and configured Business Process Choreographer Observer, you can configure it now. Perform, "Configuring the Business Process Choreographer Observer" on page 211.

**Results**

Business Process Choreographer is configured.

## Business Process Choreographer configuration

Use this panel to install and configure Business Process Choreographer.

To view this administrative console page, click **Servers** ⟶ **Clusters** ⟶ *cluster_name* or **Servers** ⟶ **Application servers** ⟶ *server_name*, then under **Containers**, click **Business Process Choreographer Containers**.

This page is divided into sections. For detailed information about the fields in each section, refer to:
- "Data Source" on page 131
- "Human Task Manager Mail Session" on page 132

- "Security" on page 133
- "State Observers" on page 135
- "SCA Bindings" on page 135
- "Bus" on page 136

## Data Source

In this section you specify the data source for Business Process Choreographer.

### Edit

Click this to edit the data source.

### Test Connection

Tests the connection to the data source.

### Database Instance

The name of the database used by the Business Flow Manager and Human Task Manager.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | WPRCSDB |

### Schema Name

The name of the schema to be used.

You only need to change the schema name if you want to use your own schema instead of the default schema.

| Property | Value |
| --- | --- |
| Data type | String |

### Create Tables

If selected, the tables will be created automatically the first time that the database is accessed. For this option to work, the database must already exist, and the user name provided must have the authority to create tables and indexes in the database.

For a production system, it is not recommended to use this option. If you do not select this option, the tables will not be created automatically, and you must create the tables manually by running scripts.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

### 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 option to create the tables automatically can be used, and when necessary, the database schema will be updated automatically after applying a service or fix pack.

| Property | Value |
| --- | --- |
| Data type | String |

### Password
The password for the data source user ID.

| Property | Value |
| --- | --- |
| Data type | String |

### Server
The address of the database server.

Specify either the host name or the IP address, and the port number.

| Property | Value |
| --- | --- |
| Data type | String |
| Example | localhost:50000 |

### Provider
The JDBC provider for Business Process Choreographer.

| Property | Value |
| --- | --- |
| Data type | Drop-down list |

### Human Task Manager Mail Session
In this section you specify the parameters for escalation e-mails.

### Enable e-mail service
You must enable a mail session if the Human Task Manager will send escalation e-mail notifications.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

### Mail transport host
The hostname or IP address where the Simple Mail Transfer Protocol (SMTP) e-mail service is located.

| Property | Value |
| --- | --- |
| Data type | String |

### Mail transport user
The user ID for the e-mail service.

If the mail server does not require authentication, you can leave this field empty.

| Property | Value |
| --- | --- |
| Data type | String |

### Mail transport password
Password for the mail transport user ID.

If the mail server does not require authentication, you can leave this field empty.

| Property | Value |
|---|---|
| Data type | String |

### Business Process Choreographer Explorer URL

Specifies the URL used for e-mail links to the Business Process Choreographer Explorer.

This URL is used to provide a link in generated e-mails so that a business administrator who receives an e-mail notification can click on the link to view the related business process or human task in their Web browser.

| Property | Value |
|---|---|
| Data type | String |
| Example | http://www.ibm.com:9080/bpc |

### Security

In this section you specify the mapping of functional roles to user IDs and groups, and the authentication credentials necessary for Business Process Choreographer.

### Administrator User

The administrator security role is mapped to the specified user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

### Administrator Group

The administrator security role is mapped to the specified group.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

### Monitor User

The system monitor security role is mapped to the specified user ID.

| Property | Value |
|---|---|
| Data type | String |
| Default | Currently logged on user |

### Monitor Group

The system monitor security role is mapped to the specified group.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

### JMS Authentication User

The authentication alias for the system integration bus.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | Currently logged on user |

### JMS Authentication Password and Confirm Password

The password for the JMS authentication user ID.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | None |

### JMS API Authentication User

The run-as user ID for the business flow manager message driven bean.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | Currently logged on user |

### JMS API Authentication Password and Confirm Password

The password for the JMS API authentication user ID.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | None |

### Escalation User Authentication User

The run-as user ID for the human task manager message driven bean.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | Currently logged on user |

### Escalation User Authentication Password and Confirm Password

The password for the escalation user authentication user ID.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | None |

### State Observers

In this section you can enable audit logging and Common Event Infrastructure (CEI) logging for the Business Flow Manager and Human Task Manager.

### Audit Logging for the Business Flow Manager

When selected, audit logging for the Business Flow Manager is enabled.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

### Audit Logging for the Human Task Manager

When selected, audit logging for the Human Task Manager is enabled.

| Property | Value |
| --- | --- |
| **Data type** | Check box |
| **Default** | Selected |

## Common Event Infrastructure Logging for the Business Flow Manager

When selected, Common Event Infrastructure logging for the Business Flow Manager is enabled.

| Property | Value |
| --- | --- |
| **Data type** | Check box |
| **Default** | Selected |

## Common Event Infrastructure Logging for the Human Task Manager

When selected, Common Event Infrastructure logging for the Human Task Manager is enabled.

| Property | Value |
| --- | --- |
| **Data type** | Check box |
| **Default** | Selected |

## SCA Bindings

For the Service Component Architecture (SCA) bindings, you can set the context root for the Web services API.

## Host

This read-only field illustrates the contextual prefix of the hosts for the Business Flow Manager and the Human Task Manager bindings to which the context roots are appended.

## Context Root for the Business Flow Manager

The context root for the Business Flow Manager Web service.

| Property | Value |
| --- | --- |
| **Data type** | String |
| **Default when configured on a server** | /BFMIF_*nodeName_serverName* |
| **Default when configured on a cluster** | /BFMIF_*clusterName* |

## Context Root for the Human Task Manager

The context root for the Human Task Manager Web service.

| Property | Value |
| --- | --- |
| **Data type** | String |
| **Default when configured on a server** | /HTMIF_*nodeName_serverName* |
| **Default when configured on a cluster** | /HTMIF_*clusterName* |

## Relative Path

This read-only field shows the relative paths for the SCA bindings for the Business Flow Manager and the Human Task Manager.

| Property | Value |
| --- | --- |
| **Data type** | Read-only string |

| Property | Value |
|---|---|
| **Relative path for the Business Flow Manager** | /sca/com/ibm/bpe/spi/sca/BFMWS |
| **Relative path for the Human Task Manager** | /sca/com/ibm/task/spi/sca/HTMWS |

## Bus

If you want to use a different data source for the Business Process Choreographer messaging engine than you configured for the Service Component Architecture (SCA), expand this section to change the settings.

## Use the default configuration

If selected, the current configuration settings of the SCA messaging engine will be used.

If you want to use different settings, clear the check box to enable the other fields in this section.

| Property | Value |
|---|---|
| **Data type** | Check box |
| **Default** | Selected |

## Bus Member Location

Determines whether the messaging engine's data will be stored locally or remotely.

Select between **Local** and **Remote**. If you select **Remote**, the remote destination location selector and the **New** button are enabled.

| Property | Value |
|---|---|
| **Data type** | Radio buttons |
| **Default** | Local |

## Remote destination location

Specifies the deployment target for the remote messaging engine store.

If the list is empty, or does not contain the location that you want to select, click **New**.

| Property | Value |
|---|---|
| **Data type** | Drop-down list |
| **Default** | None |

## New

This button opens the Browse deployment target page.

After selecting a deployment target, the target is added to the list of remote destination locations.

## Edit

Click this to edit the data source.

## Test Connection

Tests the connection to the data source.

## Database Instance

The name of the database.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | ${USER_INSTALL_ROOT}\databases\ BPEME |

## Schema Name

The name of the schema to be used.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | MEDBM00 |

## Create Tables

If selected, the tables will be created automatically the first time that the database is accessed. For this option to work, the database must already exist, and the user name provided must have the authority to create tables and indexes in the database.

If not selected, the tables will not be created automatically, and you must create the tables manually by running scripts. For a production system, you might prefer not to have the default tables that are created by this option.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

## 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 option to create the tables automatically can be used, and when necessary, the database schema will be updated automatically after applying a service or fix pack.

| Property | Value |
| --- | --- |
| Data type | String |

## Password

The password for the data source user ID.

| Property | Value |
| --- | --- |
| Data type | String |

## Server

The address of the database server.

Specify either the host name or the IP address, and the port number.

| Property | Value |
| --- | --- |
| Data type | String |
| Example | localhost:50000 |

**Provider**

The JDBC provider for the Business Process Choreographer messaging engine.

If you configured SCA to use a file store, this field is set to `File Store` and the fields for the database parameters are not available. When a database provider is selected, the database parameters are available.

| Property | Value |
|---|---|
| Data type | Drop-down list |
| Default | The provider that you configured for SCA. |

# Using the bpeconfig.jacl script to configure Business Process Choreographer

Describes how to use the bpeconfig.jacl script to configure Business Process Choreographer and all the necessary resources on a given server or cluster.

**Procedure**

1. Make sure that you know which options and parameters you are going to use. Refer to the values you planned in Chapter 3, "Planning to configure Business Process Choreographer," on page 91. You must include all the required parameters and options in the batch file or on the command line. Otherwise, any required parameters that are not provided will be prompted for. For detailed information about the script, its options and parameters, see "bpeconfig.jacl script file" on page 163.

| Option | Description |
|---|---|
| **If the server (or in a network deployment environment, the deployment manager) is not running** | Use the option: <br> `-conntype NONE` <br><br> Do not use this option if the server (or deployment manager) is running. |
| **If administrative security is enabled** | Include the parameters: <br> `-user` *userName* <br> `-password` *userPassword* |
| **If you are not using the default profile** | Include the parameter: <br> `-profileName` *profileName* |
| **If you are not configuring Business Process Choreographer on the default server** | Include either the parameter: <br> `-cluster` *clusterName* <br><br> or both parameters: <br> `-node` *nodeName* <br> `-server` *serverName* |

| Option | Description |
|---|---|
| **Because the script always creates a Business Process Choreographer configuration** | Include the parameters required for the Business Flow Manager and Human Task Manager:<br><br>```<br>{-adminBFMUsers userList |<br> -adminBFMGroups groupList}<br>{-monitorBFMUsers userList |<br> -monitorBFMGroups groupList}<br>-jmsBFMRunAsUser userID<br>-jmsBFMRunAsPwd password<br>{-adminHTMUsers userList |<br> -adminHTMGroups groupList}<br>{-monitorHTMUsers userList |<br> -monitorHTMGroups groupList}<br>-jmsHTMRunAsUser userID<br>-jmsHTMRunAsPwd password<br>-contextRootBFM contextRootBFM<br>-contextRootHTM contextRootHTM<br>```<br><br>For the parameter pairs ending in *Users* and *Groups* you must specify either one or both parameters. The two parameters starting with *contextRoot* are optional. |
| **If you want to enable a simple mail transfer protocol (SMTP) server for sending escalation e-mails** | Include the parameter:<br><br>```<br>-mailServerName mailServerName<br>```<br><br>If the mail server requires authentication, also include the parameters:<br><br>```<br>-mailUser mailUserID<br>-mailPwd mailPassword<br>``` |

| Option | Description |
| --- | --- |
| **Because you can either have the script file create the database, or just have it generate the SQL script without running the scripts** | Use the option<br><br>`-createDB { yes \| no }`<br><br>If you select `yes`, the bpeconfig.jacl script will generate and run an SQL file to create the database objects in the default table space, which is not suitable for a high-performance system. In this case also plan to stop the server and use the `-conntype NONE` option.<br><br>If you select `no`, and the database does not already exist, then you or your database administrator must run the generated SQL script. For a high-performance system, specify `no`, because you will need to customize the SQL script before running it. Also specify `no` if you do not have the authority to create the database yourself, so that you can provide the SQL script to your database administrator to customize and run. You must also specify `no` if you are using a database which has restricted support.<br>**Restriction:** The script cannot create the following types of databases:<br><br>• DB2 for z/OS<br><br>• Oracle<br><br>• A remote Microsoft SQL Server<br><br>• A remote Informix Dynamic Server<br><br>If you select `yes` and you are running the script in connected mode, creating the database or schema can fail if it takes longer than the a default timeout of 3 minutes. |

| Option | Description |
|---|---|
| **Because every Business Process Choreographer configuration requires access to a database** | Include the parameter:<br><br>`-dbType databaseType`<br><br>Also provide the parameters required for your database type (see "bpeconfig.jacl script file" on page 163 for details):<br><br>`-dbVersion version`<br>`-dbHome databaseInstallPath`<br>`-dbJava JDBCDriverPath`<br>`-dbName databaseName`<br>`-dbUser databaseUser`<br>`-dbPwd databasePassword`<br>`-dbAdmin databaseAdministratorUserID`<br>`-driverType JDBCDriverType`<br>`-dbTablespaceDir databaseTablespacePath`<br>`-dbServerName databaseServerName`<br>`-dbServerPort databaseServerPort`<br>`-dbStorageGroup DB2zOSStorageGroup`<br>`-dbSubSystem DB2zOSSubSystem`<br>`-dbSchema schemaQualifier`<br>`-dbInstance InformixInstance`<br><br>When running the script in batch mode on a cluster, if your database requires the `-dbJava` parameter, specify the parameter for each node that hosts a cluster member in the following way:<br><br>`-dbJava.nodeName JDBCDriverPath`<br>   `_on_nodeName`<br><br>**Note:** If you are using one of the following databases, bpeconfig.jacl can also create the database instance:<br>• A local DB2 for Linux, UNIX, or Windows<br>• DB2 on iSeries<br>• Derby Embedded<br>• Derby Network database and the server is running |
| **Because every Business Process Choreographer configuration uses a JMS provider** | Include the parameter:<br><br>`-mqType { WPM | MQSeries }`<br><br>Also provide the parameters required for your JMS provider (see "bpeconfig.jacl script file" on page 163 for details).<br><br>`-createQM { yes | no }`<br>`-qmNameGet getQueueManagerName`<br>`-mqClusterName mqClusterName`<br>`-qmNamePut putQueueManagerName`<br>`-mqHome MQInstallationDirectory`<br>`-mqUser JMSProviderUserID`<br>`-mqPwd JMSProviderPassword`<br><br>**Note:** The MQSeries® option is deprecated. |

| Option | Description |
|---|---|
| **If you are using the `-mqType WPM` option, specify the message engine store settings** | Include the following parameters:<br><br>`-meStoreType { FILESTORE ∣ DATASTORE }`<br>`-mqCreateTables { true ∣ false }`<br>`-mqSchemaName mqSchemaName`<br>`-mqDataSource datasourceName`<br>`-medbUser meDatabaseUser`<br>`-medbPwd meDatabasePassword` |
| **Because the script always configures a Business Process Choreographer Explorer** | Include any of these optional parameters:<br><br>`-hostName explorerVirtualHostname`<br>`-explorerHost explorerURL`<br>`-remoteNodeName nodeName`<br>`-remoteServerName serverName`<br>`-remoteClusterName clusterName`<br>`-contextRootExplorer explorerContextRoot`<br>`-maxListEntries maximum`<br><br>For more information about these parameters, including default values, see "Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer" on page 208. |
| **Know whether you want to install and configure the Business Process Choreographer Observer, or event collector applications on the deployment target** | Use the options:<br><br>`-createEventCollector { yes ∣ no }`<br>`-createObserver { yes ∣ no }`<br><br>These options can only be used when running bpeconfig.jacl in batch mode, and are not suitable for a high performance system. For a production system, perform "Configuring the Business Process Choreographer Observer" on page 211. |

2. If you selected the Business Process Choreographer sample configuration option when you created a default profile, the Business Flow Manager, Human Task Manager, Business Process Choreographer Explorer, and Business Process Choreographer Observer are already configured.

   You can check if they are configured, by looking in the administrative console for enterprise applications with names that start with:

   - BPCObserver
   - BPECollector
   - BPEContainer
   - BPCExplorer
   - TaskContainer

   The sample configuration uses a Derby database, and is not suitable for a production system. Because you can only have one Business Process Choreographer configuration on a deployment target, you must remove the sample configuration, as described in Chapter 5, "Removing the Business Process Choreographer configuration," on page 271 before you can continue configuring Business Process Choreographer.

3. If you have a network deployment environment, make sure that the Service Component Architecture (SCA) is configured:

   a. If you want to configure Business Process Choreographer on a server, click **Servers** ▸ **Application servers** ▸ *serverName*, then in the **Business Integration** section, click **Service Component Architecture**.

b. If you want to configure Business Process Choreographer on a cluster, click **Servers** → **Clusters** → *clusterName*, then in the **Business Integration** section, click **Service Component Architecture**.

c. If it is not enabled, select **Support the Service Component Architecture components**.

4. If you are using WebSphere Platform Messaging (WPM) as the JMS provider, and did not use either of the options `-meStoreType FILESTORE` with a Derby Embedded database, or `-meStoreType FILESTORE` then create the database for the data store for the Business Process Choreographer messaging engine:

   - If you want to use the `-mqCreateTables yes` option to have the messaging engine create the default schema the first time that it uses the database, perform the following:

     a. If the database does not already exist, create it.

     b. Grant the database user ID the rights to create tables and views in the schema that you planned to use.

   - Otherwise, if you will use the `-mqCreateTables no` option, 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.

5. If you plan to use the option `-createDB yes` to run the generated SQL scripts to create the database schema:

   a. If you are using one of the following databases:
      - DB2 for z/OS
      - Oracle
      - a remote Microsoft SQL Server
      - a remote Informix Dynamic Server

      and your database does not already exist, create an empty database manually according to the documentation for your database.

   b. Make sure that the database client, for example db2.exe, is on the path for the scripting client.

   c. Make sure that the application server is stopped.

6. Invoke the bpeconfig.jacl script file, either in batch mode providing the options and configuration parameters that you planned, or in interactive mode, For details about the script file, refer to "bpeconfig.jacl script file" on page 163.

7. If you either used the `-createDB no` option to defer that creation of the database, or if the bpeconfig.jacl script failed create the database, you or your database administrator should perform the actions described in "Using a generated SQL script to create the database schema for Business Process Choreographer" on page 177 before you activate Business Process Choreographer in step 8.

   **Note:** If your database is local, and it will exist by the time you activate Business Process Choreographer in step 8, and you do not perform the actions described in "Using a generated SQL script to create the database schema for Business Process Choreographer" on page 177, the default schema will be created the first time that Business Process Choreographer accesses the database.

8. Activate Business Process Choreographer: Perform "Activating Business Process Choreographer" on page 266.

9. Optional: Verify that the basic Business Process Choreographer configuration works: Perform "Verifying that Business Process Choreographer works" on page 267.

10. Optional: If you want to change the JMS authentication user IDs, the run-as user IDs, or the mappings of roles onto users and groups, click **Security** → **Business Integration security** to change the security settings.

11. Optional: Change settings for the Human Task Manager:
    - If you want to change any of the Human Task Manager settings for the escalation e-mails, such as the sender address, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and make your changes.
    - If you want to change the e-mail server address or port number, the user ID or password for the e-mail server, or the URL prefix for the Business Process Choreographer Explorer, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, under **Container Settings**, expand **Business Process Choreographer Container Settings**, click **Business Process Choreographer Containers**, and make your changes.

12. Depending on the type of people directory provider that you use for people assignment, you might need to configure it:
    - The system and user registry people directory providers can be used without configuring them.
    - If you are using Lightweight Directory Access Protocol (LDAP), perform "Configuring the LDAP people directory provider" on page 194.
    - If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 193.

13. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 199.

14. Optional: If you want to use group work items, use the administrative console to enable them. Click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name* if Business Process choreographer is configured on a cluster, then under **Business Integration**, expand **Business Process Choreographer Container**, click **Human Task Manager**, and select **Enable group work items**.

15. Optional: If you have not yet installed and configured Business Process Choreographer Explorer, you can configure it now. Perform "Configuring Business Process Choreographer Explorer" on page 207.

16. Optional: If you have not yet installed and configured Business Process Choreographer Observer, you can configure it now. Perform, "Configuring the Business Process Choreographer Observer" on page 211.

**Results**

Business Process Choreographer is configured.

# bpeconfig.jacl script file

This script file configures Business Process Choreographer and all the necessary resources on a server or cluster.

## Purpose

This script can either be run interactively, or in batch mode. It can create a local database, the necessary messaging resources, and can optionally configure the Business Process Choreographer Explorer and Business Process Choreographer Observer.

## Location

The bpeconfig.jacl script file is located in the Business Process Choreographer `config` directory:
- On Linux, UNIX, and i5/OS platforms: In the directory *install_root*/ ProcessChoreographer/config
- On Windows platforms: In the directory *install_root*\ProcessChoreographer\ config

## Restrictions

This script has the following restrictions:

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

**For a DB2 database**
> The bpeconfig.jacl script cannot create a database if the Universal Driver type 4 is selected even if DB2 is installed locally.

**For an Oracle database**
> The bpeconfig.jacl script cannot create an Oracle database. If you want to use an Oracle database for Business Process Choreographer, you must create the database manually.

**For a Microsoft SQL Server database**
> The bpeconfig.jacl script cannot create a remote database. To create a local database, use a type-2 JDBC driver, and do not specify the -dbServerName parameter. If you want to use a remote Microsoft SQL Server database for Business Process Choreographer, you must create the database manually.

## Runing 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 WebSphere administrative security is enabled, include the `-user` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

## Runing the script in a network deployment environment

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 WebSphere administrative security is enabled, include the `-user` 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, enter one of the following commands:

On Linux and UNIX platforms, if your current directory is *install_root*, enter the command:

```
bin/wsadmin.sh -f ProcessChoreographer/config/bpeconfig.jacl parameters
```

On i5/OS platforms, if your current directory is *install_root*, enter the command:

```
bin/wsadmin -f ProcessChoreographer/config/bpeconfig.jacl parameters
```

On Windows platforms, if your current directory is *install_root*, enter the command:

```
bin\wsadmin -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 VirtualHostname
-explorerHost explorerURL
-remoteNodeName nodeName
-remoteServerName serverName
-remoteClusterName clusterName
-contextRootExplorer explorerContextRoot
-maxListEntries max
-createDB { yes | no }
-dbType databaseType
-dbVersion version
-dbHome databaseInstallPath
-dbJava JDBCDriverPath
-dbName databaseName
-dbUser databaseUser
-dbPwd databasePassword
-driverType JDBCDriverType
-dbTablespaceDir databaseTablespacePath
-dbServerName databaseServerName
-dbServerPort databaseServerPort
-dbStorageGroup DB2zOSStorageGroup
-dbSubSystem DB2zOSSubSystem
```

```
                    -dbSchema schemaQualifier
                    -dbInstance InformixInstance
                    -mqType JMSProviderType
                    -createQM { yes | no }
                    -qmNameGet getQueueManagerName
                    -mqClusterName mqClusterName
                    -qmNamePut putQueueManagerName
                    -mqHome MQInstallationDirectory
                    -mqUser JMSProviderUserID
                    -mqPwd JMSProviderPassword
                    -mqSchemaName mqSchemaName
                    -mqCreateTables  { true | false }
                    -mqDataSource datasourceName
                    -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 are
described for each parameter in the following descriptions. Any required
parameters that are not specified on the command line are prompted for
interactively in the sequence that they are listed.

## 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
> network deployment) is not running.

**-user** *userName*
> If WebSphere administrative security is enabled, you must provide a user ID
> for authentication.

**-password** *userPassword*
> If WebSphere administrative 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. Specify this option if
> you are not configuring the default profile. The profile must already exist.

**-node** *nodeName*
> Where *nodeName* is the name of the node where Business Process
> Choreographer will 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 will 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 J2EE role JMSAPIUser. This property is required to configure the business process container. This parameter has no default value. It must be set.

**-jmsBFMRunAsPwd** *password*

Where *password* is the password for the jmsBFMRunAsUser user ID. This property is required to configure 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 human 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 human 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 (|). This property is needed to install the human 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 (|). This property is needed to install the human 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 J2EE role EscalationUser. This property is required to configure the human task container. This parameter has no default value. It must be set.

**-jmsHTMRunAsPwd** *password*

Where *password* is the password for the jmsHTMRunAsUser user ID. This property is required to configure 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`.

**-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`.

**-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 set to an empty value, 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. This parameter is only needed if the mail server requires authentication. Otherwise, it can be omitted. This parameter 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 for the user ID *mailUserID*. This parameter is only needed if the mail server requires authentication. Otherwise, it can be omitted. This parameter is needed to create the mail session for the Human Task Manager to send notification mails.

**-hostName** *VirtualHostname*

Where *VirtualHostname* is the virtual host where the Business Process Choreographer, and the Web service bindings of the Business Flow Manager and Human Task Manager APIs will run. The default value is `default_host`.

**-maxListEntries** *maximum*

Where *maximum* is the maximum number of results that the Business Process Choreographer Explorer will return for a query. The default is 10000.

**-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 by the Human Task Manager to link to this explorer instance.

**-precompileJSPs { no | yes }**

Determines whether Java Server Pages (JSPs) will be precompiled, or not. The default is no. Note that it is not possible to debug precompiled JSPs.

**-remoteNode** *nodeName*

Use this parameter and remoteServer if you do not want to connect to the local Business Process Choreographer configuration. If this parameter is not specified, it defaults to the value of the -node parameter.

**-remoteServer** *serverName*

Use this parameter and remoteNode if you do not want to connect to the local Business Process Choreographer configuration. If this parameter is not specified, it defaults to the value of the -server parameter.

**-remoteCluster** *clusterName*

Use this parameter, if you do not want to connect to the local Business Process Choreographer configuration and you do not specify remoteNode and remoteServer. If this parameter is not specified, it defaults to the value of 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*/bpc. 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 and Oracle, 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 production systems, use no. If you use yes, the command prompt from which bpeconfig.jacl is invoked must have the appropriate paths set to run the corresponding database commands, for example, db2.exe.

**-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:
- Derby
- DB2
- zOS-DB2
- Informix
- iSeries-DB2
- MSSQL
- Oracle

**-dbVersion** *version*

Where *version* is the database version number. It has no default value. It is only required for the following database types:
- For DB2 for z/OS, *version* must have either the value 7, 8, or 9.
- For Oracle, *version* must have either the value 9 or 10.
- For MSSQL, *version* must have either the value 2000 if the database has no Unicode support, or 2000U if the database has Unicode support.

**-dbHome** *databaseInstallPath*

Where *databaseInstallPath* is the installation directory of the database system. This parameter is only required for Informix and optionally for DB2 if the createDB parameter is set to **Yes**. It is used to create the database or the database tables, and for creating the data source. The default value, and requirements depend on the database and on the platform:

**For DB2:**
- On Windows platforms, the default is *current_drive*`\Program Files\IBM\SQLLIB` where *current_drive* is the current drive letter.
- On Solaris platforms, the default is `/export/home/${dbUser}/sqllib`.
- On other platforms, the default is `/home/${dbUser}/sqllib`.

The directories `${dbHome}/bnd` and `${dbHome}/bin` must exist.

**For Informix:**
- On Windows platforms, the default is *current_drive*`\Program Files\Informix` where *current_drive* is the current drive letter.
- On Solaris and HP-UX platforms, the default is `/opt/informix`.
- On Linux and AIX platforms, the default is `/usr/informix`.

The file `${dbHome}/jdbc/lib/ifxjdbc.jar` must exist.

**-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 Universal with a type-4 driver. The default value is *databaseInstallPath*`/java`.
- DB2 for i5/OS with a type-2 (Native) driver. The default value is `/QIBM/ProdData/Java400/ext` .
- DB2 for i5/OS with the type-4 (Toolbox) driver. The default value is `/QIBM/ProdData/HTTP/Public/jt400/lib/java`.
- DB2 for z/OS, with a type-4 driver. The default value is *databaseInstallPath*`/java`.
- Informix. The default value is *databaseInstallPath*`/jdbc/lib`.
- MSSQL DataSource with the `DataDirect` driver type. There is no default value.
- Oracle. The default value is *databaseInstallPath*`/jdbc/lib`.

Where *databaseInstallPath* is the installation directory for the database system.

When running the script in batch mode on a cluster, if your database requires the **-dbJava** parameter, specify the parameter for each node that hosts a cluster member in the following way:

`-dbJava.`*nodeName* *JDBCDriverPath*`_on_`*nodeName*

Where *JDBCDriverPath* is the path to the JDBC driver and *nodeName* is the name of the node.

**-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`.
- For Oracle, this is the TNS.
- For Derby Network (not Derby Embedded) this must be an absolute path name.
- For i5/OS, it is the database name or IASP hardware device name. When using the Toolbox JDBC driver, the default is `*SYSBAS`, when using the Native driver the default is `*LOCAL`.

**-dbUser** *databaseUser*

Where *databaseUser* is the user ID to access the database. It is used to create the data source. The default value depends on the database and platform:

- For DB2 on Windows platforms: "db2admin"
- For DB2 on other platforms: "db2inst1"
- For Derby Network: The user ID of the currently logged on user
- For Informix: "informix"
- For Oracle: "system"
- For MSSQL: The user ID of the currently logged on user

**-dbPwd** *databasePassword*

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

**-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`.
- For DB2 on i5/OS: possible values are `native` or `toolbox`.
- For Derby: possible values are `Embedded` or `Network`.
- For Oracle, possible values are `oci8` or `thin`.
- For MSSQL, possible values are `Embedded` or `DataDirect`.

**-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 Oracle, there is no default value. You must provide a value.
- For DB2, the default value is empty, which means that no table spaces are created.

**-dbServerName** *databaseServerName*

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

- For DB2, the default value is empty. For DB2 UDB, this parameter is optional, and if it is not specified, a type 2 JDBC driver will be configured for DB2, otherwise a type 4 JDBC provider will be configured.
- For DB2 on i5/OS, specify the server short name. When using the Toolbox driver, the default is the short name of the local host.
- 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.
- For Derby Network, the default value is 1527.
- For Informix, the default value is 1526.
- For MSSQL, the default value is 1433.
- For Oracle with the driver type thin, the default value is 1521.

**-dbStorageGroup** *DB2zOSStorageGroup*

Where *DB2zOSStorageGroup* is the storage group used to create the Business

Process Choreographer database tables. This parameter is only required for DB2 on z/OS. There is no default value, and must not be empty.

**-dbConnectionTarget** *DB2zOSSubSystem*
> Where *DB2zOSSubSystem* is the DB2 connection target location used to create the Business Process Choreographer database tables and the data source. This parameter is only required for DB2 on z/OS. The default value is BPEDB.

**-dbSchema** *schemaQualifier*
> For i5/OS, *schemaQualifier* is the collection name, the default value is BPEDB. For all other platforms *schemaQualifier* is the schema qualifier used to create the Business Process Choreographer database tables and the data source. The default value is empty, which means to use the implicit schema qualifier, which depends on the database type being used.

**-dbInstance** *InformixInstance*
> Where *InformixInstance* is the instance name for a Business Process Choreographer Informix database. The default value is ids1.

**-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. Using this value is deprecated.

**-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, which is deprecated. 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, which is deprecated.

**-mqClusterName** *mqClusterName*
> Where *mqClusterName* is the name of the WebSphere MQ cluster that the queue manager will join. This parameter is optional. The default value is MQCluster. This option only has an effect if the parameter mqType has the value MQSeries, which is deprecated.

**-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 platforms 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`, which is deprecated.

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

**Windows platforms:**
> *current_drive*\Program Files\IBM\WebSphere MQ

**AIX:** /usr/mqm

**i5/OS:** /QIBM/ProdData/mqm

**Solaris, HP-UX, and Linux:**
> /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 authenticate against the Business Process Choreographer SI bus; the default value is the currently logged on user.
- If mqType has the value `MQSeries`, this parameter is used on Linux and UNIX 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.

**-meStoreType** *{ FILESTORE | DATASTORE }*
Set the message store type for the Business Process Choreographer message engine. If the mqDataSource parameter is supplied, this parameter is set to `DATASTORE`. If the Service Component Architecture (SCA) is using `FILESTORE`, this parameter is set to `FILESTORE`. `FILESTORE` is not supported in Network Deployment environments. If mqDataSource is not set, and the SCA uses `DATASTORE` as its message store type, then the SCA message engine database settings, such as database type, JDBC provider, and database server, are inherited. In this case, a separate database schema has to be set though (see mqSchemaName below), and the mqCreateTables flag can be overwritten as well.

**-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 meStoreType is set to `DATASTORE`.

**-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. If this flag is set to `true` for the Service Component Architecture (SCA), this parameter is also set to `true`. If this flag is set to `false` for the SCA,

this parameter is also set to `false`. This option is only used when the mqType option is set to `WPM` and meStoreType is set to `DATASTORE`.

**-medbUser** *MEDBUserID*
Where *MEDBUserID* is the user ID to access the messaging engine database. The default value for this parameter is the value of the dbUser parameter. The parameter is only required when the meStoreType parameter is set to *DATASTORE*, and the messaging engine database is not accessed through the Derby Embedded JDBC provider.

**-medbPwd** *MEDBPassword*
Where *MEDBPassword* is the password for the user ID that is provided for the medbUser parameter. This parameter has no default value.

**-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. Using this option, you cannot specify a different database, the BPEB database is used by default, which means that this option is not suitable for a high performance system. 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. You can only use this option in non-interactive mode. Using this option, you cannot specify a different database, the BPEB database is used by default, which means that this option is not suitable for a high performance system. 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" on page 211.

1. On the server, or for network deployment, on the deployment manager, start the script:
   - On Linux and UNIX platforms, enter the command:
     ```
     install_root/bin/wsadmin.sh
             -f install_root/ProcessChoreographer/config/bpeconfig.jacl
       ( [-user userName][-password password]|[-conntype NONE])
         [-profileName profileName]
     ```
   - On i5/OS platforms, enter the command:
     ```
     install_root/bin/wsadmin
             -f install_root/ProcessChoreographer/config/bpeconfig.jacl
       ( [-user userName][-password password]|[-conntype NONE])
         [-profileName profileName]
     ```
   - On Windows platforms, enter the command:
     ```
     install_root\bin\wsadmin.bat
             -f install_root\ProcessChoreographer\config\bpeconfig.jacl
       ( [-user userName][-password password]|[-conntype NONE])
         [-profileName profileName]
     ```
2. Interactively enter responses to the questions that are displayed:

a. In a network deployment environment, you will be offered a server, or cluster, to configure in. If it is not the correct server, or cluster, enter **No** to be offered the next server, or cluster. If it is the correct server, or cluster, enter **Yes**.

b. For the question `Install the business process container?`, enter **Yes**.

c. For the question `User(s) to add to role BPESystemAdministrator`, 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 BPESystemAdministrator`, 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 BPESystemMonitor`, 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 BPESystemMonitor`, enter the groups from the domain user registry that are mapped onto the role of business process monitor.

g. For the question `Run-as UserId for role JMSAPIUser`, enter the run-as user ID that will be used for the JMSAPIUser role.

h. Enter the password for the run-as user ID.

i. For the question `Use WebSphere default messaging or WebSphere MQ [WPM/MQSeries]?`, select the JMS provider that you want to use.

j. 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` for the Service Component Architecture (SCA) Web server virtual host.

   2) For the question `Context root for the SCA Web Service [/BFMIF_PNODE_server1]:`, press **Enter** to accept the default value `BFMIF_`*nodeName_serverName*.

k. For the question `Create the DataSource for the Process Choreographer database?`, enter **Yes**.

l. For the question `Create DataSource for a Derby, a DB2, an Informix, an Oracle, or an SQL Server database [Derby/DB2/zOS-DB2/iSeries-DB2/ Informix/Oracle/MSSQL]?`, for this example, enter **DB2**. Selecting a different database results in other database-specific questions.

m. Enter the database name.

n. At the `Database schema name (may be empty)` prompt, hit **Enter** to use the implicit schema qualifier.

o. For the question `Universal or CLI?`, hit **Enter** to select the default Universal JDBC driver

p. For the question `DB2 User ID`, enter the user ID to access the database.

q. Enter the password for the database user ID.

r. For the question `Database server name (may be empty, set to use the type 2 driver)`, enter the name of the server that hosts the database.

s. For the question `Database server port`, enter the database server port, for example, 50000.

t. At the `JDBC driver directory on [yourHost]` prompt, enter the directory where the DB2 JDBC driver JAR files are located.

u. For the question `Create the Process Choreographer database objects?`, if your currently logged on user ID has sufficient authority to create the database, and DB2 has been set up in your current environment (for

example, the 'db2' exeuctable is on the PATH), you can enter **Yes**, otherwise, if your currently logged on user ID does not have sufficient authority to create the database, enter **No**.

If the answer is **Yes**:

1) For the question `DB2 tablespace directory (may be empty)` hit **Enter** to leave it empty.

2) For the question `Is 'BPEDB' an existing database (the Process Choreographer schema must not yet exist)` prompt according to your environment.

v. If you get the question `User ID for access to Process Choreographer SI bus`, enter the user ID to use to access the default JMS provider.

w. Enter the password for the SI bus authentication user ID.

x. For the question `Install the task container?`, enter **Yes**.

y. 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.

z. 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.

aa. 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.

ab. For the question `Run-as UserID for role EscalationUser`, enter the run-as user ID for the role of escalation user, for example db2admin.

ac. Enter the password for the escalation user ID. This prompt will be hidden if you used the same user ID as for step 2g on page 175.

ad. For the question `Context root for the SCA Web Service [/HTMIF_`*nodeName*`_`*serverName*`]:` , enter the context root for the Service Component Architecture (SCA) Web server, or press **Enter** to accept the default value.

ae. 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. Optionally, you can specify the user ID and password.

af. For the question `Maximum number of list entries for the Process Choreographer Explorer` , press **Enter** to use the default value 10000.

ag. For 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.

ah. For the question `Install the Business Process Choreographer Explorer?`, enter **Yes** to install Business Process Choreographer Explorer, 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.

ai. Various information is displayed, for example providing the URL of the Business Process Choreographer Explorer. For example:

```
*******************************************************************************
* NOTE: The Process Choreographer URL will be used by the
* Human Task Manager on server server1 of node viennaNode01
* to link to this Explorer instance. Set an empty URL to not create this link.
*******************************************************************************
URL for this Process Choreographer Explorer [http://host_name:9080/bpc]:
```

Enter the URL for this Business Process Choreographer Explorer instance, or press **Enter** to accept the default.

aj. A reminder is displayed about where to find the script files that you can use to configure Business Process Choreographer Observer.

```
To interactively configure the EventCollector, please use the script
setupEventCollector located in install_root\ProcessChoreographer\config.
To interactively configure the Observer, please use the script
setupObserver located in install_root\ProcessChoreographer\config.
```

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:

- On Linux, UNIX, and i5/OS platforms: In the directory *profile_root*/logs
- On Windows platforms: In the directory *profile_root*\logs

If you run the script in connected mode, also check the files SystemOut.log and SystemErr.log that can be found in the subdirectory of the logs directory that is named after the application server or deployment manager that the wsadmin scripting client connected to.

**Related tasks**

"Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 157
Describes how to use the bpeconfig.jacl script to configure Business Process Choreographer and all the necessary resources on a given server or cluster.

# Using a generated SQL script to create the database schema for Business Process Choreographer

The bpeconfig.jacl script generates an SQL script that creates the database objects for Business Process Choreographer.

**Before you begin**

You have used the bpeconfig.jacl script to configure Business Process Choreographer, and either you used the -createDB no option to defer creating the database objects, or the bpeconfig.jacl script failed to create the database.

**About this task**

All the relevant configuration parameters that you provided when configuring Business Process Choreographer have been substituted in the generated SQL file. You either want the database for a high-performance Business Process Choreographer configuration, or your database administrator must create the database for you, or both.

**Procedure**

1. Locate the generated SQL script on the node where you ran the bpeconfig.jacl script.

| Option | Description |
|---|---|
| **For Linux and UNIX** | • If you specified a schema prefix, the generated script is: *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name*/*database_schema*/createSchema.sql.<br><br>• If you did **not** specify a schema prefix, the generated script is: *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name*/createSchema.sql. |
| **For i5/OS** | The generated script is: *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*collection_name*/createSchema.sql. |
| **For Windows** | • If you specified a schema prefix, the generated script is: *profile_root*\dbscripts\ProcessChoreographer\*database_type*\*database_name*\*database_schema*\createSchema.sql<br><br>• If you did **not** specify a schema prefix, the generated script is: *profile_root*\dbscripts\ProcessChoreographer\*database_type*\*database_name*\createSchema.sql<br><br>**Note:** For SQL Server, there is also a version named createSchemaUnicode.sql, which you should use if your database is configured for Unicode. |
| **For z/OS** | There is an ASCII SQL script, named createSchema.sql, and an equivalent EBCDIC DDL script, named createSchema.ddl:<br><br>• If you specified a schema prefix, both files are located in *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name*/*database_schema*<br><br>• If you did **not** specify a schema prefix, both files are located in *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name* |

Where:

*database_type*
>is one of the following:
>• DB2
>• DB2zOSV7
>• DB2zOSV8
>• Db2iSeries
>• Derby

- Informix
- Oracle
- SQLServer

*database_name*
> is the name of your database.

*database_schema*
> is the name of the schema, if you are using one.

*collection_name*
> is the name of the collection; only for DB2 on iSeries.

2. If database does not yet exist, and it is not a Derby database, get your database administrator to create the database and user IDs according to the values you planned in "Planning the BPEDB database" on page 108 and "Planning security, user IDs, and authorizations" on page 101.

3. If the database is remote, copy the generated script to the remote node. If you are not authorized to perform this, give your database administrator a copy of the script and discuss your requirements with her.

4. You or your database administrator must customize the SQL script:

   a. For a high-performance system specify the allocation of disks and table spaces that you planned in step 5 on page 110 of "Planning the BPEDB database" on page 108.

5. Run the SQL script on the database host using one of the following commands:

| Option | Description |
|---|---|
| **For DB2 on Linux, UNIX, or Windows** | `db2 -tf createSchema.sql` |
| **For DB2 on iSeries** | `db2 -tf createSchema.sql` |
| **For DB2 on z/OS** | For the ASCII version:<br>`db2 -tf createSchema.sql`<br><br>For the EBCDIC version:<br>`db2 -tf createSchema.dll` |
| **For a Derby database** | `java -Dij.protocol=jdbc:derby:`<br>`     -Dij.database=BPEDB`<br>`     org.apache.derby.tools.ij`<br>`     createSchema.sql` |
| **For an Informix database** | `dbaccess databaseName createSchema.sql` |
| **For an Oracle database** | `sqlplus userID/password`<br>`     @database_name@createSchema.sql` |
| **For an SQL Server database** | For an ASCII database:<br>`sqlcmd -U userID`<br>`        -P password`<br>`        -d database_name`<br>`        -i createSchema.sql`<br><br>For a Unicode database:<br>`sqlcmd -U userID`<br>`        -P password`<br>`        -d database_name`<br>`        -i createSchemaUnicode.sql` |

6. Configure Java Database Connectivity (JDBC) to access a remote database: Perform the following steps either:

- On each member of a cluster where you configured Business Process Choreographer.
- On any server that runs Business Process Choreographer without a local database.
- Otherwise, if a server has a local database, do not perform the following steps.

a. Install a suitable type-2 database client or type-4 JDBC driver on the server that hosts the application server.

b. If you are using a type-2 JDBC driver, make the new database known to the database client by performing the following:

   **For Derby**
   > This step does not apply because only the type-4 JDBC provider is supported.

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

   **For DB2 for iSeries**
   > The database must be cataloged and accessible through an alias name.

   **For DB2 for z/OS**
   > The database must be cataloged and accessible through an alias name.

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

   **For Microsoft SQL Server**
   > This step does not apply because only type-4 JDBC providers are supported.

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

c. Using the administrative console, test the connection to the database.

   1) Click **Resources** → **JDBC** → **Business Integration Data Sources**

   2) If necessary, select a different scope and click **Apply**.

      **Note:** For clustered Business Process Choreographer configurations, the data source is defined at the cluster level. For non-clustered configurations, the data source is defined at the server level.

   3) Locate and select the data source with the JNDI name `jdbc/BPEDB`.

   4) Click **Test connection**.

   5) You should see a message indicating that the test connection was successful.

**Results**

The Business Process Choreographer database exists and is accessible from any remote servers or cluster members where Business Process Choreographer is configured.

# Using SQL scripts to create the database for Business Process Choreographer

Create the database for use in a production environment or an advanced topology, and make it remotely accessible.

**About this task**

SQL scripts are provided to create and administer the database schema for all supported database systems. Using this approach, Business Process Choreographer has its own database, which is important for performance. In a standalone server setup, the database is dedicated to the Business Process Choreographer configuration on one application server. In a clustered Business Process Choreographer setup, the database is shared between all members of the cluster.

The scripts assign separate storage for different types of database objects, for example, for template data and instance data. This separation enables better load balancing and performance tuning. To achieve the best performance you can customize the scripts before you run them.

**Procedure**

1. On the server that hosts the database, create the database according to the description for your database system.
   - "Creating a Derby database for Business Process Choreographer" on page 182.
   - "Creating a DB2 for i5/OS database for Business Process Choreographer" on page 183.
   - "Creating a DB2 for Linux, UNIX, and Windows database for Business Process Choreographer" on page 184.
   - "Creating a DB2 for z/OS database for Business Process Choreographer" on page 186.
   - "Creating an Informix Dynamic Server database for Business Process Choreographer" on page 188.
   - "Creating a Microsoft SQL Server database for Business Process Choreographer" on page 189.
   - "Creating an Oracle database for Business Process Choreographer" on page 191.
2. Configure Java Database Connectivity (JDBC) to access a remote database: Perform the following steps either:
   - On each member of a cluster where you configured Business Process Choreographer.
   - On any server that runs Business Process Choreographer without a local database.
   - Otherwise, if a server has a local database, do not perform the following steps.
   a. Install a suitable type-2 database client or type-4 JDBC driver on the server that hosts the application server.
   b. If you are using a type-2 JDBC driver, make the new database known to the database client by performing the following:
      **For Derby**
         This step does not apply because only the type-4 JDBC provider is supported.

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

**For DB2 for iSeries**
> The database must be cataloged and accessible through an alias name.

**For DB2 for z/OS**
> The database must be cataloged and accessible through an alias name.

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

**For Microsoft SQL Server**
> This step does not apply because only type-4 JDBC providers are supported.

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

c. Using the administrative console, test the connection to the database.

1) Click **Resources → JDBC → Business Integration Data Sources**

2) If necessary, select a different scope and click **Apply**.

   **Note:** For clustered Business Process Choreographer configurations, the data source is defined at the cluster level. For non-clustered configurations, the data source is defined at the server level.

3) Locate and select the data source with the JNDI name `jdbc/BPEDB`.

4) Click **Test connection**.

5) You should see a message indicating that the test connection was successful.

**Results**

The Business Process Choreographer database exists and is accessible from any remote servers or cluster members where Business Process Choreographer is configured.

# Creating a Derby database for Business Process Choreographer

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

**About this task**

The Derby database system comes with the WebSphere Process Server.

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

**Procedure**

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. On Windows platforms, create *profile_root*\databases. On Linux, UNIX, and i5/OS platforms create *profile_root*/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 directory that you created in step 1 on page 182.
   - On Linux, UNIX, and i5/OS platforms, copy the file *install_root*/dbscripts/ProcessChoreographer/Derby/createDatabase.sql
   - On Windows platforms, copy the file *install_root*\dbscripts\ProcessChoreographer\Derby\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 :
   - On Windows platforms, add the path: *install_root*\java\bin\
   - on Linux, UNIX, and i5/OS platforms, add the path *install_root*/java/bin/

4. Customize your copy of the database creation script, `createDatabase.sql`, according to the instructions in the header. You must include the name of the database. On Windows platforms, avoid using the Notepad editor, as it does not display the file in a readable format.

5. Create the database. From the directory where the database is to be created, run your customized version of the database creation script file `createDatabase.sql` as described in the header.

**Results**

The database for Business Process Choreographer exists.

## Creating a DB2 for i5/OS database for Business Process Choreographer

Use this task to create the DB2 for i5/OS database schema for Business Process Choreographer.

**Before you begin**

You completed "Planning the BPEDB database" on page 108.

**Procedure**

1. On the machine that hosts the database: If no collection exists for the user ID that owns the database, create a collection.

2. Copy the create schema script to the machine that hosts the database.
   - On Linux, UNIX, and i5/OS platforms, it is located in *install_root*/dbscripts/ProcessChoreographer/Db2iSeries/createSchema.sql.
   - On Windows platforms, it is located in *install_root*\dbscripts\ProcessChoreographer\Db2iSeries\createSchema.sql.
   a. If you want the default schema to be created the first time that Business Process Choreographer uses the database, and you will select the **Create tables** option on the configuration page, just skip this step.

b. If you will **not** select the **Create tables** option, make a local copy the SQL file createSchema.sql, and customize it according to the values you planned in "Planning the BPEDB database" on page 108 and "Planning security, user IDs, and authorizations" on page 101.

3. Create the database objects. In a qshell environment, run your copy of the script. For example, if the script is in your current directory, enter the command:

```
db2 -tf createSchema.sql
```

4. If the database is remote, use the Toolbox JDBC driver. Download the JAR file /QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar from the database host.

5. If the database is local, use the Native JDBC driver. Make sure that your classpath includes /QIBM/ProdData/Java400/ext/db2_classes.jar.

**Results**

The DB2 for i5/OS schema for Business Process Choreographer exists.

# Creating a DB2 for Linux, UNIX, and Windows database for Business Process Choreographer

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

**Before you begin**

You completed "Planning the BPEDB database" on page 108.

**Procedure**

1. Install DB2 on the server that hosts the database.

2. Install a DB2 client on all remote application servers that use a type-2 Java Database Connectivity (JDBC) driver to access the database.

3. 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, enter:

   ```
   cd install_root\dbscripts\ProcessChoreographer\DB2
   ```

   • On UNIX and Linux systems, enter:

   ```
   cd install_root/dbscripts/ProcessChoreographer/DB2
   ```

4. If your database is on a different server than your application server:

   a. Copy the following SQL scripts to your database server:

   ```
   createDatabase.sql
   createTablespace.sql
   createSchema.sql
   dropSchema.sql
   dropTablespace.sql
   ```

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

5. If you want to use an existing database, skip to step 9 on page 185 to create the table space and schema. Unicode support: Make sure that the database supports Unicode (UTF-8). Without Unicode support, it cannot store all characters that can be handled in Java code, and you can run into code page conversion problems when a client uses an incompatible code page.

6. Create a DB2 instance on the database server.

7. If you have a Symmetric Multi-Processor (SMP) server check how many processors can be used by DB2. Check your license:

- On AIX systems, enter the command:

  `/usr/opt/db2_08_01/adm/db2licm -l`

- On other UNIX or Linux systems, enter the command:

  `/opt/IBM/db2/V8.1/adm/db2licm -l`

If necessary, change the number of processor licenses using either the db2clim command or the DB2 License Center.

8. Create a new database:

   a. Make sure that you use a user ID that has administrator rights for the database system.

   b. If you want to create a non-production database, named BPEDB, for stand-alone development, evaluation, or demonstration purposes, enter the following command:

      `db2 -tf createDatabase.sql`

      For a production environment consider using dedicated table space containers and adjusting the DB2 parameters.

9. To create the table space and schema:

   a. Customize a copy of the createTablespace.sql table space creation script according to what you planned in "Planning the BPEDB database" on page 108.

   b. Make sure that you have administrator rights for the database system. The user ID that you use to create the schema must be the same one that you specify when configuring the data source for the business process container.

   c. Make sure that you are attached to the correct instance. Check the DB2INSTANCE environment variable.

   d. To connect to a database named *databaseName*, in the DB2 command-line processor, enter the command:

      `db2 connect to databaseName`

   e. To create the table spaces, enter the command:

      `db2 -tf createTablespace.sql`

      Make sure that the script output contains no errors. If errors occur, you can drop the table space using the dropTablespace.sql script.

   f. To create the schema (tables, indexes, and views) in the DB2 command-line processor, enter the command:

      `db2 -tf createSchema.sql`

      Make sure that the script output contains no errors. If you want to drop the schema, use the dropSchema.sql script.

   **Note:** If you do not create the table space and schema now, you must use the **Create tables** option later so that the default table space and schema will be created the first time that Business Process Choreographer attempts to use the database.

10. On each application server that remotely accesses the database:

    a. Catalog the database by entering the command:

       `db2 catalog database databaseName as databaseAlias at node nodeName`

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

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

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

**Results**

The database for Business Process Choreographer exists.

# 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.

**Before you begin**

You completed "Planning the BPEDB database" on page 108.

**About this task**

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.

**Procedure**

1. You must have already installed WebSphere Process Server on a UNIX, Linux, Windows, or i5/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. Create the database and storage group. Perform one of the following:
      - Use the DB2 administration menu to create a new database and storage group.
      - Edit a copy of the createDatabase.sql script file according to the instructions in the header, then run your copy. It is located in the directory:
        – On Windows platforms: *install_root*\dbscripts\ProcessChoreographer\ *database_type*
        – On Linux, UNIX, i5/OS, and in the UNIX System Services (USS) on z/OS platforms: *install_root*/dbscripts/ProcessChoreographer/ *database_type*

        Where *database_type* is one of the following:
        – DB2zOSV7
        – DB2zOSV8

        Make a note of the names.
   e. 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.
   f. 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.

g. 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. 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. Make sure that you have DB2 Connect™ Gateway installed. DB2 Connect Gateway is part of the DB2 UDB ESE package, but you can also install it separately.

b. Catalog the remote database using the following commands in a DB2 command line window:

```
catalog tcpip node zosnode remote host_name server IP_port ostype mvs
catalog database location as database_alias at node zosnode authentication dcs
catalog dcs database database_alias parms ',,INTERRUPT_ENABLED'
```

c. Make sure that the sync point manager instance name is specified. Enter the following commands:

```
db2 update dbm cfg using SPM_NAME host_name
db2 update dbm cfg using SPM_LOG_FILE_SZ log_file_size
```

d. 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.

e. Verify that you can establish a connection to the remote subsystem by entering the following command:

```
db2 connect to subsystem user userid using password
```

f. 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\DB2zOSV7
cd install_root\dbscripts\ProcessChoreographer\DB2zOSV8
```

- On UNIX, Linux, and i5/OS systems, depending on your DB2 version, enter one of the following commands:

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

g. Customize a copy of the createTablespace.sql table space creation script according to what you planned in "Planning the BPEDB database" on page 108. Replace @STOGRP@ with the storage group name and replace @DBNAME@ with the database name (not the subsystem name).

h. 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, customize and run the dropTablespace.sql script.

i. 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 @SCHEMA@ with the schema qualifier or remove @SCHEMA@ (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.

j. 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 @SCHEMA@ before running the script.

**Results**

The database for Business Process Choreographer exists.

## Creating an Informix Dynamic Server database for Business Process Choreographer

Use this task to create an Informix Dynamic Server database for Business Process Choreographer.

**Before you begin**

You completed "Planning the BPEDB database" on page 108.

**Procedure**

1. Install the Informix server on the computer that hosts the database.
2. Create an Informix server instance. Make sure that the Informix environment variables are set correctly. In particular, *INFORMIXSERVER* must point to the new instance and *ONCONFIG* must point to the configuration file for the instance. For more details about the different environment variables and how they need to be set up, refer to the Informix Dynamic Server documentation. Make sure that the environment variables relating to Global Language Support (GLS) are set to Unicode (UTF-8) support. Unicode support is required to store all the characters that can be handled in Java code.
3. Copy and configure the Java Database Connectivity (JDBC) driver on:
   - All remote application servers that use the database server.
   - The server that hosts the deployment manager, if you are using Network Deployment to administer Business Process Choreographer, for example, to create a clustered process choreographer setup.
4. 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, enter:
     ```
     cd install_root\dbscripts\ProcessChoreographer\Informix
     ```
   - On UNIX and Linux systems, enter:
     ```
     cd install_root/dbscripts/ProcessChoreographer/Informix
     ```
5. If your database is not hosted on the same server as your application server:
   a. Copy the scripts for your operating system from the ProcessChoreographer directory on your application server to a suitable directory on the server that is hosting your database: For Informix on Linux and UNIX copy the following files:

```
createDatabase.sql
createSchema.sql
dropSchema.sql
createDbspace.sh
dropDbSpace.sh
```

For Informix on Windows copy the following files:

```
createDatabase.sql
createSchema.sql
dropSchema.sql
createDbspace.bat
dropDbSpace.bat
```

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

6. If you want to create a non-production database using default settings that is suitable for standalone development, evaluation, or demo purposes, enter the command:

```
dbaccess - createDatabase.sql
```

This command creates an Informix database BPEDB for the user ID that you are using. Make sure that the script output contains no errors. You can use the dropSchema.sql script to drop only the schema or the SQL command DROP DATABASE to drop the whole database.

7. If you want to create a database for a production system, you must create your database manually:

   a. Create a database, for example named BPEDB.

   b. Create the Dbspaces for your database.

On Windows systems, read the instructions in the createDbspace.bat file. Adjust the value parameters in the script to values appropriate for your environment.

On UNIX and Linux systems, read the instructions in the createDbspace.sh file. Adjust the value parameters in the script to values appropriate for your environment.

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

```
dbaccess databaseName createSchema.sql
```

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

   d. Check the script output for any errors. If you want to drop the schema, use the dropSchema.sql script.

**Results**

The database for Business Process Choreographer exists.

## Creating a Microsoft SQL Server database for Business Process Choreographer

Use this task to create a Microsoft SQL Server database for Business Process Choreographer.

**Before you begin**

You completed "Planning the BPEDB database" on page 108.

**Procedure**

1. Install a Microsoft SQL Server, on the server that hosts the database. Make sure that you select the option to create a case-sensitive instance. If you already have an SQL Server that was created with the case-insensitive option, run the rebuild master tool and change the collation settings to case-sensitive. Make sure that the server supports Unicode.

2. Make sure that the database server and that the Distributed Transaction Coordinator (DTC) are running. Make sure that the database server is configured for distributed transactions. For detailed information, refer to the documentation for Microsoft SQL Server.

3. 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, enter:
     ```
     cd install_root\dbscripts\ProcessChoreographer\SQLServer
     ```
   - On UNIX and Linux systems, enter:
     ```
     cd install_root/dbscripts/ProcessChoreographer/SQLServer
     ```

4. If your database is not on the same server as your application server:
   a. Copy the SQL scripts for your database from the SQLServer directory on your application server to a suitable directory on the server that hosts your database:
      ```
      createDatabase.sql
      createDatabaseUnicode.sql
      createSchema.sql
      createSchemaUnicode.sql
      dropSchema.sql
      ```
   b. Change to the directory where you copied the SQL scripts.

5. Perform one of the following:
   - If you want to create a non-production SQL Server database, named "BPEDB", for stand-alone development, evaluation, or demonstration purposes:
     a. Run one of the following scripts, as described in the header of the file.
        - createDatabase.sql
        - createDatabaseUnicode.sql for a Unicode database

        For example, enter:
        ```
        sqlcmd -U userID -P password -i createDatabase.sql
        ```
     b. Make sure that the script output contains no errors. If errors occur, you can drop the schema using the dropSchema.sql script.
   - If you want to create a production SQL Server database, create your database manually:
     a. Create the database, for example, named "BPEDB".
     b. To create the schema, run one of the following script, as described in the header of the file.
        - createSchema.sql
        - createSchemaUnicode.sql if you created a Unicode database

        For example, enter:
        ```
        sqlcmd -U userID -P password -i createSchema.sql
        ```
     c. Make sure that the script output contains no errors. If errors occur, you can drop the schema using the dropSchema.sql script.

**Results**

The database for Business Process Choreographer exists.

# Creating an Oracle database for Business Process Choreographer

Use this task to create an Oracle database for Business Process Choreographer.

**Before you begin**

You completed "Planning the BPEDB database" on page 108.

**Procedure**

1. Install the Oracle server on the computer that hosts the database. Be sure that you are using the 32-bit Oracle libraries that are located in the lib32 subdirectory.

2. On Linux and UNIX systems, make sure that the environment variables *ORACLE_BASE* and *ORACLE_HOME* are set for the root user.

3. Check the class path to be sure that your JDBC driver is using the ojdbc14.jar file.

4. On Linux and UNIX systems, create soft links to the following Oracle libraries in the /usr/lib directory:
   - For Oracle 10g: Link to: libclnt.so.10.1.
   - For Oracle 9i: Link to: libnnz10.so, libclnt.so.10.1, libclntsh.so.10.1, and libocijdbc10.so.

   For more detailed information on how to set up the Oracle OCI client, refer to the documentation provided by Oracle.

5. Create an Oracle database using the Database Configuration Assistant, for example, with the name BPEDB. There is no script to quickly create a default Oracle database for Business Process Choreographer. Make sure that you select the JServer option for the database. The database must be created to have a Unicode code page.

6. Start the Oracle listener by entering the command:
   ```
   lsnrctl start
   ```

7. Optional: If you do not want to customize the table space and schema, you can skip the rest of the steps in this task, in which case, the default table space and schema will be created the first time that Business Process Choreographer attempts to use the database.

8. 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, enter:
     ```
     cd install_root\dbscripts\ProcessChoreographer\Oracle
     ```
   - On UNIX and Linux systems, enter:
     ```
     cd install_root/dbscripts/ProcessChoreographer/Oracle
     ```

9. If your database is on a different server from your application server:
   a. Copy the following Oracle configuration SQL scripts from the Business Process Choreographer subdirectory on the server that is hosting the application server to an appropriate directory on the server that is hosting your database:

```
createSchema.sql
createTablespace.sql
dropSchema.sql
dropTablespace.sql
```

   b. On your database server, change to the directory where you copied the SQL scripts.

10. If you do not want to create the schema in the default instance, set the *ORACLE_SID* environment variable to the SID of the database you created in step 5 on page 191.

11. Make sure that the user who runs these scripts has at least the following database privileges: CREATE SESSION, CREATE TABLESPACE, DROP TABLESPACECREATE TABLE, and CREATE VIEW.

12. Customize a copy of the createTablespace.sql table space creation script according to what you planned in "Planning the BPEDB database" on page 108.

13. To create the table spaces, run the createTablespace.sql script. For test purposes, you can use the same location for all table spaces and pass the path as a command-line argument to the script, for example, on a Windows system, using user ID "bpeuser", password "bpepwd", database name "BPEDB", and table space path d:\mydb\ts, enter:

   ```
   sqlplus bpeuser/bpepwd@BPEDB @createTablespace.sql d:\mydb\ts
   ```

   If you want to drop the table spaces, you can use the dropTablespace.sql script.

14. Make sure that the user who will own the tables is granted sufficient quota on all the table spaces that were created in the previous step.

15. Edit the schema creation script createSchema.sql according to the instructions at the top of the file, and replace the placeholder @SCHEMA@ with the name of the schema. If the @SCHEMA@ is different from the user who runs the createSchema.sql script, ensure that this user has the following database privileges: CREATE ANY TABLE, ALTER ANY TABLE, CREATE ANY INDEX, and CREATE ANY VIEW.

16. To create the schema, run the createSchema.sql script. For example, on Windows systems, enter:

   ```
   sqlplus bpeuser/bpepwd@BPEDB @createSchema.sql
   ```

   **Results**

   The database for Business Process Choreographer exists.

# Configuring the people directory provider

Use this task to configure the Lightweight Directory Access Protocol (LDAP) or Virtual Member Manager (VMM) people directory provider that Business Process Choreographer uses to determine who can start a process or claim an activity or a task.

   **About this task**

   Each type of supported people directory service requires a corresponding people directory provider. The following people directory providers are supported:

*Table 14. Supported people directory providers*

| People directory provider | People directory provider option |
|---|---|
| Lightweight Directory Access Protocol (LDAP) directory | `LDAP people directory provider` |
| Virtual Member Manager | `VMM people directory provider` |
| Local operating system user registry | `System people directory provider` |
| WebSphere Application Server user registry | `User Registry people directory provider` |

All of these plug-ins are already installed with the default configurations. You can use the user registry and system plug-ins with the default configurations. For VMM, the default configuration is normally sufficient.

# Configuring the Virtual Member Manager people directory provider

Configure the Virtual Member Manager (VMM) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task. The default people directory provider is ready to use, and only needs to be configured if you introduce custom people assignment criteria.

**Before you begin**

You have already configured a federated repository.

**Procedure**

1. Make a copy of the standard transformation file for VMM, and give it a different name, for example, myVMMTransformation.xsl.
   - On Windows platforms, in *install_root*\ProcessChoreographer\Staff\ VMMTransformation.xsl
   - On Linux, UNIX, and i5/OS platforms in *install_root*/ ProcessChoreographer/Staff/VMMTransformation.xsl

   **Attention:** Do not make modifications to the original version of the transformation file because it might be overwritten without warning in the future when applying a service pack or fix pack.
2. If Business Process Choreographer is configured on a cluster, make your copy of the transformation file available on each WebSphere Process Server installation that hosts members of the cluster.
3. In the administrative console, click **Resources** → **People directory provider**.
4. Select the appropriate node.

| Option | Description |
|---|---|
| **For a standalone profile:** | There is only one node displayed. |
| **In a network deployment environment, where Business Process Choreographer is configured on a single server:** | Select the node that contains the server. |

| Option | Description |
| --- | --- |
| **In a network deployment environment, where Business Process Choreographer is configured on a cluster:** | You must configure the people directory provider on every node that hosts members of the cluster. Select the first node, configure the people directory provider on that node, then repeat the configuration for all other nodes that host members of the cluster. |

5. To create a new VMM people directory configuration:

   a. Click **VMM People Directory Provider**.

   b. In the **Additional Properties**, select **People directory configuration**.

   c. Click **New → Browse**, and select your copy of the Extensible Stylesheet Language (XSL) transformation file that you copied in step 1 on page 193.

   d. Click **Next**.

   e. In the **General Properties** section, enter an administrative name for the new people directory configuration.

   f. Optional: Enter a description.

   g. Enter a unique Java Naming and Directory Interface (JNDI) name that will identify this configuration to the system. For example, bpe/staff/myvmmconfiguration

      **Note:** There are no other configuration parameters

   h. Click **OK**, then click **Save**.

6. To activate the provider configuration, stop and start the server or servers where you configured the provider.

7. Optional: If you want to change the XSL transformation file:

   a. Edit the XSL file as desired.

   b. Restart the server. In a network deployment environment, the new version of the XSL file must be made available to all servers and all servers have to restarted.

8. If you have problems with any of these steps, refer to the *Troubleshooting WebSphere Process Server* PDF.

**Results**

The VMM people directory provider is configured.

# Configuring the LDAP people directory provider

Use this task to configure the Lightweight Directory Access Protocol (LDAP) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task.

**Before you begin**

You have performed the planning for LDAP described in "Planning for the people directory provider" on page 119.

**About this task**

The LDAP people directory provider configuration is initialized with a URL that points to a local LDAP server. You must change the URL later, to point to the real

LDAP server, which is normally remote to the application server. The LDAP people directory provider is configured for an LDAP server that allows anonymous access.

**Procedure**

1. Make a copy of the standard transformation file for LDAP, and give it a different name, for example, myLDAPTransformation.xsl.
   - On Windows platforms, it is in *install_root*\ProcessChoreographer\Staff\LDAPTransformation.xsl
   - On Linux, UNIX, and i5/OS platforms it is in *install_root*/ProcessChoreographer/Staff/LDAPTransformation.xsl
2. Optional: Adapt your copy of the transformation file to suit the LDAP schema for your organization. You can either perform "Adapting the LDAP transformation file" on page 196 now, or later.

   **Attention:** Do not modify the original version of the transformation file because it can be overwritten without warning in the future when applying a service or fix pack.
3. If Business Process Choreographer is configured on a cluster, make your copy of the transformation file available on each WebSphere Process Server installation that hosts members of the cluster.
4. In the administrative console, click **Resources** → **People directory provider**.
5. Select the appropriate node.

| Option | Description |
|---|---|
| **For a standalone profile:** | There is only one node displayed. |
| **In a network deployment environment, where Business Process Choreographer is configured on a single server:** | Select the node that contains the server. |
| **In a network deployment environment, where Business Process Choreographer is configured on a cluster:** | You must configure the people directory provider on every node that hosts members of the cluster. Select the first node, configure the people directory provider on that node, then repeat the configuration for all other nodes that host members of the cluster. |

6. To create a new LDAP configuration on the selected node:
   a. Click **LDAP People Directory Provider**.
   b. Under Additional Properties, click **People directory configuration**.
   c. Click **New** → **Browse**, and select your copy of the transformation file.
   d. Click **Next**.
   e. Enter an administrative name for the people directory configuration.
   f. Enter a description.
   g. Enter the Java Naming and Directory Interface (JNDI) name for human tasks to use to reference this provider. 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 planned in 2 on page 120, click the name of the property, enter a value, and click **OK**.

> **Note:** For the optional additional properties, you can set properties that are defined for JNDI, for example to enable LDAP referrals. For providerURL, you can specify a URL starting with `ldap://` or `ldaps://`.

   k.  To apply the changes, click **Save**.

7.  Optional: If you want to change the XSL transformation file:

   a.  Edit the XSL file as desired.

   b.  Restart the server. In a network deployment environment, the new version of the XSL file must be made available to all servers and all servers have to restarted.

8.  To activate the provider configuration, stop and start the server or servers where you configured the provider.

9.  If you have problems with any of these steps, refer to the *Troubleshooting WebSphere Process Server* PDF.

**Results**

Human tasks and processes can now use the people assignment services to resolve people assignment queries, and to determine which activities can be performed by which people.

## Adapting the LDAP transformation file

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

The default LDAPTransformation.xsl file maps predefined people assignment criteria 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 uses different object class and attribute names, you must change these settings in the LDAP transformation files that you use. Make a copy of the original LDAPTransformation.xsl file, as described in "Configuring the LDAP people directory provider" on page 194.

**Attention:**  Do not make modifications to the original version of the transformation file because it might be overwritten without warning in the future when applying a service pack or fix pack.

It is normally sufficient to change the settings for all people assignment criteria by editing 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 assignment criteria, as illustrated in the following examples. In each example an individual ResultObjectSpecForUserData or ResultObjectSpecForGroupUserData has to be specified reflecting the desired changes. By default, ResultObjectSpecForUserData and ResultObjectSpecForGroupUserData are shared among all people assignment criteria and are defined using the global settings.

## Example: DepartmentMembers

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

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

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

  </sldap:search>
</sldap: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:

```
<sldap:usersOfGroup>
...

  <sldap:attribute>
    <xsl:attribute name="name">uniqueMember</xsl:attribute>
    <xsl:attribute name="objectclass">groupOfUniqueNames</xsl:attribute>
    <xsl:attribute name="usage">recursive</xsl:attribute>
  </sldap:attribute>

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

</sldap:usersOfGroup>
```

## Example: GroupMembersWithoutFilteredUsers

Changing the LDAP filter operator to >=.

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

  <sldap:intermediateResult>
    <xsl:attribute name="name">filteredusers</xsl:attribute>
    <sldap:search>
      <xsl:attribute name="filter">
```

```
            <xsl:value-of select="staff:parameter[@id='FilterAttribute']"/>
             >=
            <xsl:value-of select="staff:parameter[@id='FilterValue']"/>
          </xsl:attribute>
        ...
      <sldap: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">
      (&amp;
       ...
      <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">
      (&amp;
       ...
      <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>

    <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: Users**

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

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

# Configuring people substitution

Create and activate a Virtual Member Manager (VMM) property extension repository for Business Process Choreographer to support user substitution.

**Before you begin**

You have configured WebSphere security for Federated Repositories, and if you introduce custom people assignment criteria, you have also performed "Configuring the Virtual Member Manager people directory provider" on page 193. You know whether you will use a file registry, property extension registry, or an existing LDAP schema to store the property extensions.

**Procedure**
1. Add the two attributes, "isAbsent" as a single-valued string, and "substitutes" as a multi-valued string, to the VMM definition for PersonAccount:
   a. Locate the wimxmlextension.xml file:
      - On Linux, UNIX, and i5/OS platforms, it is located in *profile_root*/config/cells/*cell_name*/wim/model
      - On Windows platforms, it is located in *profile_root*\config\cells\ *cell_name*\wim\model

b. Make a backup copy of the wimxmlextension.xml file.

c. Edit the original copy of the wimxmlextension.xml file, and make sure that it contains the following definitions, which add the two attributes that are needed for user substitution to the PersonAccount entity type:

```
<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
    dataType="STRING" multiValued="false" propertyName="isAbsent">
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
</wim:propertySchema>

<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
    dataType="STRING" multiValued="true" propertyName="substitutes">
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
</wim:propertySchema>
```

If you are using a file registry, fileRegistry.xml, skip to step 4 on page 202.

2. If you use a property extension registry to hold the substitution information, perform the following. For more information about setting up a property extension repository, see Configuring a property extension repository in a federated repository configuration.

a. Make sure that a database is available to store the property extensions.

b. Make sure that the JDBC driver class is available on the server classpath. Click **Environments** ‣ **WebSphere Variable** to check. If necessary, add the JDBC driver to the classpath by clicking **Application servers** ‣ *server_name* ‣ **Process Definition** ‣ **Java Virtual Machine** ‣ **Configuration**. For DB2, add db2jcc.jar,db2jcc_license_cu.jar and db2jcc_license_cisuz.jar to the server's classpath, and click **Apply** ‣ **Save**

c. Configure a DB2 Universal JDBC driver provider and type-4 data source for VMM using the administrative console. Set the webSphereDefaultIsolationLevel custom property for the data source to the value 2. For more information about changing the default isolation level, see Changing the default isolation level for non-CMP applications and describing how to do so using a new custom property webSphereDefaultIsolationLevel.

d. Restart the server.

e. Make a backup copy of the wimlaproperties.xml file.
   - On Linux, UNIX, and i5/OS platforms, it is located in *profile_root*/config/cells/*cell_name*/wim/model
   - On Windows platforms, it is located in *profile_root*\config\cells\ *cell_name*\wim\model

f. Edit the original copy of the wimlaproperties.xml file, and add the following definitions:

```
<wimprop:property wimPropertyName="isAbsent" dataType="String"
    valueLength="128" multiValued="false">
  <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
  </wimprop:applicableEntityName>
</wimprop:property>

<wimprop:property wimPropertyName="substitutes" dataType="String"
    valueLength="128" multiValued="true">
  <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
  </wimprop:applicableEntityName>
</wimprop:property>
```

g. Make sure that the application server (or in a network deployment environment, the deployment manager) is running. Be aware not to use the -conntype NONE option for the wsadmin utility.

h. Use the VMM administrative task setupIdMgrPropertyExtensionRepositoryTables to create the substitution

properties in the Property Extension Repository database. For more details, see Setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands. For example, using a DB2 database:

```
$AdminTask setupIdMgrPropertyExtensionRepositoryTables {
-reportSqlError true
-schemaLocation C:\w\p\etc\wim\setup
-laPropXML  C:\w\p\etc\wim\setup\wimlaproperties.xml
-databaseType db2
-dbURL jdbc:db2:
-dbDriver com.ibm.db2.jcc.DB2Driver
-dbAdminId userID
-dbAdminPassword password }
```

i. If you are using a Lightweight Directory Access Protocol (LDAP) user repository, edit the wimconfig.xml file in the config subdirectory of the win subdirectory for your cell, and add the following entries to exclude the substitution attributes from the LDAP repository:

```
<config:repositories xsi:type="config:LdapRepositoryType"
     adapterClassName="com.ibm.ws.wim.adapter.ldap.LdapAdapter"
     id="ldaprepo1" ...>
...
     <config:attributeConfiguration>
        <config:propertiesNotSupported name="isAbsent"/>
        <config:propertiesNotSupported name="substitutes"/>
     </config:attributeConfiguration>
```

j. Activate the extension property repository:

1) Using the setIdMgrPropertyExtensionRepository command. For more details, see Setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands. For example, using a DB2 database named VMMDB, a data source named VMMDS:

```
$AdminTask setIdMgrPropertyExtensionRepository {
-dataSourceName jdbc/VMMDS
-databaseType db2
-dbURL jdbc:db2:VMMDB
-dbAdminId userID
 -dbAdminPassword password
 -JDBCDriverClass com.ibm.db2.jcc.DB2Driver
-entityRetrievalLimit 10 }
```

2) Verify that the wimconfig.xml file contains an entry similar to the following:

```
<config:propertyExtensionRepository
     adapterClassName="com.ibm.ws.wim.lookaside.LookasideAdapter"
     id="LA"
     databaseType="db2"
     dataSourceName="jdbc/VMMDS"
     dbAdminId="userID"
     dbAdminPassword="{xor}PasswordXOR"
     dbURL="jdbc:db2:VMMDB"
     entityRetrievalLimit="10"
     JDBCDriverClass="com.ibm.db2.jcc.DB2Driver"/>>
```

3. If you use an LDAP schema to hold the substitution information: It may or may not already have definitions for "isAbsent" and "substitutes" (possibly with different names). Whether you have existing definitions, or you will create new ones, make sure of the following:

a. The LDAP directory must allow write operations.

b. The attribute for absence information ("isAbsent") must be of type Boolean or a String.

c. The attribute that defines who the person can substitute for ("substitutes") must be of type String, multi-valued, and permit a length up to 128 characters.

d. If your existing or chosen attribute names are not "isAbsent" and "substitutes", you must define your attribute names in the administrative console by clicking **Servers** → **Application servers** → *server_name* or **Servers**

→ **Clusters** → *cluster_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Human Task Manager** → **Configuration** → **Custom properties**, then set the desired names for the custom properties `Substitution.SubstitutesAttribute` and `Substitution.AbsenceAttribute`.

4. Restart the server.
5. Enable substitution in the Human Task Manager:
   a. Using the administrative console, click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Human Task Manager** → **Runtime**
   b. To enable substitution, select **Enable substitution**.
   c. If only administrators are allowed to perform substitution for other users, select **Restrict substitute management to administrators**.

      **Note:** This settings does not affect the ability of users to change substitution for themselves.
   d. Click **Apply**.
6. If you have problems with any of these steps, refer to the *Troubleshooting WebSphere Process Server* PDF.

**Results**

The people assignment service is configured to support user substitution for absent users.

# Creating the queue manager and queues for Business Process Choreographer

This describes how to create the WebSphere MQ queue manager and queues.

**Before you begin**

WebSphere MQ must already be installed.

**Note:** Support for WebSphere MQ is deprecated.

**About this task**

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

**Procedure**

1. Optional: If you are creating a production system, plan which disk drives the queue manager will use. Using default locations for persistent queue data and WebSphere MQ logs can have a negative impact on the performance of the queue manager. Consider changing these locations according to recommendations in the WebSphere MQ documentation.
2. If you are not creating a WebSphere MQ cluster setup, perform the following actions:
   a. Make sure that your user ID has the authority to create WebSphere MQ queues.
   b. Create the queue manager and queues: On Windows platforms, enter:

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

On UNIX and Linux platforms, enter:

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

where *queueManager* is the name of an existing queue manager, or the name to give to a new queue manager. If the named queue manager already exists, it is used to create the queues. If the queue manager does not exist, it is created and started before the default queues are created.

3. If you are creating a WebSphere cluster setup that uses a WebSphere MQ cluster, only perform Creating clustered queue managers and queues.

4. If you are creating a WebSphere cluster setup that uses a central queue manager, perform the following actions:

   a. Copy the create queues script file from the `config` subdirectory of the `ProcessChoreographer` directory on the server that hosts WebSphere Process Server machine to the server that hosts the central queue manager:

      - If your central queue manager is on a Windows workstation, copy the file: `createQueues.bat`
      - If your central queue manager is on a UNIX or Linux server, copy the file: `createQueues.sh`

   b. On the server that hosts the queue manager, make sure that WebSphere MQ is installed, and that your user ID has the authority to create WebSphere MQ queues.

   c. Create the queue manager and queues: On Windows platforms, enter:

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

      On Linux and UNIX platforms, enter:

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

      where *queueManager* is the name to give the new queue manager.

   d. Add a listener for the new queue manager by entering the command:

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

      Where *port* is the port on which the listener listens.

**Results**

The queue manager and queues exist.

# Creating clustered queue managers and queues for Business Process Choreographer

**About this task**

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.

**Procedure**

1. If your WebSphere cluster consists of UNIX nodes, perform the following actions on each node:

   a. Make sure that your user ID has the authority to create WebSphere MQ queues.

   b. Create the get and put queue managers, make them members of the WebSphere MQ cluster, and create the queues by entering the commands:

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

      where:

      *getQueueManager*
      >    The unique name to give to the get queue manager. This queue manager hosts all of the local queues.

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

      *putQueueManager*
      >    The unique name for the put queue manager. This queue manager hosts no queues, which ensures that messages are distributed across all the get queues.

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

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

      ```
      runmqsc getQueueManager
      ```

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

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

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

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

   f. Define a sender and a receiver channel for the queue manager to each repository that is not hosted on this 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 TCPIP') +
          MAXMSGL(4194304) +
          TRPTYPE(TCP) +
          MCAUSER('principal') +
          REPLACE
      ```

      For each cluster sender channel:

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

      where:

> *repositoryQueueManager*
>> The name of the queue manager hosting a repository.
>
> *clusterName*
>> The name of the WebSphere MQ cluster of which all the queue managers are a member.
>
> *repositoryIP-Address*
>> The IP address of the node where the repository queue manager resides.
>
> *port* The IP port that the repository queue manager is using.
>
> *principal*, *targetPrincipal*
>> The MCAUSER to use for the receive and send channels. For more information about this value refer to the WebSphere MQ documentation.

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

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

2. If your WebSphere cluster consists of Windows nodes, perform the following actions on each node:

   a. Make sure that your user ID has the authority to create WebSphere MQ queues.

   b. Create the "get" queue manager, make it a member of the WebSphere MQ cluster, and create the queues by entering the commands:

```
cd install_root\ProcessChoreographer\config
createQueues.bat getQueueManager clusterName putQueueManager
```

     where:

> *getQueueManager*
>> The unique name to give to the get queue manager. This queue manager hosts all of the local queues.
>
> *clusterName*
>> The name of the WebSphere MQ cluster of which all the queue managers are a member.
>
> *putQueueManager*
>> The unique name for the put queue manager. This queue manager hosts no queues, which ensures that messages are distributed across all the get queues.

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

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

```
runmqsc queueManager
```

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

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

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

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

   f. Define a sender and a receiver channel for the queue manager to each repository that is not hosted on this 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 TCPIP') +
                    MAXMSGL(4194304) +
                    TRPTYPE(TCP) +
                    MCAUSER('principal') +
                    REPLACE
```

For each cluster sender channel:

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

where:

*repositoryQueueManager*
> The name of the queue manager hosting a repository.

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

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

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

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

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

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

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

```
display chstatus(*)
```

**Results**

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

# Overview: Configuring Business Process Choreographer Explorer

**About this task**

Business Process Choreographer Explorer provides a user interface for administering business processes and handling human 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" on page 126
• "Configuring Business Process Choreographer Explorer" on page 207

## 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 configure one or more Business Process Choreographer Explorer instances on a server or cluster. It is sufficient to have a WebSphere Process Server installation with a WebSphere Process Server profile, or a WebSphere Process Server client installation – it is not necessary to have Business Process Choreographer configured on the server or cluster. The client installation is only the infrastructure that you need to connect a client to a WebSphere Process Server, but it does not contain the Business Process Choreographer Explorer. However if you have a deployment manager, then the Business Process Choreographer Explorer can be installed on the servers in the WebSphere Process Server client installation as well.

A single Business Process Choreographer Explorer can only connect to one Business Process Choreographer configuration, though it does not have to connect to a local configuration. However, you can configure multiple instances of the Business Process Choreographer Explorer on the same server or cluster, and each instance can connect to different Business Process Choreographer configurations.



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 a business process administrator, you are responsible for the smooth operation of deployed business processes. 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

You can either run a script or use the administrative console to configure Business Process Choreographer Explorer.

**Before you begin**

You have configured Business Process Choreographer.

**About this task**

One or more of the following applies:
- You have not yet installed Business Process Choreographer Explorer.

- You want to manage an existing Business Process Choreographer configuration.
- You want to add another instance of Business Process Choreographer Explorer to an already managed Business Process Choreographer configuration.

To configure Business Process Choreographer Explorer, perform one of the following:
- If you want to use a script, perform "Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer."
- If you want to use the administrative console, perform "Using the administrative console to configure the Business Process Choreographer Explorer."

**Results**

Business Process Choreographer Explorer is configured and ready to use.

**What to do next**

Start Business Process Choreographer Explorer.

## Using the administrative console to configure the Business Process Choreographer Explorer

You can use the administrative console to configure Business Process Choreographer Explorer.

**Procedure**

1. Click **Servers** → **Application servers** → *server_name* or **Servers** → **Clusters** → *cluster_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**.
2. To create a new explorer configuration, click **Add**.
3. Enter values for the following fields:
   - The **Context root** must be unique on the deployment target server or cluster.
   - **Explorer search result limit**.
   - **Managed Business Process Choreographer container**

   For more information about these fields, see "Business Process Choreographer Explorer settings" on page 138.
4. Click **Apply**. Messages are displayed indicating the progress.
5. Optional: If any problems are reported, check the SystemOut.log file.
6. Start the enterprise application named BPCExplorer_*scope*. Where *scope* identifies the server or cluster where you configure the Business Process Choreographer Explorer.

**Results**

Business Process Choreographer Explorer is configured and ready to use.

**What to do next**

Start Business Process Choreographer Explorer.

## Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer

This script file configures Business Process Choreographer Explorer and all the necessary resources on a server or cluster.

**Purpose**

This script file configures Business Process Choreographer Explorer. This script file can either be run interactively, or in batch mode.

**Location**

The clientconfig.jacl script file is located in the Business Process Choreographer `config` directory:
- On Linux, UNIX, and i5/OS platforms: In the directory *install_root/* ProcessChoreographer/config
- On Windows platforms: In the directory *install_root*\ProcessChoreographer\ config

**Runing 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 WebSphere administrative security is enabled, include the `-user` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

**Runing the script in a network deployment environment**

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 WebSphere administrative security is enabled, include the `-user` and `-password` options.
- If you are not configuring the default profile, add the `-profileName` option.

**Configuring the Business Process Choreographer Explorer non-interactively**

Change your current directory to *install_root* and perform the following:

On Linux and UNIX platforms, enter the command:
```
bin/wsadmin.sh -f ProcessChoreographer/config/clientconfig.jacl options
```

On i5/OS, enter the command:
```
bin/wsadmin -f ProcessChoreographer/config/clientconfig.jacl options
```

On Windows platforms, enter the command:
```
bin\wsadmin.bat -f ProcessChoreographer/config/clientconfig.jacl options
```

Where *options* are:
```
( [-user userName][-password password] | [-conntype NONE] )
  [-profileName profileName]
( [-node nodeName][-server serverName] )
  [-cluster clusterName]
  [-contextRootExplorer explorerContextRoot]
  [-hostName explorerVirtualHostname]
  [-precompileJSPs { yes | no }]
```

```
( ( [-remoteNode nodeName][-remoteServer serverName] )
      | [-remoteCluster clusterName] )
   [-maxListEntries maximum]
   [-explorerHost explorerURL]
```

**Note:** If you provide all the necessary parameters on the command line, you will not be prompted for them. Any required parameters that are not specified on the command line are prompted for interactively in the sequence that they are listed.

## Parameters

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

**-node** *nodeName*
   Where *nodeName* is the name of the node where Business Process Choreographer Explorer will be configured. If you do not specify this parameter, the default is the local node.

**-server** *serverName*
   Where *serverName* is the name of the server where Business Process Choreographer Explorer will 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 Explorer will be configured. This parameter is optional. Do not specify this option in a standalone server environment, nor if you specify the node and server.

**-contextRootExplorer** *contextRootExplorer*
   Where *contextRootExplorer* is the context root for the Business Process Choreographer Explorer. The context root must be unique within the WebSphere cell. The default value is /bpc.

**-hostName** *VirtualHostname*
   Where *VirtualHostname* is the virtual host where the Business Process Choreographer, and the Web service bindings of the Business Flow Manager and Human Task Manager APIs will run. The default value is default_host.

**-precompileJSPs { no | yes }**
   Determines whether Java Server Pages (JSPs) will be precompiled, or not. The default is no. Note that it is not possible to debug precompiled JSPs.

**-remoteNode** *nodeName*
   Use this parameter and remoteServer if you do not want to connect to the local Business Process Choreographer configuration. If this parameter is not specified, it defaults to the value of the -node parameter.

**-remoteServer** *serverName*
   Use this parameter and remoteNode if you do not want to connect to the local Business Process Choreographer configuration. If this parameter is not specified, it defaults to the value of the -server parameter.

**-remoteCluster** *clusterName*
   Use this parameter, if you do not want to connect to the local Business Process Choreographer configuration and you do not specify remoteNode and remoteServer. If this parameter is not specified, it defaults to the value of the -cluster parameter.

**-maxListEntries** *maximum*
   Where *maximum* is the maximum number of results that the Business Process Choreographer Explorer will return for a query. The default is 10000.

**-explorerHost** *explorerURL*

Where *explorerURL* is the URL of the Business Process Choreographer Explorer. The value of this parameter is used to link the Human Task Manager of the managed Business Process Choreographer configuration to this particular Business Process Choreographer Explorer instance. This parameter defaults to an empty string, which means that the link will not be made. You can make or change the link later using the administrative console.

### Log files

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

- clientconfig.log
- wsadmin.traceout

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

- On Linux, UNIX, and i5/OS platforms: In the directory *profile_root*/logs
- On Windows platforms: In the directory *profile_root*\logs

If you run the script in connected mode, also check the files SystemOut.log and SystemErr.log that can be found in the subdirectory of the logs directory that is named after the application server or deployment manager that the wsadmin scripting client connected to.

# Configuring the Business Process Choreographer Observer

Using the Business Process Choreographer Observer is optional, however, before you can use it, you must setup the database and install the applications.

**Before you begin**

You have performed "About Business Process Choreographer Observer" on page 127.

**About this task**

You want to configure the Business Process Choreographer Observer, with its own database.

**Procedure**
1. If you migrated from WebSphere Process Server Version 6.0.1, and used the Business Process Choreographer Observer sample, perform "Removing the Business Process Choreographer Observer Version 6.0.1 sample" on page 212.
2. Perform "Preparing a database for the Business Process Choreographer Observer" on page 213
3. Perform "Configuring the Business Process Choreographer event collector application" on page 246.
4. Perform "Configuring the Business Process Choreographer Observer application" on page 252.
5. Perform "Changing configuration parameters for the Business Process Choreographer Observer" on page 256.
6. Perform "Enabling logging for Business Process Choreographer" on page 254.
7. Perform "Verifying the Business Process Choreographer Observer" on page 265.

**Results**

The Business Process Choreographer Observer is configured and working.

You can use Business Process Choreographer Observer to generate reports, as described in "Reporting on business processes and activities" on page 357.

# Removing the Business Process Choreographer Observer Version 6.0.1 sample

Describes how to remove the Business Process Choreographer Observer sample provided with Version 6.0.1.

**About this task**

If you upgraded from WebSphere Process Server Version 6.0.1, and used the Business Process Choreographer Observer sample, you must remove the sample before you configure the latest version of the Business Process Choreographer Observer. The applications themselves are not migrated, but the database views and indexes must be dropped. The data that was collected by the sample is not migrated, and cannot be used.

**Procedure**

Drop the database views and indexes that are used by the Business Process Choreographer Observer and event collector.

1. If you use a DB2 database:
    a. Connect to the database where the observer schema is located.
    b. Locate and run the script to drop the observer sample DB2 schema:
        - On Windows platforms: *install_root*\ProcessChoreographer\sample\ observer\dropObserverSampleSchema_DB2.sql.
        - On Linux and UNIX platforms: *install_root*/ProcessChoreographer/ sample/observer/dropObserverSampleSchema_DB2.sql.

    For example, enter:

    ```
    db2 -tf dropObserverSampleSchema_DB2.sql
    ```
2. If you used a Cloudscape™ database, perform the following in a command window:
    a. Add the derby.jar and derbytools.jar files to your CLASSPATH.
        - On Windows platforms: They are located in *install_root*\derby\bin\ embedded.
        - On Linux and UNIX platforms: They are located in *install_root*/derby/ bin/embedded.
    b. Locate and run the script to drop the observer sample Cloudscape schema as described in the comments in the file:
        - On Windows platforms: *install_root*\ProcessChoreographer\sample\ observer\dropObserverSampleSchema_Cloudscape.sql.
        - On Linux and UNIX platforms: *install_root*/ProcessChoreographer/ sample/observer/dropObserverSampleSchema_Cloudscape.sql.

    For example, enter the command:

```
java -Dij.protocol=jdbc:derby:
    -Dij.database=OBSVRDB
    org.apache.derby.tools.ij
    dropObserverSampleSchema_Cloudscape.sql
```

**Results**

The Business Process Choreographer Observer sample has been removed.

# Preparing a database for the Business Process Choreographer Observer

Perform the actions for your database.

## Preparing a DB2 Universal database for the Business Process Choreographer Observer

You can either use scripts or an interactive tool to prepare the database.

**Using SQL scripts to prepare a DB2 Universal database for the Business Process Choreographer Observer:**

This describes how to use the createTablespace_Observer.sql and createSchema_Observer.sql scripts to prepare a DB2 Universal database on Linux, UNIX, and Windows platforms.

**About this task**

Your database must already exist. You can either use an existing database, or create a new one according to the database documentation. To perform this task, you must have administration rights for the destination database.

**Procedure**

1. Change to the Business Process Choreographer subdirectory where the configuration scripts for your database are located.
   - On Linux and UNIX platforms, change to the directory *install_root*/dbscripts/ProcessChoreographer/DB2.
   - On Windows platforms, change to the directory *install_root*\dbscripts\ProcessChoreographer\DB2.
2. Copy all *Observer.sql script files to your database server.
3. On the database server, change to the directory where you copied the script files.
4. Create the table space:
   a. Edit the `createTablespace_Observer.sql` script file according to the instruction at the top of the file.
   b. Run the table space creation script file, enter the command:
      `db2 -tf createTablespace_Observer.sql`
   c. Make sure that the script output contains no errors. If errors occur, you can drop the table space using the `dropTablespace_Observer.sql` script file.
5. Create the schema (tables, indexes, and views).
   a. Edit the `createSchema_Observer.sql` script file according to the instructions at the top of the file.
   b. In the DB2 command-line processor, enter the command:
      `db2 -tf createSchema_Observer.sql`

c.  Make sure that the script output contains no errors. If you want to drop the schema, use the `dropSchema_Observer.sql` script file.

6.  If you want to use the Java implementation instead of the SQL implementation of the Business Process Choreographer Observer UDFs perform "Selecting between Java and SQL user-defined functions" on page 240.

7.  Use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

### Related concepts

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

### Related tasks

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

**Using the setupEventCollector tool to prepare a DB2 Universal database for the Business Process Choreographer Observer:**

This describes how to use an interactive menu-driven tool and createTablespace_Observer.sql script to prepare a DB2 Universal database on Linux, UNIX, and Windows platforms.

**Before you begin**

Your database must already exist, and you must have administrative rights for it.

**Procedure**

1.  If you use a type 2 JDBC connection:
    a.  Prepare the command line environment:
        - On Linux and UNIX platforms, run the db2profile for your DB2 instance.
        - On Windows, open a DB2 command window.
    b.  If your database is remote, catalog the database on a local DB2 instance.
2.  Create the table space:
    a.  Change to the Business Process Choreographer subdirectory where the SQL scripts for your database are located.
        - On Linux and UNIX platforms, change to the directory *install_root*/dbscripts/ProcessChoreographer/DB2.
        - On Windows platforms, change to the directory *install_root*\dbscripts\ProcessChoreographer\DB2.
    b.  Make a copy of the createTablespace_Observer.sql script file.
    c.  Edit your copy of the createTablespace_Observer.sql script file according to the instruction at the top of the file.

d. Run the table space creation script file, enter the command:

```
db2 -tf createTablespace_Observer.sql
```

e. Make sure that the script output contains no errors. If errors occur, you can drop the table space using the dropTablespace_Observer.sql script file.

3. If you are using a user ID that is not a database administrator, make sure that it has the following permissions:

```
GRANT CREATETAB, CONNECT, CREATE_EXTERNAL_ROUTINE ON DATABASE
     TO USER user_name;
GRANT USE OF TABLESPACE tablespace_name TO USER user_name;
```

where *user_name* is the user ID, and *tablespace_name* is a list of all observer table space names, as can be found in the script createTablespace_Observer.sql.

4. Change to the Business Process Choreographer directory where the configuration scripts are located.

On Linux and UNIX platforms, enter:

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

On Windows platforms, enter:

```
cd install_root\ProcessChoreographer\config
```

5. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261.

6. Prepare the database:

a. When you see:

```
1) Prepare a database for the Event Collector and Observer
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

Select option 1 to prepare a database for the event collector and observer applications.

b. When you see:

```
Prepare a database for the WebSphere Business Process Choreographer
Event Collector and Observer

Select the type of your database provider:

   c) Derby
   d) DB2 Universal
   i) DB2 iSeries
   7) DB2 V7 on z/OS
   8) DB2 V8 on z/OS
   o) Oracle

   0) Exit Menu
```

Enter d to select DB2 Universal.

c. The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

- If you do not want to delay running the SQL, enter n.

Chapter 4. Configuring Business Process Choreographer **215**

- If you want to delay running the SQL, enter y. You will see:

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

  – If you want the values that you enter to be checked within the database, enter y.

  – Otherwise enter n.

Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

d. If you see:

```
Specify the JDBC driver type to be used:

  2) Connect using type 2 (using a native database client)
  4) Connect using type 4 (directly via JDBC)
```

Specify the JDBC driver type:

- If you are using a native database client, enter 2 .

- Otherwise, enter 4 to select the type 4 JDBC driver.

e. If you see one of the following prompts:

```
Specify the name of your database:  [BPEDB]
Specify the name of database in local catalog:  [BPEDB]
```

Either the database name, or an alias.

**Note:** The default value, BPEDB, is the same database that is used by Business Process Choreographer. For a high-performance system, you should use a different database.

f. If you see:

```
Specify the hostname of the database server:  [localhost]
```

Enter the host name or IP address of your database server.

g. If you see:

```
Specify the port where the database server is listening:  [50000]
```

Enter the port number for your database server.

h.

```
Specify the directory of your JDBC driver:  [D:\opt\SQLLIB\java]
```

Enter the directory where the db2jcc.jar and db2jcc_license_cu.jar JAR files for the JDBC driver reside.

i. If you see:

```
Specify userid to connect to the database 'database_name' [db2admin] :
Specify the password for userid 'user_ID' :
```

Enter the user ID and password to connect to the database. The following is displayed:

```
 Trying to connect to database 'database_name', using user 'user_ID'
Connected to 'database_name'
```

j. If you see:

```
Specify the database schema to be used. [user_ID] :
```

Enter the database schema (the collection name) to be used for the database objects. If you enter a space character or leave the field empty, the schema of the user ID is used.

k. When you see:

```
Choose the implementation of the Observer user-defined functions.

  Note: The Java UDFs are more precise, but they require a jar file
  installed to the database.
  Visit the Observer documentation for details.

  1) Java
  2) SQL

0) Exit Menu
```

- If you want to use the more precise Java-based user-defined functions (UDFs), which requires that a JAR file is installed in the database, enter 1.
- If you want to use the less precise SQL-based UDFs, enter 2.

The following is displayed:

```
Checking for required tablespace(es) ['OBSVRTS']
All required tablespaces were found.
Loading the jar file 'B:\W\P\lib\bpcodbutil.jar' into the database.
The jar file 'B:\W\P\lib\bpcodbutil.jar' was successfully installed.

The setup of the database completed successfully.
```

7. Use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

### Related concepts

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

### Related tasks

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

## Preparing a DB2 for iSeries database for the Business Process Choreographer Observer

You can either use scripts or an interactive tool to prepare the database.

**Using an SQL script to prepare a DB2 for iSeries database for the Business Process Choreographer Observer:**

This describes how to use the createSchema_Observer.sql script in an i5/OS qshell environment to prepare a DB2 for iSeries database.

**Before you begin**

Your collection must already exist. You can either use an existing collection, or create a new one according to the database documentation.

**Procedure**

1. In a qshell environment, change to the Business Process Choreographer subdirectory where the configuration scripts for your database are located. Enter:

   `cd install_root/dbscripts/ProcessChoreographer/DB2iSeries`

2. Copy all *Observer.sql script files to your database server.

3. On the database server, change to the directory where you copied the script files.

4. Create the schema (tables, indexes, and views).

   a. Edit the createSchema_Observer.sql script file according to the instruction at the top of the file.

   b. Either in the DB2 command-line processor, or in a qshell, enter the command:

      `db2 -tf createSchema_Observer.sql`

   c. Make sure that the script output contains no errors. If you want to drop the schema, use the `dropSchema_Observer.sql` script file.

5. If you want to use the Java implementation of the necessary user-defined functions (UDFs), perform "Selecting between Java and SQL user-defined functions" on page 240.

6. Use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

**Using the setupEventCollector tool to prepare a DB2 for iSeries database for the Business Process Choreographer Observer:**

This describes how to use an interactive menu-driven tool to prepare a DB2 for iSeries database from within an i5/OS qshell environment.

**About this task**

You can either use an existing collection, or create a new one according to the database documentation.

**Procedure**

1. Start a qshell environment.

2. Change to the Business Process Choreographer directory where the configuration scripts are located. Enter:

   cd *install_root*/ProcessChoreographer/config

3. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261.

4. Prepare the database:

   a. When you see:

      ```
      1) Prepare a database for the Event Collector and Observer
      2) Install the Event Collector application
      3) Remove the Event Collector application and related objects
      4) Change configuration settings of an installed Event Collector
      5) Drop the database schema of the Event Collector and Observer
      6) Administer Observer related user-defined functions

      0) Exit Menu
      ```

      Select option 1 to prepare a database for the event collector and observer applications.

   b. When you see:

      ```
      Prepare a database for the WebSphere Business Process Choreographer
      Event Collector and Observer

      Select the type of your database provider:

         c) Derby
         d) DB2 Universal
         i) DB2 iSeries
         7) DB2 V7 on z/OS
         8) DB2 V8 on z/OS
         o) Oracle

         0) Exit Menu
      ```

      Enter i to select DB2 for iSeries.

   c. The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

      ```
      Do you want to create an SQL file only (delay database preparation)?
        y) yes
        n) no
      ```

      • If you do not want to delay running the SQL, enter n.

      • If you want to delay running the SQL, enter y. You will see:

        ```
        Even if you want to delay the configuration,
        your entered values can be checked within the database.
        Do you want to perform these checks?
          y) yes
          n) no
        ```

        – If you want the values that you enter to be checked within the database, enter y.

        – Otherwise enter n.

      Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

   d. If you see:

```
Specify the JDBC driver to be used:

    1) Connect using the IBM Toolbox for Java JDBC driver
    2) Connect using the native JDBC driver
  Your selection: [2]
```

- If you are configuring a remote database, enter 1 to select the IBM Toolbox for Java JDBC driver.
- If you are configuring the local database, enter 2 to select the native JDBC driver.

e. If you see:

```
Specify the name of database in local catalog:  [*LOCAL]
```

or

```
Specify the name of your database:  [*SYSBAS]
```

Enter the service identifier, or accept the default.

f. If you see:

```
Specify the hostname of the database server:  [localhost]
```

Enter the host name or IP address of your database server.

g. If you see:

```
Specify the directory of your JDBC driver:
[/QIBM/ProdData/HTTP/Public/jt400/lib]
```

Enter the directory where the JDBC driver JAR files reside.

- For the native driver (db2_classes.zip), this is typically /QIBM/ProdData/Java400/ext.
- For the toolbox driver (jt400.jar), this is typically /QIBM/ProdData/HTTP/Public/jt400/lib

h. If you see:

```
Specify userid to connect to the database 'database_name' [db2admin] :
Specify the password for userid 'user_ID' :
```

Enter the user ID and password to connect to the database.

i. If you see:

```
Specify the database schema to be used. [user_ID] :
```

Enter the database schema (the collection name) to be used for the database objects. You must specify a schema that already exists. If you enter a space character or leave the field empty, the schema of the user ID is used.

j. If you see:

```
Note: The Java UDFs are more precise, but they require a jar file
installed to the database.
Visit the Observer documentation for details.

  1) Java
  2) SQL

  0) Exit Menu
```

- If you want to use the more precise Java-based user-defined functions (UDFs), which requires that a JAR file is installed in the database, enter 1.
- If you want to use the less precise SQL-based UDFs, enter 2.

k. When the database has been successfully prepared, the following is displayed:

```
     The setup of the database completed successfully.
```

5. If you used a separate database (not BPEDB), then use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

**Using the setupEventCollector tool to prepare a DB2 for iSeries database from a remote system:**

This describes how to use an interactive menu-driven tool to prepare a DB2 for iSeries database for the Business Process Choreographer Observer from a remote Linux, Windows, or UNIX system.

**About this task**

You can either use an existing collection, or create a new one according to the database documentation. The collection to be used must already exist.

**Procedure**

1. To prepare the database remotely, you need to download the IBM Toolbox JDBC driver to connect to your iSeries machine. After downloading, note the location of the jar file jt400.jar.
2. Start a command line environment.
3. Change to the Business Process Choreographer directory where the configuration scripts are located.
   - On Windows platforms, enter:
     ```
     cd install_root\ProcessChoreographer\config
     ```
   - On Linux and UNIX platforms, enter:
     ```
     cd install_root/ProcessChoreographer/config
     ```
4. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261.
5. Prepare the database:
   a. When you see:
      ```
      1) Prepare a database for the Event Collector and Observer
      2) Install the Event Collector application
      3) Remove the Event Collector application and related objects
      4) Change configuration settings of an installed Event Collector
      ```

```
5) Drop the database schema of the Event Collector and Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

Select option 1 to prepare a database for the event collector and observer applications.

b.  When you see:

```
Prepare a database for the WebSphere Business Process Choreographer
Event Collector and Observer

Select the type of your database provider:

    c) Derby
    d) DB2 Universal
    i) DB2 iSeries
    7) DB2 V7 on z/OS
    8) DB2 V8 on z/OS
    o) Oracle

    0) Exit Menu
```

Enter i to select DB2 for iSeries.

c.  The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

- If you do not want to delay running the SQL, enter n.

- If you want to delay running the SQL, enter y. You will see:

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

    – If you want the values that you enter to be checked within the database, enter y.

    – Otherwise enter n.

    Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

d.  If you see:

```
Specify the name of your database:  [*SYSBAS]
```

Enter the service identifier, or accept the default.

e.  If you see:

```
Specify the hostname of the database server:  [localhost]
```

Enter the host name or IP address of your database server.

f.  If you see:

```
Specify the directory of your JDBC driver:
[/QIBM/ProdData/HTTP/Public/jt400/lib]
```

Enter the directory where you downloaded the JDBC driver file jt400.jar.

g.  If you see:

```
    Specify userid to connect to the database 'database_name' [db2admin] :
    Specify the password for userid 'user_ID' :
```

Enter the user ID and password to connect to the database.

h. If you see:

```
    Specify the database schema to be used. [user_ID] :
```

Enter the database schema (the collection name) to be used for the database objects. You must specify a schema that already exists. If you enter a space character or leave the field empty, the schema of the user ID is used.

i. If you see:

```
    Note: The Java UDFs are more precise, but they require a
    jar file installed to the database.
    Visit the Observer documentation for details.

     1) Java
     2) SQL

     0) Exit Menu
```

- If you want to use the more precise Java-based user-defined functions (UDFs), which requires that a JAR file is installed in the database, enter 1.
- If you want to use the less precise SQL-based UDFs, enter 2.

j. When the database has been successfully prepared, the following is displayed:

```
    The setup of the database completed successfully.
```

6. If you used a separate database (not BPEDB), then use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

## Preparing a DB2 for z/OS database for the Business Process Choreographer Observer

You can prepare the database remotely or within the UNIX System Services.

**Creating a DB2 for z/OS database for the Business Process Choreographer Observer in USS:**

This describes how to use an interactive menu-driven tool, and the createTablespace_Observer.sql script in UNIX System Services (USS) on a z/OS machine, to create a DB2 for z/OS database.

**Procedure**

1. Prepare the DB2 environment:
   a. Log on to 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. Determine the location name of the subsystem. To find out the location name, either check on the DB2 Systems panel or select the DB2 administration menu option **Execute SQL statements** for your subsystem, and enter the following SQL query:

   ```
   select current server from sysibm.sysdummy1
   ```

   e. Create a storage group and note the name, for example OBSVRSG.
   f. If you want to use a new database, create a new database, for example, named OBSVRDB. If you want, you can reuse an existing database and storage group, for example, the Business Process Choreographer database, BPEDB.
   g. Decide which schema qualifier to use (_SQLID).
   h. Decide which user ID, *user_ID*, will be used to set up the database. This is not the user ID used to access the database at runtime.
   i. Ensure that the user ID has the following rights to access the database and storage group:
      - Permission to use the storage group.
      - Permission to use the database OBSVRDB.
      - Permission to create table spaces within the database OBSVRDB.
      - Permission to create tables within the database OBSVRDB.
   j. If you intend to use the Java implementation of the Business Process Choreographer user-defined functions (UDFs), ensure that the user ID also has the following rights:
      - Permission to perform a select on SYSIBM.SYSJAROBJECTS.
      - Permission to execute the following stored procedures for the schema SQLJ:
        – INSTALL_JAR
        – REMOVE_JAR
        – REPLACE_JAR
        – DB2_INSTALL_JAR
        – DB2_REMOVE_JAR
        – DB2_REPLACE_JAR
      - Permission to execute packages belonging to the collection DSNJAR.
   k. If you intend to use the Java implementation of the Business Process Choreographer user-defined functions (UDFs), prepare the DB2 environment to run Java user defined functions and interpreted Java routines. Perform the following:
      1) Enabling the DB2-supplied stored procedures and defining the tables used by the DB2 Universal JDBC Driver
      2) Setting up the environment for interpreted Java routines

Note the name of the WLM application environment created during this procedure.

2. Log on to the USS.

3. Create the table space:

   a. Change to the directory where the Business Process Choreographer Observer database scripts for your database system are located. Depending on your DB2 version, enter one of the following commands:

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

   b. Edit the `createTablespace_Observer.sql` script. Replace @STOGRP@ with the storage group name and replace @DBNAME@ with the database name (not the subsystem name).

      **Note:** The SQL files are delivered in ASCII format. Depending on the tool that you use to view, edit or run this file, you might need to convert the file to EBCDIC. To convert the file to EBCDIC, enter the following command:

      ```
      iconv -t IBM-1047 -f ISO8859-1 createTablespace_Observer.sql > createTablespace_Observer.sql_EBCDIC.sql
      ```

      To convert it back to ASCII enter the command:

      ```
      iconv -t ISO8859-1 -f IBM-1047 createTablespace_Observer_EBCDIC.sql > createTablespace_Observer_ASCII.sql
      ```

   c. Make sure that you are connected to your database, and run your customized version of the `createTablespace_Observer.sql` script.

4. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261.

5. Prepare the database: When you see:

   ```
   1) Prepare a database for the Event Collector and Observer
   2) Install the Event Collector application
   3) Remove the Event Collector application and related objects
   4) Change configuration settings of an installed Event Collector
   5) Drop the database schema of the Event Collector and Observer
   6) Administer Observer related user-defined functions

   0) Exit Menu
   ```

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

   b. Enter 7 or 8 to select your DB2 on z/OS version number.

   c. The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

      ```
      Do you want to create an SQL file only (delay database preparation)?
        y) yes
        n) no
      ```

      • If you do not want to delay running the SQL, enter n.

      • If you want to delay running the SQL, enter y. You will see:

        ```
        Even if you want to delay the configuration,
        your entered values can be checked within the database.
        Do you want to perform these checks?
          y) yes
          n) no
        ```

        – If you want the values that you enter to be checked within the database, enter y.

        – Otherwise enter n.

      Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

   d. If you see:

```
Specify the database location name:
   (as returned by SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1):
```

Enter the location name of your database. This is the value you noted in step 1d on page 224.

e. If you see:
```
Specify the name of the database as known by the susbsystem [subsystem]
```

Enter the name that your database has in the subsystem on the z/OS host. This is the value you noted in step 1f on page 224.

f. If you see:
```
Specify the hostname of the z/OS DB2 database server: [localhost]
```

Enter the hostname of your database server.

g. If you see:
```
Specify the port where the database subsystem is listening:
```

Enter the port number that is used by the database subsystem. This is the value you noted in step 1c on page 224.

h. If you see:
```
Specify userid to connect to the database 'database_alias' [db2admin] :
```

Enter the user ID to connect to the database . This is the user ID, *user_ID*, described in step 1h on page 224.

i. If you see:
```
Specify the password for userid 'user_ID' :
```

Enter the password for the user ID.

j. If you see:
```
Trying to connect to database 'database_alias', using user 'user_ID'
Connected to 'database_alias'
Specify the database schema to be used. [user_ID] :
```

Enter the database schema to be used for the database objects, or press Enter to accept the default, which is the user ID that is used to connect to the database. This is the schema qualifier _SQLID.

k. If you see:
```
Note: The Java UDFs are more precise, but they require a jar file
installed to the database.
Visit the Observer documentation for details.

   1) Java
   2) SQL

   0) Exit Menu
```
- If you want to use the more precise Java-based user-defined functions (UDFs), which requires that a JAR file is installed in the database, enter 1.
- If you want to use the less precise SQL-based UDFs, enter 2.

l. If you see:
```
Specify the DB2 storage group name to be used. [OBSVRSG] :
```

Enter the storage group name from step 1e on page 224, or press Enter to accept the default value.

m. If you see:

```
Specify the WLM environment name where the UDF should run. [] :
```

Enter the WLM environment that you noted in step 1k on page 224.

n. After checking for the required table spaces and loading a JAR file into the database, success is indicated by the following:

```
The setup of the database completed successfully.
```

6. Using the administrative console, create an XA data source that points to the database.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

**Creating a DB2 for z/OS database for the Business Process Choreographer Observer from a remote system:**

This describes how to use an interactive menu-driven tool, and the createTablespace_Observer.sql script on a Linux, UNIX, or Windows system to create a DB2 for z/OS database.

**Before you begin**

You must have already installed WebSphere Process Server on a Linux, UNIX, or Windows server.

**Procedure**

1. On the z/OS server that hosts the database:
   a. Log on to 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. Create a storage group and note the name, for example OBSVRSG.
   e. If you want to use a new database, create a new database, for example, named OBSVRDB. If you want, you can reuse an existing database and storage group, for example, the Business Process Choreographer database, BPEDB.
   f. Decide which schema qualifier to use (_SQLID).
   g. Decide which user ID, *user_ID*, will be used to set up the database. This is not the user ID used to access the database at runtime.

h. Ensure that the user ID has the following rights to access the database and storage group:
   - Permission to use the storage group.
   - Permission to use the database OBSVRDB.
   - Permission to create table spaces within the database OBSVRDB.
   - Permission to create tables within the database OBSVRDB.

i. If you intend to use the Java implementation of the Business Process Choreographer user-defined functions (UDFs), ensure that the user ID also has the following rights:
   - Permission to perform a select on SYSIBM.SYSJAROBJECTS.
   - Permission to execute the following stored procedures for the schema SQLJ:
     - INSTALL_JAR
     - REMOVE_JAR
     - REPLACE_JAR
     - DB2_INSTALL_JAR
     - DB2_REMOVE_JAR
     - DB2_REPLACE_JAR
   - Permission to execute packages belonging to the collection DSNJAR.

j. If you intend to use the Java implementation of the Business Process Choreographer user-defined functions (UDFs), prepare the DB2 environment to run Java user defined functions and interpreted Java routines. Perform the following:
   1) Enabling the DB2-supplied stored procedures and defining the tables used by the DB2 Universal JDBC Driver
   2) Setting up the environment for interpreted Java routines

   Note the name of the WLM application environment created during this procedure.

2. On the server that hosts the WebSphere Process Server:
   a. Install a suitable DB2 client.

      **Note:** If you plan to use a native DB2 client to connect to the remote database (using a type 2 JDBC connection) make sure that you have DB2 Connect Gateway installed. DB2 Connect Gateway is part of the DB2 UDB ESE package, but you can also install it separately.

   b. If you are using a native DB2 client, catalog the remote database and verify that you can establish a connection to it. Enter the following commands in a DB2 command line window:

```
catalog tcpip node zosnode remote host_name server IP_port ostype mvs
catalog database location as database_alias at node zosnode
        authentication dcs
catalog dcs database database_alias parms ',,INTERRUPT_ENABLED'
```

   where

   *zosnode*
       is a local alias for the remote z/OS node.

   *host_name*
       is either the TCP/IP address or alias of the remote z/OS machine.

*IP_port*
is the port number where the DB2 subsystem is listening.

*database_alias*
is the local alias to access the remote database.

*location*
is the remote DB2 location name. To find out the location name, log on to TSO and enter the following SQL query on the selected subsystem using one of the available query tools.

```
select current server from sysibm.sysdummy1
```

To verify that you can connect to the remote system, enter:

```
db2 connect to database_alias user userid using password
```

c.  Change to the directory where the Business Process Choreographer Observer database scripts for your database system are located:

- On Linux and UNIX platforms, depending on your DB2 version, enter one of the following commands:

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

- On Windows platforms, depending on your DB2 version, enter one of the following commands:

```
cd install_root\dbscripts\ProcessChoreographer\DB2zOSV7
cd install_root\dbscripts\ProcessChoreographer\DB2zOSV8
```

d.  Edit the `createTablespace_Observer.sql` script. Replace @STOGRP@ with the storage group name and replace @DBNAME@ with the database name (not the subsystem name).

e.  Run your customized version of the `createTablespace_Observer.sql` script. If you want to drop the table space, use the `dropTablespace_Observer.sql` script.

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

On Linux and UNIX platforms, enter:

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

On Windows platforms, enter:

```
cd install_root\ProcessChoreographer\config
```

g.  Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261.

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

i.  Enter 7 or 8 to select your DB2 on z/OS version number.

j.  The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

- If you do not want to delay running the SQL, enter n.

- If you want to delay running the SQL, enter y. You will see:

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

- If you want the values that you enter to be checked within the database, enter y.
- Otherwise enter n.

Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

k. If you see:

```
Specify the JDBC driver type to be used:

2) Connect using type 2 (using a native database client)
4) Connect using type 4 (directly via JDBC)
```

Specify the JDBC driver type:
- If you are using a native database client, enter 2 .
- Otherwise, enter 4 to select the type 4 JDBC driver.

l. If you see:

```
Specify the name of database in local catalog:  [BPEDB]
```

Enter the name of your database as it is cataloged on the local DB2 client, this is the value that you used for *database_alias* in step 2b on page 228.

m. If you see:

```
Specify the location name/connection target: []
```

Enter the location name of the subsystem to connect to.

**Note:** To determine the location name, log on with a SQL processor and execute the following SQL statement:

```
SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1
```

n. If you see:

```
Specify the name of the database as known by the subsystem: [OBSVRDB]
```

Enter the name by which the database is known within the subsystem on the z/OS host.

o. If you see:

```
Specify the hostname of the database server:  [localhost]
Specify the port where the database server is listening:  [446]
```

Enter the host name and port number used by your z/OS database server.

p. If you see:

```
Specify the directory of your JDBC driver:  []
```

Enter the directory where the db2jcc.jar and db2jcc_license_cisuz.jar JAR files for the DB2 JDBC driver reside.

q. If you see:

```
Specify userid to connect to the database 'database_name' [db2admin] :
Specify the password for userid 'user_ID' :
```

Enter the user ID and password to connect to the database. This is the user ID, *user_ID*, described in step 1g on page 227.

r. If you see:

```
Specify the database schema to be used. [user_ID] :
```

Enter the name of the database schema to use for the database objects.

s. If you see:

```
Note: The Java UDFs are more precise, but they require a jar file
installed to the database.
Visit the Observer documentation for details.

  1) Java
  2) SQL

  0) Exit Menu
```

- If you want to use the more precise Java-based user-defined functions (UDFs), which requires that a JAR file is installed in the database, enter 1.
- If you want to use the less precise SQL-based UDFs, enter 2.

t. If you see:

```
Specify the DB2 storage group name to be used. [OBSVRSG] :
```

Enter the storage group name from step 1d on page 227.

u. If you see:

```
Specify the WLM environment name where the UDF should run. [] :
```

Enter the WLM environment that you noted in step 1j on page 228. After checking for the required table spaces and loading a JAR file into the database, success is indicated by the following:

```
The setup of the database completed successfully.
```

3. Using the administrative console, create an XA data source that points to the database.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

## Preparing a Derby database for the Business Process Choreographer Observer

You can either use scripts or an interactive tool to prepare the database.

**Using an SQL script to prepare a Derby database for the Business Process Choreographer Observer:**

This describes how to use the createSchema_Observer.sql script to prepare a Derby database.

**About this task**

You must create the schema for the Business Process Choreographer Observer database. You can either create it in an existing database, or have the script file create a new database for you.

**Procedure**

1. Change to the Business Process Choreographer subdirectory where the configuration scripts for your database are located.
   - On Linux, UNIX, and i5/OS platforms, change to the directory *install_root*/dbscripts/ProcessChoreographer/Derby.
   - On Windows platforms, change to the directory *install_root*\dbscripts\ ProcessChoreographer\Derby.
2. Perform the following:
   a. In a derby network server environment, copy the *Observer.sql scripts to the network server. Also copy the JAR file bpcodbutil.jar, from the lib subdirectory of the *install_root* directory to the same directory on your database server.
   b. In a text editor, read the instructions in the header of the script file `createSchema_Observer.sql`. If you want to create a new database, append `;create=true` to the database name. For example, if your database name is OBSVRDB, replace the parameter `-Dij.database=OBSVRDB` with `-Dij.database=OBSVRDB;create=true`

   **Note:** On Windows platforms, avoid using the Notepad editor, because it does not display the file in a readable format.
   c. If you connect to an existing database using the embedded Derby driver, stop the server and any other applications that use the database.
   d. 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.
   e. In case of errors, you can run the script file `dropSchema_Observer.sql` to drop the schema.
3. Use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Using the setupEventCollector tool to prepare a Derby database for the Business Process Choreographer Observer:**

This describes how to use an interactive menu-driven tool, setupEventcollector, to prepare a Derby database on any supported platform.

**Procedure**

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

   On Linux, UNIX, and i5/OS platforms, enter:

   `cd install_root/ProcessChoreographer/config`

   On Windows platforms, enter:

   `cd install_root\ProcessChoreographer\config`

2. If you connect to an existing database using the embedded Derby driver, stop the server and any other applications that use the database. Plan to use the `-conntype none` when starting the tool.

3. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261.

4. When you see:

```
1) Prepare a database for the Event Collector and Observer
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

   Select option 1 to prepare a database for the event collector application. The following menu is displayed:

```
Prepare a database for the WebSphere Business Process Choreographer
Event Collector and Observer

Select the type of your database provider:

    c) Derby
    d) DB2 Universal
    i) DB2 iSeries
    7) DB2 V7 on z/OS
    8) DB2 V8 on z/OS
    o) Oracle

    0) Exit Menu
```

5. Enter c to select Derby.

6. The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

   • If you do not want to delay running the SQL, enter n.

   • If you want to delay running the SQL, enter y. You will see:

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

   – If you want the values that you enter to be checked within the database, enter y.

   – Otherwise enter n.

   Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

7. If you see:

```
Specify the JDBC driver type to be used:

  1) Connect using the embedded JDBC driver
  2) Connect using the network JDBC driver

Your selection: [1]
```

   • To connect using the embedded JDBC driver, enter 1.

> **Important:** While configuring the database using this driver, make sure that no other application (including the WebSphere Process Server) is connected to the database.

- To use the network JDBC driver, enter 2.

8. When you see: Specify the name of your database [*database_name*]

   Enter the fully qualified path to the database.

   **Note:** The default value, `...\BPEDB`, is the same database that is used by the Business Process Choreographer. For better performance, use a separate database.

9. If you see:
   ```
   Specify the database schema to be used. [APP] :
   ```

   Enter the database schema name to be used for the database objects. If you enter a space character or leave the field empty, the default schema, APP, is used.

10. If you see:
    ```
    Specify the hostname of the database server:  [localhost]
    Specify the port where the database server is listening:  [1527]
    ```

    Enter the host name and port number for your Derby network server.

11. If you see:
    ```
    Specify the directory of your JDBC driver:  [B:\w\p\derby\lib]
    ```
    - For the embedded JDBC driver, enter the directory where the derby.jar file is located.
    - For the network JDBC driver, enter the directory where the derbyclient.jar is located.

12. If you see:
    ```
    Specify userid to connect to the database database_name:  []
    ```
    - If the server requires authentication, enter a user ID that is authorized to connect to your Derby network server.
    - Otherwise, entering no value results in the user ID, dummy, being used. This is because the Derby JDBC driver always requires a user ID to connect to a network server.

13. If you see:
    ```
    The application server must be stopped to update a Derby /
    Cloudscape database.
    This must be done outside wsadmin using 'stopServer server_name'.
    After the server is stopped, come back to this prompt and enter
    'c' to continue.
    Please stop the server 'server_name' now.
    Press 'c' to continue, 'a' to abort:
    ```
    a. Stop the server, outside wsadmin, using the command:

       stopServer *server_name*

    b. If you stopped the server, press c to continue. Otherwise, press a to return to the main menu shown in step 4 on page 233.

14. If you see:
    ```
    Specify the database schema to be used. [APP] :
    ```

    Enter the name of schema to be used for the database objects, or press Enter to use the default.

15. Make sure that you see the following message, which confirms that the database was prepared successfully:

    ```
    The setup of the database completed successfully.
    ```

16. If you see:

    ```
    Restart the server now using 'startServer server_name'.
    After the server is up again, come back to this prompt and enter
    'c' to continue.
    Press 'c' to continue, 'a' to abort:
    ```

    a. Start the server, using the command:

       ```
       startServer server_name
       ```

    b. Wait until the server has started, then back at this prompt, press c to continue. Otherwise, press a to return to the main menu shown in step 4 on page 233.

    Success is indicated by the message:

    ```
    WASX7074I: Reconnect of SOAP connector to host localhost completed.
    ```

17. Use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

## Preparing an Oracle database for the Business Process Choreographer Observer

You can either use scripts or an interactive tool to prepare the database.

**Using SQL scripts to prepare an Oracle database for the Business Process Choreographer Observer:**

This describes how to use the createTablespace_Observer.sql and createSchema_Observer.sql scripts to prepare an Oracle database.

**About this task**

Your database must already exist. You can either use an existing database, or create a new one according to the database documentation.

**Procedure**

1. Change to the Business Process Choreographer subdirectory where the configuration scripts for your database are located.
   - On Linux, UNIX, and i5/OS platforms, change to the directory *install_root*/dbscripts/ProcessChoreographer/Oracle.
   - On Windows platforms, change to the directory *install_root*\dbscripts\ ProcessChoreographer\Oracle.
2. Copy all *Observer.sql script files to your database server.
3. If you want to use the Java-based Business Process Choreographer user-defined functions:
   a. Also copy the JAR file bpcodbutil.jar, from the lib subdirectory of the *install_root* directory to the directory that contains the SQL script file.
   b. Install the JAR file that contains UDFs for Business Process Choreographer Observer.

1) Log on to your database server as a user with Oracle administration rights, and change to the directory where the JAR file `bpcodbutil.jar` is located:

   - If your database is on the same server as the application server, change to the `lib` subdirectory of the *install_root* directory.
   - If your database is not on the same machine as your application server, change to the directory where you copied the JAR file `bpcodbutil.jar` .

2) Run the Oracle `loadjava` utility to install the JAR file `bpcodbutil.jar`, by entering the following command:

   ```
   loadjava -user user/password@database
       -schema  schema_name
       -resolve bpcodbutil.jar
   ```

   where:

   > *user*, *password*, and *database* are valid values for user ID, password, and the database name.

   > *schema_name* is the schema name under which the classes are to be stored. This must be the same schema as the one that is used for the Observer tables.

3) In case of problems, you can drop the JAR file using the command:

   ```
   dropjava -user user/password@database
       -schema  schema_name bpcodbutil.jar
   ```

   .

4. Create the table space

   a. Edit your copy of the `createTablespace_Observer.sql` script file according to the instruction at the top of the file.

   b. Run your copy of the `createTablespace_Observer.sql` script file according to the instruction at the top of the file.

   c. Make sure that the script output contains no errors. If errors occur, you can drop the table space using the `dropTablespace_Observer.sql` script file.

5. Create the schema (tables, indexes, and views).

   a. Edit and run your copy of the `createSchema_Observer.sql` script file according to the instruction at the top of the file.

   b. Make sure that the script output contains no errors. If you want to drop the schema, use the `dropSchema_Observer.sql` script file.

6. Use the administrative console to create an XA data source that points to the database, and test the connection.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch
between the Java-based and SQL-based user-defined functions (UDFs) in the
database for the Business Process Choreographer Observer.

**Using the setupEventCollector tool to prepare an Oracle database for the
Business Process Choreographer Observer:**

This describes how to use an interactive menu-driven tool, and the
createTablespace_Observer.sql script to prepare an Oracle database.

**About this task**

Your database must already exist. You can either use an existing database, or create
a new one according to the database documentation.

**Note:** To create a remote Oracle database from an i5/OS platform, perform "Using
SQL scripts to prepare an Oracle database for the Business Process Choreographer
Observer" on page 235.

**Procedure**
1. On Linux and UNIX platforms: Add $ORACLE_HOME/bin to the PATH
   variable.
2. Create the table space:
   a. Change to the Business Process Choreographer subdirectory where the
      configuration scripts for your database are located.
      - On Linux and UNIX platforms, change to the directory
        *install_root*/dbscripts/ProcessChoreographer/Oracle.
      - On Windows platforms, change to the directory *install_root*\dbscripts\
        ProcessChoreographer\Oracle.
   b. Edit the createTablespace_Observer.sql script file according to the instruction
      at the top of the file.
   c. Run the table space creation script file according to the instructions at the
      top of the file.
   d. Make sure that the script output contains no errors. If errors occur, you can
      drop the table space using the dropTablespace_Observer.sql script file.
3. Change to the Business Process Choreographer directory where the
   configuration scripts are located.

   On Linux and UNIX platforms, enter:

   `cd install_root/ProcessChoreographer/config`

   On Windows platforms, enter:

   `cd install_root\ProcessChoreographer\config`
4. Start the tool to set up the event collector, as described in "setupEventCollector
   tool" on page 261.
5. Prepare the database: When you see:

   ```
   1) Prepare a database for the Event Collector and Observer
   2) Install the Event Collector application
   3) Remove the Event Collector application and related objects
   4) Change configuration settings of an installed Event Collector
   5) Drop the database schema of the Event Collector and Observer
   6) Administer Observer related user-defined functions

   0) Exit Menu
   ```

Perform the following:

a. Select option 1 to prepare a database for the event collector and observer applications. The following menu is displayed:

```
Prepare a database for the WebSphere Business Process Choreographer
Event Collector and Observer

Select the type of your database provider:

    c) Derby
    d) DB2 Universal
    i) DB2 iSeries
    7) DB2 V7 on z/OS
    8) DB2 V8 on z/OS
    o) Oracle

    0) Exit Menu
```

b. Enter o to select Oracle.

c. The tool allows you to create an SQL file that you can give to your database administrator to run, rather than running it with your current user ID. When you see:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

  • If you do not want to delay running the SQL, enter n.

  • If you want to delay running the SQL, enter y. You will see:

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

    – If you want the values that you enter to be checked within the database, enter y.

    – Otherwise enter n.

  Depending on what you entered, you might not see all of the following prompts. Skip any steps that you do not see.

d. If you see:

```
Specify the database to be used.

Note: Database must already exist.

Specify the name of your database [BPEDB] :
```

  Enter the SID name of the database.

e. If you see:

```
Specify the hostname where the oracle database resides: [localhost]
```

  Enter the hostname or IP address of the database server.

f. If you see:

```
Specify the port where the oracle listener is listening: [1521]
```

  Enter the port number for the Oracle listener.

g. If you see:

```
Specify userid to connect to the database 'database_name' [system] :
```

  Enter the user ID to connect to the database. The default is system.

h. If you see:

```
Specify the password for userid 'user_ID' :
```

   Enter the password for the user ID.

i. When you see:

```
Choose the implementation of the Observer user-defined functions.

  Note: The Java UDFs are more precise, but they require a jar file
  installed to the database.
  Visit the Observer documentation for details.

  1) Java
  2) SQL

 0) Exit Menu
```

   - If you want to use the more precise Java-based user-defined functions (UDFs), which requires that a JAR file is installed in the database, enter 1.
   - If you want to use the less precise SQL-based UDFs, enter 2.

j. If you see:

k. If you see:

l. The following is displayed:

m.

   The following is displayed:

```
Trying to connect to database 'database_name', using user 'user_ID'
Connected to 'database_name'
Checking for required tablespace(es) ['OBSVRTS', 'OBSVRLOB', 'OBSVRIDX']
All required tablespaces were found.
Loading the jar file 'B:\W\P\lib\bpcodbutil.jar' into the database.
The jar file 'B:\W\P\lib\bpcodbutil.jar' was successfully installed.

The setup of the database completed successfully.
```

6. Using the administrative console, create an XA data source that points to the database.

**Results**

The database schema for the Business Process Choreographer Observer has been prepared.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

# Selecting between Java and SQL user-defined functions

You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

**Related concepts**

"User-defined functions for Business Process Choreographer Observer" on page 244
With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

## Using scripts to select between Java and SQL user-defined functions

This describes how to use scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

**About this task**

You want to use the Java implementation instead of the SQL implementation of the Business Process Choreographer Observer UDFs.

**Procedure**

1. If you intend to use the Java implementation of the Business Process Choreographer Observer user-defined functions (UDFs), copy the jar file `bpcodbutil.jar`, from the `lib` subdirectory of the *install_root* directory to the same directory on your database server.

2. If you intend to use the Java implementation of the Business Process Choreographer Observer UDFs, install the jar file `bpcodbutil.jar`:

   a. If you are not already connected to the database, connect to it by entering the following command in a DB2 command-line processor:

      ```
      db2 connect to databaseName
      ```

   b. Install the jar file by entering the following command:

      ```
      db2 call sqlj.install_jar('file:pathURL','schema.BPCODBUTIL')
      ```

      where *pathURL* is a fully qualified URL to the jar file, and *schema* is the name of the schema for the Business Process Choreographer database. For example:

      - On Linux and UNIX platforms, if the jar file is in the directory /tmp, you must enter the command:

        ```
        db2 call sqlj.install_jar('file:/tmp/bpcodbutil.jar',
            'schema.BPCODBUTIL')
        ```

      - On Windows platforms, if the jar file is in the directory c:\tmp, you must enter the command:

        ```
        db2 call sqlj.install_jar('file:c:/tmp/bpcodbutil.jar',
            'schema.BPCODBUTIL')
        ```

      **Note:** If you want to drop the jar file, use the following command:

      ```
      db2 call sqlj.remove_jar('schema.BPCODBUTIL')
      ```

3. Drop the SQL implementation of the UDFs. Edit the
   `dropFunctions_Observer.sql` script file according to the instruction at the top of
   the file. For example, for DB2, in the DB2 command-line processor, enter the
   command:

   `db2 -tf dropFunctions_Observer.sql`

4. Create the Java implementation of the UDFs. Edit the
   `createFunctionsJava_Observer.sql` script file according to the instruction at the
   top of the file. For example, for DB2, in the DB2 command-line processor, enter
   the command:

   `db2 -tf createFunctionsJava_Observer.sql`

**Results**

The UDF implementation used has been switched.

## Using the setupEventCollector tool to select between Java and SQL user-defined functions

This describes how to use an interactive menu-driven tool to switch between the
Java-based and SQL-based user-defined functions (UDFs) in the database for the
Business Process Choreographer Observer.

**About this task**

For a Derby database, setupEventCollector always uses Java-based UDFs. For other
database types, it is the default for setupEventCollector to use Java-based UDFs,
but you can use the tool to switch to SQL-based UDFs. If you change your mind
again, you can use the tool to switch back to using Java-based UDFs.

**Procedure**

1. Start the tool to set up the event collector, as described in "setupEventCollector
   tool" on page 261. You see the following menu:

   ```
   1) Prepare a database for the Event Collector and Observer
   2) Install the Event Collector application
   3) Remove the Event Collector application and related objects
   4) Change configuration settings of an installed Event Collector
   5) Drop the database schema of the Event Collector and Observer
   6) Administer Observer related user-defined functions

   0) Exit Menu
   ```

2. Select option 6 to administer the observer-related user-defined functions. You
   will see the following menu:

   ```
   c) Derby
   d) DB2 Universal
   i) DB2 iSeries
   7) DB2 V7 on z/OS
   8) DB2 V8 on z/OS
   o) Oracle
   ```

3. If you are using either DB2 for Linux, UNIX, or Windows, or DB2 for z/OS,
   select the option for your database version: d, 7, or 8 7 or 8.

   a. When you see the following menu:

      ```
      Specify which type should be used to connect to the Database:

         2) Connect using type 2 (using a native DB2 client)
         4) Connect using type 4 (directly via JDBC)
      ```

      Select one of the following options:

| 2 | For a type-2 JDBC connection, which uses a native DB2 client. In this case, you are prompted to enter the following: |
|---|---|

      **Database name**

      **Database user ID**

      **Password**

      **Directory of your JDBC driver**

| 4 | For a type-4 JDBC driver, which connects directly. In this case, you are prompted to enter the following: |
|---|---|

      **Database name**

      **Database server host name**

      **Database server port number**

      **Directory of your JDBC driver**

      **Database user ID**

      **Password**

4. If you are using Oracle, select option o.

   a. Enter the following connection information:

      **Database server host name**

      **Database server port number**

      **Database name**

      **Database user ID**

      **Password**

      **Directory of your JDBC driver**

5. If a connection can be established to the database, you will see the menu for administering the UDFs for the observer database:

```
6) Administer Observer related user-defined functions

 1) Activate Java based user-defined functions
 2) Activate SQL based user-defined functions
 3) Determine current state
 4) List, install or remove the jar file containing the java based functions
```

**Note:** The "activate" options do not apply for a Derby database.

   a. If you want to activate the Java-based UDFs, select option 1.

      1) When you see:

```
Specify the database schema to be used:
```

        Enter the name of the database schema.

      2) When you see:

```
WARNING: Switching the UDF implementation type may break any
running Observer applications. Continue anyway?
y) yes
n) no
Your selection:
```

        Enter y to continue or n not to continue.

      3) If you continue, you see something like the following:

```
Removing the user-defined functions ...

The jar file with jar_id 'DB2INST1.BPCODBUTIL' is updated
with the current version.
Loading the jar file 'B:\w\p\lib\bpcodbutil.jar' into the database.
The jar file 'BPCODBUTIL' was successfully installed.

Creating the Java based user-defined functions ...
```

   4) Success is indicated by the following message:

```
The setup of the database completed successfully.
```

b. If you want to activate the SQL-based UDFs, select option 2.

   1) When you see:

```
Specify the database schema to be used:
```

   Enter the name of the database schema.

   2) When you see:

```
WARNING: Switching the UDF implementation type may break any
running Observer applications. Continue anyway?
y) yes
n) no
Your selection:
```

   Enter y to continue or n not to continue.

   3) If you see:

```
Removing the user-defined functions ...

Creating the SQL based user-defined functions ...

Do you also want to remove the jar file from the database?
y) yes
n) no
Your selection:
```

   Enter y to remove the JAR file from the database, or n not to remove it.

   4) Success is indicated by the following message:

```
The setup of the database completed successfully.
```

c. Optional: To determine whether the selected UDF implementation is Java or SQL, and in the case that Java is active, to also verify whether the JAR file is installed, select option 3. If, for example, the Java implementation is active, you should get a message like the following:

```
The active UDF implementation is Java.
Tested functionality of the UDF, is working
```

d. Optional: To install or remove the JAR file that is required for the Java-based UDFs, or to list all JAR files that are installed in the database, select option 4, then when you see the following menu:

```
List, install or remove jar files containing the java based functions

  1) Install the jar file containing the Observer functions
     into the database
  2) Remove the jar file containing the Observer functions
     from the database
  3) List installed jar files

0) Exit Menu
```

   - Select option 1 to install the JAR file.
   - Select option 2 to remove the JAR file.
   - Select option 3 to list which JAR files are installed in the database.

- Select option 0 to exit from the menu.
  e. Select option 0 repeatedly to return to the menu shown in step 1 on page 241.

**Results**

The Business Process Choreographer Observer database will use the UDFs that you selected.

## User-defined functions for Business Process Choreographer Observer

With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

UDFs can be installed in either of the following implementations:

**SQL implementation**

Use the SQL implementation for UDFs that are implemented in plain SQL, using the build-in time functions that are provided by the database system.

Installing the SQL implementation is easier than installing the Java implementation because the SQL implementation requires to run provided SQL scripts only. For these scripts, less administration rights are required to install them. In addition, the SQL implementation has a higher performance than the Java implementation. However, because of limitations of the build-in time functions SQL implemented UDFs might not be precise enough for your needs. For example, on DB2, the build-in time function assumes that each month has the length of 30 days, which might falsify your results.

SQL implementation is not available on Derby databases.

**Java implementation**

Use the Java implementation for UDFs that are implemented using the Java language.

To install the Java implementation, use the mechanisms provided by the database system. Java implemented UDFs grant precise reports. However, installing the Java implementation requires more steps then installing the SQL implementation, and it requires more administration rights on the database. For example, on DB2 z/OS databases, a work load manager (WLM) environment has to be set up to run the UDFs.

Depending on which way you choose to setup your database, the default implementation varies:

- If you set up your database to use SQL scripts, or to use the create tables on first touch feature, the SQL implementation is installed by default.
- If you set up your database to use the setupEventCollector tool, or to use the Business Process Choreographer sample configuration in the Profile creation wizard (only provided on Derby databases), the Java implementation is installed by default.

The implementation of the UDFs can be changed after the initial setup. This is described in "Selecting between Java and SQL user-defined functions" on page 240.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch
between the Java-based and SQL-based user-defined functions (UDFs) in the
database for the Business Process Choreographer Observer.

"Using an SQL script to prepare a DB2 for iSeries database for the Business
Process Choreographer Observer" on page 217
This describes how to use the createSchema_Observer.sql script in an i5/OS
qshell environment to prepare a DB2 for iSeries database.

"Using the setupEventCollector tool to prepare a DB2 for iSeries database for
the Business Process Choreographer Observer" on page 218
This describes how to use an interactive menu-driven tool to prepare a DB2 for
iSeries database from within an i5/OS qshell environment.

"Using the setupEventCollector tool to prepare a DB2 for iSeries database from
a remote system" on page 221
This describes how to use an interactive menu-driven tool to prepare a DB2 for
iSeries database for the Business Process Choreographer Observer from a
remote Linux, Windows, or UNIX system.

"Creating a DB2 for z/OS database for the Business Process Choreographer
Observer in USS" on page 223
This describes how to use an interactive menu-driven tool, and the
createTablespace_Observer.sql script in UNIX System Services (USS) on a z/OS
machine, to create a DB2 for z/OS database.

"Creating a DB2 for z/OS database for the Business Process Choreographer
Observer from a remote system" on page 227
This describes how to use an interactive menu-driven tool, and the
createTablespace_Observer.sql script on a Linux, UNIX, or Windows system to
create a DB2 for z/OS database.

"Using SQL scripts to prepare an Oracle database for the Business Process
Choreographer Observer" on page 235
This describes how to use the createTablespace_Observer.sql and
createSchema_Observer.sql scripts to prepare an Oracle database.

"Using the setupEventCollector tool to prepare an Oracle database for the
Business Process Choreographer Observer" on page 237
This describes how to use an interactive menu-driven tool, and the
createTablespace_Observer.sql script to prepare an Oracle database.

"Using SQL scripts to prepare a DB2 Universal database for the Business
Process Choreographer Observer" on page 213
This describes how to use the createTablespace_Observer.sql and
createSchema_Observer.sql scripts to prepare a DB2 Universal database on
Linux, UNIX, and Windows platforms.

"Using the setupEventCollector tool to prepare a DB2 Universal database for
the Business Process Choreographer Observer" on page 214
This describes how to use an interactive menu-driven tool and
createTablespace_Observer.sql script to prepare a DB2 Universal database on
Linux, UNIX, and Windows platforms.

"setupEventCollector tool" on page 261
Use setupEventCollector to interactively configure or remove the Business
Process Choreographer event collector application, to setup the database, and to
administer user-defined functions for the database. This tool uses wsadmin
scripting.

# Configuring the Business Process Choreographer event collector application

Install and configure the event collector application using an interactive tool or the administrative console.

**Before you begin**

The Common Event Infrastructure (CEI) must be configured on the deployment target where you want to install the event collector application.

**About this task**

To configure Business Process Choreographer event collector, perform one of the following:

## Using the setupEventCollector tool to configure a Business Process Choreographer event collector

This describes how to use an interactive menu-driven tool to install and configure the event collector application on a server or cluster.

**Procedure**

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

   On Linux, UNIX, and i5/OS platforms, enter:

   `cd install_root/ProcessChoreographer/config`

   On Windows platforms, enter:

   `cd install_root\ProcessChoreographer\config`

2. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261. For example:

   **On Linux and UNIX platforms:**
   ```
   setupEventCollector.sh -server server1
   ```

   **On i5/OS platforms:**
   ```
   setupEventCollector -server server1
   ```

   **On Windows platforms:**
   ```
   setupEventCollector.bat -server server1
   ```

   You see the Commands Menu:
   ```
   Commands Menu

     1) Prepare a database for the Event Collector and Observer
     2) Install the Event Collector application
     3) Remove the Event Collector application and related objects
     4) Change configuration settings of an installed Event Collector
     5) Drop the database schema of the Event Collector and Observer
     6) Administer Observer related user-defined funtions

     0) Exit Menu
   ```

3. To install the Business Process Choreographer event collector application:

   a. Select option 2. The following is displayed:
      ```
      Create required objects and install the WebSphere Business Process
      Choreographer Event Collector application ...
      ```

   b. If you are installing on a standalone server, you see:
      ```
      Working on node 'your_node_name', server 'your_server_name'.
      ```

c. If you are installing the application on a deployment manager, you must select the deployment target from a list of all available targets. For example:

```
Select the deployment target to install to:

   1) Cluster 'cluster1'
   2) Node 'Node04', Server 'managed1'
   3) Node 'Node04', Server 'managed2'

   0) Exit Menu
```

d. While the tool searches for an existing event collector installation on the deployment target, you will see something like:

```
Searching for an already installed Event Collector on 'deployment_target'
```

e. If there is already an instance of the event collector application installed, you see:

```
Do you want to overwrite the existing application?
   o) Overwrite
   a) Abort
```

- Enter o to overwrite the existing event collector application. All installation values may be reentered and the event collector application is updated.

- Enter a to exit without installing the event collector.

4. When you see:

```
Specify the JNDI name of the database where the WebSphere Business Process
Choreographer Event Collector should store the collected events.
Enter '?' to get a list.
Your selection : [jdbc/BPEDB]
```

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. For example:

```
jdbc/BPEDB
jdbc/DefaultEJBTimerDataSource
jdbc/mediation/messageLog
```

5. When you see:

```
Specify the database schema to be used.
Enter a space character or leave empty to use the default schema of the
datasource. [] :
```

Enter the name of the schema for the database tables where the event collector stores the events. To use the user ID that is specified in the authentication alias of the data source definition as the schema, enter a space character or leave the field empty.

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!
```

6. 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)
```

- If you want the script to enable CEI logging on the named server, enter y.

- If you do not want the script to enable CEI logging on the named server, enter n.

**Note:** It is important that CEI logging is enabled when you start working with the Business Process Choreographer Observer.

7. When prompted:

```
Do you want to save the changes? (y/n)
```

If there were no error messages, enter y to save the configuration. If there were errors, enter n to discard the changes and keep your original configuration.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.

8. Enter 0 to exit the menu.

9. Activate the changes:
   - If you specified the `-conntype NONE` option when starting the tool, your changes become active after a server restart.
   - If you did not specify the `-conntype NONE` option when starting the tool, and you enabled CEI logging on the server during the installation of the Business Process Choreographer event collector, use the administrative console to stop and restart the `BPEContainer` application.

**Results**

The Business Process Choreographer event collector application is installed and configured.

## Using the administrative console to configure a Business Process Choreographer event collector

This describes how to use the administrative console to install an instance of the Business Process Choreographer event collector on a given server or cluster.

**Before you begin**

You have prepared the Business Process Choreographer Observer database.

**Procedure**

1. In the administrative console, navigate to the Business Process Choreographer event collector configuration page: Click **Servers** ⇒ **Clusters** ⇒ *cluster_name* or **Servers** ⇒ **Application servers** ⇒ *server_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Business Process Choreographer Event Collector**.

2. To create a new configuration:
   a. Enter or select values for the following fields:
      - Database instance name.
      - Schema name.
      - Enable or clear the option to create the database tables the first time that database is used.
      - User name and password to connect to the database.
      - Database server's host name or IP address.
      - Port number for the database server.
      - JDBC provider.
      - Observation target:

- Managed Business Process Choreographer container
- Event group
- Enter event group

b. Click **Apply** to deploy the application.

c. In case of problems, check the SystemOut.log file. Otherwise, save the changes to the master configuration.

d. Start the application by clicking **Applications** → **Enterprise Applications**, select the application BPCECollector_*scope*, where *scope* identifies the deployment target, then click **Start**.

**Results**

The Business Process Choreographer event collector is configured.

**Business Process Choreographer Event Collector:**

The event collector must be configured before you can use the Business Process Choreographer Observer.

To view this administrative console page, click **Servers** → **Clusters** → *cluster_name* or **Servers** → **Application servers** → *server_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Event Collector Configuration**.

*Data Source:*

In this section you configure the data source for the Business Process Choreographer event collector.

*Edit:*

Click this to edit the data source.

*Test Connection:*

Tests the connection to the selected data source.

*Database Instance:*

The name of the database.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | ${USER_INSTALL_ROOT}\databases\ BPOBEC00 |

*Schema Name:*

The name of the database schema.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | None |

*Create Tables:*

If selected, the tables will be created automatically the first time that the database is accessed. For this option to work, the database must already exist, and the user name provided must have the authority to create tables and indexes in the database.

For a production system, it is not recommended to use this option. If you do not select this option, the tables will not be created automatically, and you must create the tables manually by running scripts.

| Property | Value |
| --- | --- |
| Data type | Check box |
| Default | Selected |

*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.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | The currently logged on user. |

*Password:*

The password for the data source user ID.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | None |

*Server:*

The address of the database server.

Specify either the hostname or the IP address.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | None |

*Port:*

The port number used by the database server.

| Property | Value |
| --- | --- |
| Data type | String |
| Default | Depends on which JDBC provider is selected. |

*Provider:*

The JDBC provider.

| Property | Value |
| --- | --- |
| Data type | Drop-down list |
| Default | DERBY_EMBEDDED |

*JMS user name:*

| Property | Value |
| --- | --- |
| Data type | String |
| Default | The same user ID that was specified for the Business Process Choreographer JMS user. |

*JMS password:*

| Property | Value |
| --- | --- |
| Data type | String |
| Default | The same password that was specified for the Business Process Choreographer JMS user. |

*Observation target:*

In this section you specify the target for the event collector.

| Property | Value |
| --- | --- |
| Data type | Radio buttons |
| Choice | • Managed Business Process Choreographer container<br>• Event Group Profile List<br>• Enter Event Group Profile List |

*Managed Business Process Choreographer container:*

Select a configured Business Process Choreographer container.

| Property | Value |
| --- | --- |
| Data type | Drop-down list |
| Contents | All Business Process Choreographer configurations |

*Event group profile list:*

Select a group profile.

| Property | Value |
| --- | --- |
| Data type | Drop-down list |

*Enter event group profile list:*

Enter an event group profile list.

| Property | Value |
|---|---|
| Data type | String |
| Default | None |

# Configuring the Business Process Choreographer Observer application

You can either use a tool or the administrative console to configure the Business Process Choreographer Observer application.

**Before you begin**

You have configured a Business Process Choreographer event collector.

**About this task**

To configure Business Process Choreographer Observer, perform one of the following:

## Using the setupObserver tool to configure a Business Process Choreographer Observer

This describes how to use an interactive menu-driven tool to install an instance of the Business Process Choreographer Observer application and configure it to connect to the data source for a particular event collector.

**Before you begin**

You have prepared the Business Process Choreographer Observer database, and used the administrative console to create a data source for it.

**Procedure**

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

   On Linux, UNIX, and i5/OS platforms, enter:

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

   On Windows platforms, enter:

   ```
   cd install_root\ProcessChoreographer\config
   ```

2. Start the tool to set up the observer, as described in "setupObserver tool" on page 264. You see the menu:

   ```
   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 the Business Process Choreographer Observer. The following is displayed:

   ```
   Create required objects and install the WebSphere Business Process
   Choreographer Observer application ...
   ```

4. If you are installing on a standalone server, you see:

   ```
   Working on node 'your_node_name', server 'your_server_name'.
   ```

5. If you are installing the application on a deployment manager, you must select the deployment target from a list of all available targets. For example:

```
Select the deployment target to install to:

1) Cluster 'cluster1'
2) Node 'Node04', Server 'managed1'
3) Node 'Node04', Server 'managed2'

0) Exit Menu
```

6. When you see:

```
Specify the JNDI name of the database containing the event tables.
Enter '?' to get a list.
Your selection : [jdbc/BPEDB]
```

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. For example:

```
jdbc/BPEDB
jdbc/DefaultEJBTimerDataSource
jdbc/mediation/messageLog
```

7. When you see:

```
Specify the database schema to be used.
Enter a space character or leave empty to use the default schema of the
datasource. [] :
```

Enter the name of the schema for the database tables where the event collector stores the events. To use the user ID that is specified in the authentication alias of the data source definition as the schema, enter a space character or leave the field empty.

All required objects are created and the enterprise application is installed. Success is indicated by the message:

```
WebSphere BPC Observer installed successfully!
```

8. When prompted: Do you want to save the changes? (y/n), if there were no error messages, enter y to save the configuration. If there were errors, enter n to discard the changes and keep your original configuration. Check the log file named setupObserver.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.

9. Enter 0 to exit the menu.

**Results**

The Business Process Choreographer Observer is installed and configured.

## Using the administrative console to configure a Business Process Choreographer Observer

This describes how to use the administrative console to install an instance of the Business Process Choreographer Observer application and configure it to connect to the data source for a particular event collector.

**Before you begin**

You have configured the Business Process Choreographer event collector.

**Procedure**

1. In the administrative console, navigate to the Business Process Choreographer Observer configuration page: Click **Servers** → **Clusters** → *cluster_name* or

Servers ▸ **Application servers** ▸ *server_name*, then under **Business Integration**, expand **Business Process Choreographer**, and click **Business Process Choreographer Observer**.

2. To create a new configuration:

   a. Click **Add**.

   b. Enter or select values for the following fields:
      - Context root for this instance
      - Visualize monitoring data from this Business Process Choreographer event collector

      For more information about these configuration parameters, see "Business Process Choreographer Observer settings" on page 139.

   c. Click **Apply** to deploy the application.

   d. In case of problems, check the SystemOut.log file. Otherwise, save the changes to the master configuration.

   e. Start the application by clicking **Applications** ▸ **Enterprise Applications**, select the application BPCObserver_*scope*, where *scope* identifies the deployment target, then click **Start**.

**Results**

The Business Process Choreographer Observer configured and ready to use.

**What to do next**

You can configure more instances of the Business Process Choreographer Observer, on the same or different deployment targets, by repeating this task. However, each instance must connect to a different event collector data source.

# Enabling logging for Business Process Choreographer

This describes how to enable Common Event Infrastructure (CEI) events for Business Process Choreographer.

**Before you begin**

To monitor business process events with the Business Process Choreographer Observer, your business process must be enabled to emit Common Event Infrastructure (CEI) events. You specify this when modelling your business process. In order to properly monitor a business process, at least the "Process Started" event has to be emitted. For a list of CEI events that you can monitor using the Business Process Choreographer Observer, see Business process events. For information about how to enable a business process to emit CEI events, refer to the WebSphere Integration Developer Information Center.

**About this task**

If you installed the Business Process Choreographer event collector on the same deployment target as the one where Business Process Choreographer is configured, you can use the setupEventCollector tool to enable CEI logging when you install the application. If you installed the Business Process Choreographer event collector using the administrative console you must enable CEI logging, either by using a script or using the administrative console.

To use a Jython script to enable CEI logging for the Business Process Choreographer, perform "Using a script to enable logging for the Business Process Choreographer."

To enable CEI logging for Business Process Choreographer, using the administrative console, perform "Enabling Common Base Events and the audit trail, using the administrative console" on page 288.

**Results**

Common Event Infrastructure events for your business processes and activities will be emitted, and can be received by a Business Process Choreographer event collector.

## Using a script to enable logging for the Business Process Choreographer

This describes how to use the setStateObserver.py script to enable or disable Common Event Infrastructure (CEI) or audit events for Business Process Choreographer.

### Location

The setStateObserver.py script is located in the Business Process Choreographer config directory.

### Running the script

To run the setStateObserver script:

On Linux and UNIX platforms, enter:

```
install_root/bin/wsadmin.sh -lang jython
      -f install_root/ProcessChoreographer/config/setStateObserver.py
```

On i5/OS platforms, enter:

```
install_root/bin/wsadmin -lang jython
      -f install_root/ProcessChoreographer/config/setStateObserver.py
```

On Windows platforms, enter:

```
install_root\bin\wsadmin.bat -lang jython
      -f install_root\ProcessChoreographer\config\setStateObserver.py
```

### Parameters

The script file can take the following parameters:

**-conntype** *NONE*
    Only include this option if the application server (for stand-alone) or deployment manager is not running.

**-node** *nodeName*
    Where *nodeName* is the name of the node. Do not specify this option if you specify a cluster.

**-server** *serverName*
    Where *serverName* is the name of the server. Do not specify this option if you specify a cluster.

**-cluster** *clusterName*
> Where *clusterName* is the name of the cluster. Do not specify this option in a stand-alone server environment, nor if you specify the node and server.

**-profileName** *profileName*
> Where *profileName* is the name of the profile to use.

**-enable ( CEI | AuditLog | CEI;AuditLog)**
> Optionally specifies whether to enable CEI logging, audit logging, or both.

**-disable ( CEI | AuditLog | CEI;AuditLog)**
> Optionally specifies whether to disable CEI logging, audit logging, or both.

**-bfm**
> Optionally specifies that the enabling or disabling is to apply to the Business Process Choreographer's Business Flow Manager, which runs business processes.

**-htm**
> Optionally specifies that the enabling or disabling is to apply to the Business Process Choreographer's Human Task Manager, which runs human tasks.

### Example

To enable CEI logging for business process events on `server1`, on Linux or UNIX platforms:
```
wsadmin.sh -lang jython -f setStateObserver.py -server server1 -enable CEI -bfm
```

**Note:** On Windows, use `wsadmin.bat`, and on i5/OS use `wsadmin`.

# Changing configuration parameters for the Business Process Choreographer Observer

Tuning the configuration parameters for the Business Process Choreographer Observer and event collector applications is important to enable verification and improve performance.

## Changing default values

The default values are more suitable for a production system than for a test system. If you are setting up the Business Process Choreographer for development or testing, it makes sense to change the following configuration parameters before you verify that the configuration is working:

*   Change BPCEventTransformerEventCount to the value zero.
*   Change BPCEventTransformerToleranceTime to the value one.

Making these changes ensures that even when events that are emitted at lower rates than in a production system, the events become available in one minute.

## Configuration parameters for the event collector

Tuning the numerical parameters affects how often the event transformer is triggered, and the age at which events are made available to the Business Process Choreographer Observer.

| Configuration parameter | Data type / Units | Default value | Description |
|---|---|---|---|
| ObserverSchemaName | String | not set | This identifies the database schema that is used as a prefix for all database objects. If it is left empty, the default is to use as a prefix, the user ID that is used to connect to the database. This user ID is set as part of the data source definition in the administrative console. If you specify a value for this parameter, the user ID specified at the data source must have sufficient rights to access the database objects for this schema. |
| BPCEventTransformer EventCount | Integer / Events | 500 | The number of events after which the event collector triggers the transformer to transform the collected events into a format suitable for the observer application.\n\nWhen you are developing, testing, and experimenting, the default value is probably too high, and causes events to remain unobservable for a long time. To make events available faster, you can set this value to zero. Every future event will then trigger the transformer, and will become visible in the observer. If you change the value to zero, any past events that have not yet been transformed will be transformed as soon as a new event is generated. Using a zero value is not recommended for a production system. |
| BPCEventTransformer MaxWaitTime | Integer / Minutes | 10 | The maximum time that can pass before the transformer is triggered - even though the number of events specified with BPCEventTransformer EventCount is not reached. |

| Configuration parameter | Data type / Units | Default value | Description |
| --- | --- | --- | --- |
| BPCEventTransformer ToleranceTime | Integer / Minutes | 10 | The minimum age in minutes for an event to become visible in the observer. This enables related events to be reliably correlated. Using the value zero should be avoided, otherwise it is possible that an event is processed before the predecessor event has arrived. |
| | | | When you are developing, testing, and experimenting, the default value is probably too high, and causes new events to remain unobservable for 10 minutes. If you set this to the value 1, all transformed events that are more than one minute old will be visible in the observer. |
| ObserverCreateTables | Boolean | | This parameter indicates if the Observer schema should be created when the EJB connects to the database the first time. Valid values are 'true' and 'false'. |

When the event collector receives a business relevant event from the Common Event Infrastructure (CEI), the event is saved in the database. After some time has passed and more events have been received, the transformer is started. The transformer performs a batch transformation of the stored events and writes them back to the database in a format that can be used for generating reports. Only events that have been processed by the transformer are available for the observer reports.

Every time that a new event is received by the event collector, if one or both of the following conditions are true, then the transformer process is started:

- The number of events received since the transformer was last started is greater than the value for BPCEventTransformerEventCount.
- The time since the transformer was last started is greater than the value of BPCEventTransformerMaxWaitTime, in minutes.

If these values are made smaller, events will be available sooner for generating reports, but there is some extra cost in transforming small numbers of events. This requires a balance between having better transformation throughput, by processing larger numbers of events, against the possible need to make the events available in the observer database as fast as possible.

Each time that the transformer is started, it processes all events that are older than BPCEventTransformerToleranceTime in minutes. It does not process more recent events because events are not necessarily published in the order that they occur.

The default setting of BPCEventTransformerToleranceTime assumes that no event will take more than 10 minutes to be received and written to the event collector table.

## Changing configuration parameters for the event collector

To change event collector parameters, perform the following:

1. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261. You see the following menu:

   ```
   1) Prepare a database for the Event Collector and Observer
   2) Install the Event Collector application
   3) Remove the Event Collector application and related objects
   4) Change configuration settings of an installed Event Collector
   5) Drop the database schema of the Event Collector and Observer
   6) Administer Observer related user-defined functions

   0) Exit Menu
   ```

2. Select option 4 to display the list of parameters that you can change:

   ```
   1) BPCEventTransformerEventCount
   2) BPCEventTransformerMaxWaitTime
   3) BPCEventTransformerToleranceTime
   4) ObserverCreateTables
   5) ObserverSchemaName

   0) Exit Menu
   ```

3. Select the number of the parameter that you want to change. The parameter's name, description, type, units, and current value are displayed.

4. To change the specified value, enter a new value and press Enter. Pressing Enter without a new value returns to the parameter list.

5. If you want to change the value of another parameter, repeat from step 3.

6. Enter 0 to exit the list. You are asked whether you want to save the changes.

7. To save all changes, enter y, otherwise enter n to discard all changes.

8. To activate the changes, restart the BPCECollector application.

## Configuration parameters for the observer

The value for the ReportAtSnapshotRange parameter can have a big impact on the performance of snapshot reports.

| Configuration parameter | Data type / Units | Default value | Description |
|---|---|---|---|
| ObserverSchemaName | String | not set | This identifies the database schema that is used as a prefix for all database objects. If it is left empty, the default is to use as a prefix, the user ID that is used to connect to the database. This user ID is set as part of the data source definition in the administrative console. If you specify a value for this parameter, the user ID specified at the data source must have sufficient rights to access the database objects for this schema. This must match the value for the event collector. |
| ReportAtSnapshotRange | Integer / Days | 60 | A snapshot report is built by evaluating all events that are older than the qualifying snapshot date and time. ReportAtSnapshotRange defines the period of time in which events can be included in a snapshot report. Only events that have been emitted within this period are evaluated by the snapshot report.<br><br>If this value is too high, a very large number of events might have to be processed, and generating a report can take a long time. Try setting this value to the maximum duration of a process instance in your business environment. |
| ObserverCreateTables | Boolean | | This parameter indicates if the Observer schema should be created when the EJB connects to the database the first time. Valid values are 'true' and 'false'. |

## Changing configuration parameters for the observer

To change observer parameters, perform the following:

1. Start the tool to set up the observer, as described in "setupObserver tool" on page 264. You see the menu:

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

   0) Exit Menu
```

2. Select option 3 to display the list of parameters that you can change:

```
   1) ReportAtSnapshotRange
   2) ObserverCreateTables
   3) ObserverSchemaName

   0) Exit Menu
```

3. Select the number of the parameter that you want to change. The parameter's name, description, type, units, and current value are displayed.

4. To change the specified value, enter a new value and press Enter. Pressing Enter without a new value returns to the parameter list.

5. If you want to change the value of another parameter, repeat from step 3.

6. Enter 0 to exit the list. You are asked whether you want to save the changes.

7. To save all changes, enter y, otherwise enter n to discard all changes.

8. To activate the changes, restart the BPCObserver application.

## setupEventCollector tool

Use setupEventCollector to interactively configure or remove the Business Process Choreographer event collector application, to setup the database, and to administer user-defined functions for the database. This tool uses wsadmin scripting.

### Location

This tool is located in the Business Process Choreographer subdirectory for configuration scripts:

On Linux, UNIX, and i5/OS platforms: *install_root*/ProcessChoreographer/config.

On Windows platforms: *install_root*\ProcessChoreographer\config.

### Restrictions

- In a network deployment environment, you must start the tool on the deployment manager node, using the -profileName option to specify the deployment manager profile .
- This tool is only available in English.
- On i5/OS, you must run the tool using qshell.

### Parameters

```
   [-conntype SOAP | RMI | JMS | NONE]
   [-user userID -password password]
   [-profileName profileName]
   ( [-node nodeName] [-server serverName] ) | ( -cluster clusterName )
   [-remove [-silent]]
```

Where:

**-conntype SOAP | RMI | JMS | NONE**
    The connection mode that wsadmin tool uses. In a standalone server environment only include the -conntype NONE option if the application server is not running. In a network deployment environment, only include -conntype NONE option if the deployment manager is not running.

**-user** *userID* **-password** *password*

> If global security is enabled, also provide a valid user ID and password for the tool to use.

**-profileName** *profileName*

> If you are not configuring the default profile, provide the name of the profile that you want to configure.

**-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. This parameter is optional.

**-cluster** *clusterName*

> The cluster name *clusterName* . This parameter is optional.

**-remove**

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

**-silent**

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

**Note:** If you do not specify the -node, -server, nor -cluster parameters, you will be prompted for the deployment target during configuration.

## Example: Starting the tool

To start the tool to work with a server named server1 enter one of the following commands.

**On Linux and UNIX platforms:**
```
setupEventCollector.sh -server server1
```

**On i5/OS platforms:**
```
setupEventCollector -server server1
```

**On Windows platforms:**
```
setupEventCollector.bat -server server1
```

You will see the Commands menu:
```
1) Prepare a database for the Event Collector and Observer
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

## Using the tool

How to use this tool for particular tasks is described in the following topics.

> **Related concepts**
>
> "User-defined functions for Business Process Choreographer Observer" on page 244
> With Business Process Choreographer Observer you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports,

Business Process Choreographer Observer requires some specific user-defined functions (UDFs) to be installed in the database.

**Related tasks**

"Configuring the Business Process Choreographer event collector application" on page 246
Install and configure the event collector application using an interactive tool or the administrative console.

"Using the setupEventCollector tool to prepare a DB2 Universal database for the Business Process Choreographer Observer" on page 214
This describes how to use an interactive menu-driven tool and createTablespace_Observer.sql script to prepare a DB2 Universal database on Linux, UNIX, and Windows platforms.

"Using the setupEventCollector tool to prepare a DB2 for iSeries database for the Business Process Choreographer Observer" on page 218
This describes how to use an interactive menu-driven tool to prepare a DB2 for iSeries database from within an i5/OS qshell environment.

"Creating a DB2 for z/OS database for the Business Process Choreographer Observer in USS" on page 223
This describes how to use an interactive menu-driven tool, and the createTablespace_Observer.sql script in UNIX System Services (USS) on a z/OS machine, to create a DB2 for z/OS database.

"Creating a DB2 for z/OS database for the Business Process Choreographer Observer from a remote system" on page 227
This describes how to use an interactive menu-driven tool, and the createTablespace_Observer.sql script on a Linux, UNIX, or Windows system to create a DB2 for z/OS database.

"Using the setupEventCollector tool to prepare a Derby database for the Business Process Choreographer Observer" on page 232
This describes how to use an interactive menu-driven tool, setupEventcollector, to prepare a Derby database on any supported platform.

"Using the setupEventCollector tool to prepare an Oracle database for the Business Process Choreographer Observer" on page 237
This describes how to use an interactive menu-driven tool, and the createTablespace_Observer.sql script to prepare an Oracle database.

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

"Using a script to remove the Business Process Choreographer configuration" on page 271
Use this task to remove the Business Flow Manager, Human Task Manager, Business Process Choreographer Explorer and Business Process Choreographer Observer configuration, and the associated resources from a server or cluster.

**Related reference**

"Changing configuration parameters for the Business Process Choreographer Observer" on page 256
Tuning the configuration parameters for the Business Process Choreographer Observer and event collector applications is important to enable verification and improve performance.

## setupObserver tool

Use setupObserver to interactively configure or remove the Business Process Choreographer Observer application, or to change configuration parameters. This tool uses wsadmin scripting.

### Location

This tool is located in the Business Process Choreographer subdirectory for configuration scripts:

On Linux, UNIX, and i5/OS platforms: *install_root*/ProcessChoreographer/config.

On Windows platforms: *install_root*\ProcessChoreographer\config.

### Restrictions

- In a network deployment environment, you must start the tool on the deployment manager node, using the `-profileName` option to specify the deployment manager profile .
- This tool is only available in English.
- On i5/OS, you must run the tool using qshell.

### Parameters

```
[-conntype SOAP | RMI | JMS | NONE]
[-user userID -password password]
[-profileName profileName]
( [-node nodeName] [-server serverName] ) | ( -cluster clusterName )
[-remove [-silent]]
```

Where:

**-conntype SOAP | RMI | JMS | NONE**
> The connection mode that wsadmin tool uses. In a standalone server environment only include the `-conntype NONE` option if the application server is not running. In a network deployment environment, only include `-conntype NONE` option if the deployment manager is not running.

**-user** *userID* **-password** *password*
> If global security is enabled, also provide a valid user ID and password for the tool to use.

**-profileName** *profileName*
> If you are not configuring the default profile, provide the name of the profile that you want to configure.

**-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. This parameter is optional.

**-cluster** *clusterName*
> The cluster name *clusterName* . This parameter is optional.

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

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

**Note:** If you do not specify the -node, -server, nor -cluster parameters, you will be prompted for the deployment target during configuration.

### Example: Starting the tool

To start the tool to work with a server named server1 enter one of the following commands.

**On Linux and UNIX platforms:**
```
setupObserver.sh -server server1
```

**On i5/OS platforms:**
```
setupObserver -server server1
```

**On Windows platforms:**
```
setupObserver.bat -server server1
```

You will see the Commands menu:
```
1) Install the Observer application
2) Remove the Observer application and related objects
3) Change configuration settings of an installed Observer

0) Exit Menu
```

### Using the tool

How to use this tool for particular tasks is described in the following topics.

> **Related tasks**
>
> "Using the setupObserver tool to configure a Business Process Choreographer Observer" on page 252
> This describes how to use an interactive menu-driven tool to install an instance of the Business Process Choreographer Observer application and configure it to connect to the data source for a particular event collector.
>
> "Using a script to remove the Business Process Choreographer configuration" on page 271
> Use this task to remove the Business Flow Manager, Human Task Manager, Business Process Choreographer Explorer and Business Process Choreographer Observer configuration, and the associated resources from a server or cluster.
>
> **Related reference**
>
> "Changing configuration parameters for the Business Process Choreographer Observer" on page 256
> Tuning the configuration parameters for the Business Process Choreographer Observer and event collector applications is important to enable verification and improve performance.

## Verifying the Business Process Choreographer Observer

After installing and configuring the Business Process Choreographer Observer, verify that the observer is working correctly.

**Before you begin**

Initially, the Business Process Choreographer Observer database is empty.

**Procedure**

1. Perform actions that will generate business events, for example, use the Business Process Choreographer Explorer to start a process instance.

2. In a browser, start Business Process Choreographer Observer by opening the URL `http://host:port/context_root/`. Where *host* is the name of the host where your application server is running, *port* is the port number for your application server (the default is 9080), and *context_root* is typically `bpcobserver`.

3. Verify that the events you expect to be available are displayed. If you see no events, wait a few minutes, restart the event collector application, and then refresh your browser view.

   **Note:** Using the default values for BPCEventTransfomerMaxWaitTime and BPCEventTransfomerToleranceTime, it could take up to 20 minutes until the transformer is triggered and the events in the event collector table are old enough to be processed and made available. For information about these parameters, including how to change them and suggested values for testing purposes, see "Changing configuration parameters for the Business Process Choreographer Observer" on page 256.

4. In case of problems, see "Troubleshooting Business Process Choreographer Observer" on page 641.

**Results**

The Business Process Choreographer Observer is working.

# Activating Business Process Choreographer

After configuring Business Process Choreographer, you must restart the affected server or cluster.

**About this task**

To activate Business Process Choreographer:

**Procedure**

1. If you configured Business Process Choreographer on a server, restart the server.

2. If you configured Business Process Choreographer on a cluster, restart the cluster.

3. Make sure that there are no error messages in the `SystemOut.log` file for the application server. On a cluster, check the log for all application servers in the cluster.

4. Verify that the Business Flow Manager and Human Task Manager applications have started successfully: In the administrative console, select **Applications** → **Enterprise Applications** and verify that the applications with names starting with `BPEContainer_scope` and `TaskContainer_scope` have started.

   Where, the value of *scope* is *nodeName_serverName*, if you configured Business Process Choreographer on an application server, or the *clusterName* if you configured Business Process Choreographer on a cluster.

**Results**

Business Process Choreographer is running.

**What to do next**

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.

**Procedure**

1. Using either the administrative console or the wsadmin command, install the application in *install_root*/installableApps/bpcivt.ear. 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. Depending on where you configured Business Process Choreographer, make sure that either:
   - The application server is running.
   - At least one member of the cluster is running.

3. Make sure that the database system, and messaging service are running.

4. Select the application BPCIVTApp and click **Start** to start the application.

5. 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.

6. Optional: Stop and remove the bpcivt application.

7. If an error occurs, it can be caused by any of the following:
   - If Business Process Choreographer cannot access the database, check that the database system is running, that all database clients are correctly configured, and that the data source is defined correctly. Make sure that the user ID and password for the data source are valid.
   - If Business Process Choreographer cannot read the input queues, check that the messaging service is running, and make sure that the JMS provider and JMS resources are defined correctly.

**Results**

The basic functionality of your Business Process Choreographer configuration works.

**What to do next**

If you have configured other optional parts, such as the Business Process Choreographer Observer, Business Process Choreographer Explorer, or a people directory provider, they will need to be tested separately.

## Understanding the startup behavior of Business Process Choreographer

This topic explains why Business Process Choreographer is unavailable until all enterprise applications are started.

When the Business Process Choreographer is started or restarted, no messages in the internal queues are processed until all enterprise applications have started. It is not possible to change this behavior. The time that the Business Flow Manager and Human Task Manager are unavailable during a restart depends on how long it takes until all enterprise applications are started. This behavior is necessary to avoid navigating any 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 ClassNotFound exceptions.

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

If your server is not running in development mode, you can federate a server that is in a stand-alone profile to a new deployment manager cell.

**Before you begin**

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 the embedded Derby database. Also, the database for the messaging engine database must be remotely accessible, that is, it cannot be Derby Embedded and it cannot be FILESTORE.

**About this task**

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.

**Procedure**

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 Network Deployment 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. Start the server to activate the changes.
4. 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.

**Results**

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.

**What to do next**

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

# Chapter 5. Removing the Business Process Choreographer configuration

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

**Procedure**

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 all human task templates and all business process templates.
3. Uninstall all enterprise applications that contain human tasks or business processes.
4. Perform one of the following actions:
   - To remove the Business Process Choreographer configuration, Business Process Choreographer Explorer, Business Process Choreographer Observer, event collector, 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 274.

**Results**

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 Flow Manager, Human Task Manager, Business Process Choreographer Explorer and Business Process Choreographer Observer configuration, and the associated resources from a server or cluster.

**Before you begin**

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 all enterprise applications that contain business processes or human tasks.

**Procedure**

1. Change to the Business Process Choreographer `config` directory:

   On Windows platforms, enter the command:

   cd *install_root*\ProcessChoreographer\config

   On Linux, UNIX, and i5/OS platforms, enter:

   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 Derby databases are not locked and can be removed automatically.
- In a network deployment 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 local Derby databases.
- If WebSphere administrative security is enabled, specify also the user ID and password:

  ```
  -user userID  -password password
  ```
- If you are not removing the configuring from the default profile, specify also the profile name:

  ```
  -profileName profileName
  ```

| Option | Description |
|---|---|
| **For a single server on Linux or UNIX** | Enter the command:<br><br>*install_root*/bin/wsadmin.sh<br>    -f bpeunconfig.jacl<br>    -server *Server* -node *Node*<br>    [-deleteDB *deleteDatabase*] |
| **For a single server on Windows** | Enter the command:<br><br>*install_root*\bin\wsadmin.bat<br>    -f bpeunconfig.jacl<br>    -server *Server* -node *Node*<br>    [-deleteDB *deleteDatabase*] |
| **For a single server on i5/OS** | Enter the command:<br><br>*install_root*/bin/wsadmin<br>    -f bpeunconfig.jacl<br>    -server *Server* -node *Node*<br>    [-deleteDB *deleteDatabase*] |
| **For a cluster on Linux or UNIX** | Enter the command:<br><br>*install_root*/bin/wsadmin.sh<br>    -f bpeunconfig.jacl<br>    -cluster *Cluster* |
| **For a cluster on Windows** | Enter the command:<br><br>*install_root*\bin\wsadmin.bat<br>    -f bpeunconfig.jacl<br>    -cluster *Cluster* |
| **For a cluster on i5/OS** | Enter the command:<br><br>*install_root*/bin/wsadmin<br>    -f bpeunconfig.jacl<br>    -cluster *Cluster* |

Where:
*userID*  The user ID.
*password*
      The password for the user ID.

*profileName*
>    The name of the profile that is being configured. If you are configuring the default profile, this option is optional.

*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.

*deleteDatabase*
>    A Boolean value that specifies whether to delete any Derby embedded databases and FILESTORE directories:
>
>    **yes**
>    **no**
>
>    To use this option, the server must not be running. If you do not have any non-Derby Embedded databases, and you use this option, after running the script you can skip to step 4.

If you omit a required parameter, you are prompted for it.

3. Optional: Delete the databases 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. The list of databases is also written to the *install_root*/profiles/*profileName*/logs/bpeunconfig.log log file. You can use this list to identify which databases you might want to delete manually.

   - When a Derby 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.

   - When Derby 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.

   - When FILESTORE is used for the Business Process Choreographer messaging engine message store, using the bpeunconfig.jacl script's -deleteDB yes option will also delete the associated directories.

   - To remove the Business Process Choreographer Observer database, start the tool to set up the event collector, as described in "setupEventCollector tool" on page 261, and select the option **Drop the database schema of the Event Collector and Observer**.

4. Optional: Check the bpeunconfig.log log file. It is located in the logs subdirectory of the *profile_root* directory.

5. Optional: If you used WebSphere MQ, delete the queue manager used by Business Process Choreographer.

6. 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
- Setting WebSphere variables

**Results**

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 tools to remove the Business Process Choreographer Observer and event collector

Remove the Business Process Choreographer Observer and event collector applications, and the associated resources from a server or cluster.

**Procedure**

1. If you configured Business Process Choreographer Observer, run the "setupObserver tool" on page 264, and specify the -remove option
2. If you configured Business Process Choreographer event collector application, run the "setupEventCollector tool" on page 261, and specify the -remove option.
3. Optional: If you installed the Java user-defined functions, remove them.
4. Optional: Delete the database.

**Results**

The Business Process Choreographer Observer and event collector applications 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 Choreographer configuration, including the Business Process Choreographer Explorer, Business Process Choreographer Observer, and the associated resources.

**Before you begin**

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

**Procedure**

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 names that start with the following:

- `BPEContainer_scope` is the Business Flow Manager application.
- `TaskContainer_scope` is the Human Task Manager application.
- `BPCExplorer_scope` is the Business Process Choreographer Explorer application.

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* – even if the server was later promoted to a cluster.
- If Business Process Choreographer was configured on a cluster, *scope* has the value *clusterName*.

c. Optional: If you configured Business Process Choreographer, uninstall the Business Flow Manager and Human Task Manager applications. Select `BPEContainer_scope`, and `TaskContainer_scope`, then click **Uninstall** ▸ **OK** ▸ **Save**.

d. Optional: If you configured the Business Process Choreographer Explorer, uninstall it.

- If you used the default context root, `/bpc`, select `BPCExplorer_scope`, then click **Uninstall** ▸ **OK** ▸ **Save**.
- Otherwise, , select `BPCExplorer_scope_context_root`, then click **Uninstall** ▸ **OK** ▸ **Save**.

**Note:** Uninstalling the Business Process Choreographer Observer and event collector applications are described in step 10 on page 279.

2. Remove all or any of the following resources that you do not want to reuse:

a. Optional: Find the Business Process Choreographer data source (the default name is `BPEDataSourcedbType`) and make a note of its name and its associated authentication data alias (if any) and Java Naming and Directory Interface (JNDI) name before removing it (the default name is `jdbc/BPEDB`).

To find the data source:

1) Click **Resources** ▸ **JDBC** ▸ **Data sources**.
2) For **Scope**, select the server or cluster where Business Process Choreographer was configured.

b. Optional: For a database other than a Derby database, remove the JDBC provider of the data source identified in step 2, unless it contains further data sources that you still need. Click **Resources** ▸ **JDBC** ▸ **JDBC Providers**, select the JDBC driver for your database and click **Delete**.

**Note:** If the Business Process Choreographer configuration uses the built-in default JDBC provider for the Derby Embedded database, this JDBC provider cannot be deleted.

c. 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** ▸ **Connection factories**. For **Scope**, select the server or cluster where Business Process Choreographer was configured. Then select the connection factory, and click **Delete**.
2) Click **Resources** ▸ **JMS** ▸ **Queues**. For **Scope**, select the server or cluster where Business Process Choreographer was configured. Then select the queues, and click **Delete**.

- For WebSphere MQ, remove the JMS queue connection factories and JMS queues.

    1) Click **Resources** → **JMS** → **Queue connection factories**. For **Scope**, select the server or cluster where Business Process Choreographer was configured. Then select the connection factory, and click **Delete**.

    2) Click **Resources** → **JMS** → **Queues**. For **Scope**, select the server or cluster where Business Process Choreographer was configured. Then select the queues, and click **Delete**.

    For the business process container the JNDI names are normally as follows:
    ```
    jms/BPECF
    jms/BPECFC
    jms/BFMJMSReplyCF
    jms/BPEIntQueue
    jms/BPERetQueue
    jms/BPEHldQueue
    jms/BFMJMSAPIQueue
    jms/BFMJMSCallbackQueue
    jms/BFMJMSReplyQueue
    ```

    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** → **Activation specifications**. For **Scope**, select the server or cluster where Business Process Choreographer was configured.

    2) Remove the following activation specifications:
    ```
    BPEInternalActivationSpec
    BFMJMSAS
    HTMInternalActivationSpec
    ```

e. Optional: If you are using WebSphere MQ as the JMS provider, remove the listener ports for the server.

    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
    BPEHoldListenerPort
    HTMInternalListenerPort
    ```

    If you configured Business Process Choreographer on a cluster, repeat this step for each member of the cluster.

f. Optional: Delete the authentication data aliases.

    1) Click **Security** → **Secure administration, applications, and infrastructure** then in the **Authentication** section, expand **Java Authentication and Authorization Service** , click **J2C authentication data**.

    2) If the data source identified in step 2 on page 275 had an authentication data alias, remove that alias. If you did not migrate your Business Process Choreographer configuration from Version 6.0.x, the name depends on the deployment target in the following manner:

- When Business Process Choreographer is configured on a server named *serverName*, on a node named *nodeName*, the name is usually `BPCDB_nodeName.serverName_Auth_Alias`.
- When Business Process Choreographer is configured on a cluster named *clusterName*, the name is usually `BPCDB_clusterName_Auth_Alias`

3) If any of the connection factories identified in step 2c on page 275 have an authentication data alias, remove the alias with great care:

- If you did **not** migrate your Business Process Choreographer configuration from Version 6.0.x, the name is `BPC_Auth_Alias` and it is shared between all Business Process Choreographer configurations in a network deployment environment.

  **Attention:** Only remove this authentication alias if you are removing the last Business Process Choreographer configuration, otherwise the remaining Business Process Choreographer configurations will stop working.

- If you migrated your Business Process Choreographer configuration from Version 6.0.x, the name is normally *cellName/*`BPEAuthDataAliasJMS_scope`, where *cellName* is the name of the cell, and *scope* identifies the deployment target. You can remove this authentication alias without affecting other Business Process Choreographer configurations.

g. Optional: Remove the scheduler configuration for the data source JNDI name.

  1) Click **Resources** → **Schedulers**.

  2) For **Scope**, select the server or cluster where Business Process Choreographer was configured.

  3) On the Schedulers pane, note the JNDI name of the work manager, then select and delete the scheduler named `BPEScheduler`.

h. Optional: Remove the work manager.

  1) Click **Resources** → **Asynchronous beans** → **Work managers**.

  2) For **Scope**, select the server or cluster where Business Process Choreographer was configured.

  3) On the Work managers pane, select and delete the work manager whose JNDI name you noted in step 2g.

  4) Also delete the work manager with the JNDI name `wm/BPENavigationWorkManager`.

i. Optional: Remove the work area partition.

  1) Click **Servers** → **Application servers** → *serverName*.

  2) Under the **Container Settings** section, expand **Business Process Services**, click **Work area partition service**.

  3) On the Application servers pane, select and delete the work area partition `BPECompensation`.

  If you configured Business Process Choreographer on a cluster, repeat this step for each member of the cluster.

j. Optional: Remove the mail session.

  1) Click **Resources** → **Mail** → **Mail Providers**.

  2) For **Scope**, select the **Cell=***cellName*, where *cellName* is the name of the cell.

  3) Click **Built-in Mail Provider**.

4) Under the **Additional Properties** section, select **Mail sessions**.

5) Select and delete HTMMailSession_*scope*, where *scope* is the scope identified in step 1b on page 274

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**, under the **Topology** section, click **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, *nodeName.serverName* or *clusterName* will not match the name deployment target where you configured Business Process Choreographer.

   c. Under **Additional Properties**, select **Message store**.
      - It the message store type is DATASTORE, note the JNDI name for the data source. On a server the JNDI name of the data source is usually jdbc/com.ibm.ws.sib/*nodeName.serverName*-BPC.*cellName*.Bus″. On a cluster the JNDI name of the data source is usually jdbc/com.ibm.ws.sib/*clusterName*-BPC.*cellName*.Bus.
      - If the message store type is FILESTORE, note the paths for Log, Permanent store, and Temporary store.

   d. Go to **Service integration** → **Buses** → **BPC.*cellName*.Bus**, under the **Topology** section, click **Bus members** and remove the bus member identified by one of the following names:
      - *nodeName*:*serverName* if you configured Business Process Choreographer on a server.
      - *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. If the message store type that you noted in step 3c was DATASTORE, then click **Resources** → **JDBC** → **Data Sources**. The scope of the messaging engine might not be the same as the deployment target where you configured Business Process Choreographer. If necessary, trying different scopes, look for the JNDI name that you noted in step 3c. If the data source is for a Derby database, note the file system path for the database. If you configured Business Process Choreographer on a cluster, repeat this step for each member of the cluster.

4. Delete the BPC_REMOTE_DESTINATION_LOCATION variable. Click **Environment** → **WebSphere Variables**, for **Scope** select the deployment target where Business Process Choreographer was configured, then select and delete the variable BPC_REMOTE_DESTINATION_LOCATION variable.

5. Restart the application server or cluster.

6. Click **Save** to save all your deletions in the master configuration.

7. Optional: Delete the Business Process Choreographer database.

8. Optional: If you are using WebSphere MQ, delete the queue manager used by Business Process Choreographer.

9. If you use WebSphere default messaging for Business Process Choreographer, delete the message store for the message engine; because it cannot be reused.

   a. If the message store type that you noted in step 3c on page 278 was `FILESTORE`, remove the directories that you noted for `Log`, `Permanent store`, and `Temporary store`.

   b. If the message store type that you noted in step 3c on page 278 was `DATASTORE`, remove the database to which the data source pointed. If this was a Derby data source, then delete the file system path that you noted in step 3f on page 278. Usually, the Derby database location is the following:

     • On Linux, UNIX, and i5/OS platforms:

      *profile_root*/databases/com.ibm.ws.sib/
        *nodeName.serverName*-BPC.*cellName*.Bus

     • On Windows platforms:

      *profile_root*\databases\com.ibm.ws.sib\
        *nodeName.serverName*-BPC.*cellName*.Bus

10. Optional: If you configured the Business Process Choreographer Observer, perform:

   a. "Using the administrative console to remove Business Process Choreographer Observer" on page 281 for each observer application instance.

   b. "Using the administrative console to remove the Business Process Choreographer event collector" for each event collector application instance.

**Results**

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.

**Procedure**

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 for `BPCECollector_scope`, click **Uninstall** → **OK**. Where *scope* identifies the server or cluster where the event collector was configured.

3. Delete the destination queues:

   a. Click **Service integration** → **Buses** → **CommonEventInfrastructure_Bus** .

   b. Under **Destination resources**, click **Destinations**.

   c. Select the following destination queues:

     • BPCCEIConsumerQueueDestination_*scope*

     • BPCTransformerQueueDestination_*scope*

   Where *scope* identifies the server or cluster where the event collector was configured.

    d. Click **Delete**.

4. Delete the JMS queue connection factory:

    a. Click **Resources** → **JMS** → **Queue connection factories**.

    b. For **Scope**, select the server or cluster where the event collector was configured.

    c. Select the check box for BPCCEIConsumerQueueConnectionFactory.

    d. Click **Delete**.

5. Delete the JMS queues:

    a. Click **Resources** → **JMS** → **Queues**.

    b. Select the check boxes for the following queues:

      • BPCCEIConsumerQueue_*scope*

      • BPCTransformerQueue_*scope*

    c. Click **Delete**.

6. Delete the JMS activation specifications:

    a. Click **Resources** → **JMS** → **Activation specifications**.

    b. Select the check boxes for the following activation specifications:

      • BPCCEIConsumerActivationSpec

      • BPCTransformerActivationSpec

    c. Click **Delete**.

7. Delete the event profile group with server scope for BFMEvents:

    a. Click **Service integration** → **Common Event Infrastructure** → **Event service**.

    b. Under **Additional Properties** click **Event services** .

    c. Click **Default Common Event Infrastructure event server**.

    d. Under **Additional Properties** click **Event groups** .

    e. Select the check box for BFMEvents.

    f. Click **Delete**.

8. If you migrated your configuration from version 6.0.2, delete the authentication data alias:

    a. Click **Security** → **Secure administration, applications, and infrastructure** → **Authentication** → **Java Authentication and Authorization Service** → **J2C authentication data**.

    b. select BPCEventCollectorJMSAuthenticationAlias_*scope*.

    c. Click **Delete**.

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, UNIX, and i5/OS platforms in the directory *install_root*/dbscripts/ProcessChoreographer/*database_type*:

    • dropSchema_Observer.sql

    • dropTablespace_Observer.sql

**Results**

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.

**Procedure**

1. Display the enterprise applications.

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

2. Locate the Business Process Choreographer Observer instances.

   Look for applications whose names start with 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*.

   **Note:** If the context root is not the default /bpcobserver, the application name also has the context root appended to it, *_contextRoot*.

3. Uninstall the Business Process Choreographer Observer application. Select the application instance that you want to deleted, then click **Uninstall** → **OK** → **Save**.

**Results**

The Business Process Choreographer Observer configuration has been removed.

# Part 3. Administering

# Chapter 6. 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 server starts, and to specify the location and maximum size of the recovery log.

**About this task**

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. When you set up a new application server, the compensation service is enabled by default.

You can use the administrative console to view and change properties of the compensation service for application servers.

**Procedure**

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. Make sure that the **Enable service at server startup** check box is selected. If you run your business processes on a cluster, enable the compensation service for each server in the cluster.
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.

### 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.

**About this task**

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.

**Procedure**

1. To check how many messages are in the hold and retention queues:

   a. Click **Servers** → **Application servers** → *server_name*, or **Servers** → **Clusters** → *cluster_name* if Business Process Choreographer is configured on a cluster.

   b. On the **Configuration** tab, under Business Integration, click **Business Process Choreographer**. Choose one of the following options:

      • For business processes, click **Business Flow Manager**.

      • For human tasks, click **Human Task Manager**.

      The number of messages in the hold queue and retention queue are displayed on the **Runtime** tab 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 WebSphere administrative security is enabled, the replay buttons are only visible to users who have operator authority.

**Results**

Business Process Choreographer tries to service all replayed messages again.

   **Related concepts**

   "Recovery from infrastructure failures" on page 30
   Business Flow Manager provides a facility for handling temporary infrastructure failures.

## Refreshing the failed message counts

Use the administrative console to refresh the count of failed messages for business processes or human tasks.

**About this task**

The displayed number of messages in the hold queue and in 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.

**Procedure**

1. Select the appropriate application server.

   Click **Servers** → **Application servers** → *server_name*, or **Servers** → **Clusters** → *cluster_name* if Business Process Choreographer is configured on a cluster.

2. Refresh the message counts.

   a. On the **Configuration** tab, under Business Integration, click **Business Process Choreographer**. Choose one of the following options:

      • For business processes, click **Business Flow Manager**.

      • For human tasks, click **Human Task Manager**.

b. On the **Runtime** tab, click **Refresh Message Count**.

**Results**

The following updated values are displayed under **General Properties**:
- For business processes: The number of messages in the hold queue and in the retention queue
- For human tasks: The number of messages in the hold queue
- If any exceptions occurred while accessing the queues, the message text is displayed in the Message exceptions field.

**What to do next**

On this page, you can also replay the messages in these queues.

> **Related concepts**
>
> "Recovery from infrastructure failures" on page 30
> Business Flow Manager provides a facility for handling temporary infrastructure failures.

# Refreshing people query results, using the administrative console

The results of a people query are static. Use the administrative console to refresh people queries.

**About this task**

Business Process Choreographer caches the results of people queries, which have been evaluated against a people directory, such as a Lightweight Directory Access Protocol (LDAP) server, in the runtime database. If the people directory changes, you can force the people assignments to be evaluated again.

**Procedure**

To refresh the people queries:
1. Click **Servers** → **Application servers** → *server_name*, or **Servers** → **Clusters** → *cluster_name* if Business Process Choreographer is configured on a cluster.
2. On the **Configuration** tab, under Business Integration, click **Business Process Choreographer** → **Human Task Manager**.
3. On the **Runtime** tab, click **Refresh People Queries**. All people queries are refreshed.

**Note:** When WebSphere administrative security is enabled, the refresh button is only visible to users who have operator authority.
Refreshing the people query results in this way can cause a high load on the application and database. Consider using the alternative methods listed below.

**Results**

> **Related concepts**
>
> "Managing people assignment criteria and people resolution results" on page 86
> A people assignment criteria that is associated with a task authorization role is valid throughout the lifetime of a deployed task template or task instance.

**Related tasks**

"Refreshing people query results, using administrative commands" on page 306
The results of a people query are static. Use the administrative commands to
refresh people queries.

"Refreshing people query results, using the refresh daemon" on page 289
Use this method if you want to set up a regular and automatic refresh of all
expired people query results.

# Enabling Common Base Events and the audit trail, using the administrative console

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.

**About this task**

You can change the state observers settings for the Business Flow Manager or the
Human Task Manager, 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 Flow Manager and the Human Task Manager, you must change
the settings separately for them both.

## Changing the configured logging infrastructure, using the administrative console

Use this task to change the state observer logging for the audit log or common
event infrastructure logging for the configuration.

**About this task**

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:

**Procedure**

1. Display the Business Flow Manager or Human Task Manager pane.
   a. Click **Servers** → **Application servers** → *server_name*, or **Servers** → **Cluster** →
      *cluster_name* if Business Process Choreographer is configured on a cluster.
   b. On the **Configuration** tab, under Business Integration, click **Business
      Process Choreographer**. Choose one of the following options:
      - For business processes, click **Business Flow Manager**.
      - For human tasks, click **Human Task Manager**.
2. On the **Configuration** tab, in the General Properties section, select the logging
   to be enabled. 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 Business Process Choreographer database.

3. Accept the change.
    a. Click **OK**.
    b. In the Messages box, click **Save**.

**Results**

The state observers are set, as you required. The changes take place after server restart.

**What to do next**

Restart the server, to effect the changes. If Business Process Choreographer is configured on a cluster, restart the cluster.

## Configuring the logging infrastructure for the session, using the administrative console

Use this task to change the state observer logging for the audit log or common event infrastructure logging for the session.

**About this task**

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:

**Procedure**

1. Display the Business Flow Manager or Human Task Manager pane.
    a. Click **Servers** → **Application servers** → *server_name*, or **Servers** → **Clusters** → *cluster_name* if Business Process Choreographer is configured on a cluster.
    b. On the **Configuration** tab, under Business Integration, click **Business Process Choreographer**. Choose one of the following options:
        • For business processes, click **Business Flow Manager**.
        • For human tasks, click **Human Task Manager**.
2. On the **Runtime** tab, in the General Properties section, select the logging to be enabled. 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 Business Process Choreographer database.
3. Click **OK** to accept the change.

**Results**

The state observers are set, as you required.

# Refreshing people query results, using the refresh daemon

Use this method if you want to set up a regular and automatic refresh of all expired people query results.

**About this task**

People queries are resolved by the specified people directory provider. 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 people query refresh daemon is invoked.

In order to keep people query results up to date, a daemon is provided that refreshes expired people query results on a regular schedule. The daemon refreshes all cached people query results that have expired.

**Procedure**

1. Open the custom properties page for the Human Task Manager:
   a. Click **Servers** → **Application servers** → *server_name*, or **Servers** → **Clusters** → *cluster_name* if Business Process Choreographer is configured on a cluster.
   b. On the **Configuration** tab, under Business Integration, click **Business Process Choreographer** → **Human Task Manager**.
   c. Choose one of the following options:
      • To change the settings permanently, click the **Configuration** tab. The changes are valid after the application server is restarted.
      • To change the settings temporarily, click the **Runtime** tab. The changes are valid immediately, but will be reset next time the application server restarts.
2. In the field **People 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 people query results. The default value is ″0 0 1 * * ?″, which causes a refresh every day at 1 am.
3. In the field **Timeout for people query result** enter a new value in seconds. This value determines how long a people query result is considered to be valid. After this time period, the people query result is considered to be no longer valid, and the people query will be refreshed the next time that the daemon runs. The default is one hour.
4. Click **OK**.
5. Save the changes. To make your changes that you performed on the Configuration tab effective, restart the application server.

   The new expiration time value applies only to new people queries, it does not apply to existing people queries.

   **Related concepts**

   "Managing people assignment criteria and people resolution results" on page 86
   A people assignment criteria that is associated with a task authorization role is valid throughout the lifetime of a deployed task template or task instance.

# Using scripts to administer Business Process Choreographer

Describes the administrative actions that can be performed using scripts.

**Related information**

"Using a script to enable logging for the Business Process Choreographer" on page 255
This describes how to use the setStateObserver.py script to enable or disable Common Event Infrastructure (CEI) or audit events for Business Process Choreographer.

# Deleting audit log entries, using administrative commands

Use the administrative commands to delete some or all audit log entries for the Business Flow Manager.

**Before you begin**

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.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- When WebSphere administrative security is enabled, you must have operator authority.

**About this task**

You can use the deleteAuditLog.py script to delete audit log entries for the Business Flow Manager from the database.

**Procedure**

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

    On Windows platforms, enter:

    cd *install_root*\ProcessChoreographer\admin

    On Linux, UNIX, and i5/OS platforms, enter:

    cd *install_root*/ProcessChoreographer/admin

2. Delete the entries in the audit log table.

    On Windows platforms, 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 *server_name***
                            [-profileName *profileName*]
                            [options]

    *install_root*\bin\wsadmin —lang jython -f deleteAuditLog.py
                            **-node *nodeName***
                            **-server *server_name***
                            [-profileName *profileName*]
                            [options]

    *install_root*\bin\wsadmin —lang jython -f deleteAuditLog.py
                            **-cluster *cluster_name***
                            [-profileName *profileName*]
                            [options]

    On Linux and UNIX platforms, enter one or more of the following commands. The differences between the commands are emphasized:

    *install_root*/bin/wsadmin.sh —lang jython -f deleteAuditLog.py
                            **-server *server_name***
                            [-profileName *profileName*]
                            [options]

    *install_root*/bin/wsadmin.sh —lang jython -f deleteAuditLog.py
                            **-node *nodeName***
                            **-server *server_name***
                            [-profileName *profileName*]

```
                              [options]

install_root/bin/wsadmin.sh –lang jython -f deleteAuditLog.py
                              -cluster cluster_name
                              [-profileName profileName]
                              [options]
```

On i5/OS platforms, 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 server_name
                              [-profileName profileName]
                              [options]

install_root/bin/wsadmin –lang jython -f deleteAuditLog.py
                              -node nodeName
                              -server server_name
                              [-profileName profileName]
                              [options]

install_root/bin/wsadmin –lang jython -f deleteAuditLog.py
                              -cluster cluster_name
                              [-profileName profileName]
                              [options]
```

Where:

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Flow Manager 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** *server_name*
> 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 people 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 that are not valid

Use the administrative commands to delete, from the Business Process Choreographer database, process templates that are no longer valid.

**Before you begin**

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 WebSphere administrative security is enabled. If the Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.

**About this task**

Use the deleteInvalidProcessTemplate.py script to remove, from the database, those templates, and all objects that belong to them, that are not contained in any corresponding valid application in the WebSphere configuration repository. This situation can occur if an application installation was canceled or not stored in 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.

**Procedure**
1. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

   On Windows platforms, enter:

   `cd install_root\ProcessChoreographer\admin`

   On Linux, UNIX, and i5/OS platforms, enter:

   `cd install_root/ProcessChoreographer/admin`

2. Delete, from the database, business process templates that are no longer valid.

   To delete, on Windows systems, 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 –lang jython -f deleteInvalidProcessTemplate.py
                   -server server_name
                   -templateName templateName
   ```

```
                                  -validFrom validFromString
                                  [-profileName profileName]
```

*install_root*\bin\wsadmin –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -server server_name
                                  -node nodeName
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

*install_root*\bin\wsadmin –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -cluster cluster_name
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

To delete, on Linux and UNIX systems, 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 server_name
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

*install_root*/bin/wsadmin.sh –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -server server_name
                                  -node nodeName
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

*install_root*/bin/wsadmin.sh –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -cluster cluster_name
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

To delete, on i5/OS systems, 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 –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -server server_name
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

*install_root*/bin/wsadmin –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -server server_name
                                  -node nodeName
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

*install_root*/bin/wsadmin –lang jython -f deleteInvalidProcessTemplate.py
```
                                  -cluster cluster_name
                                  -templateName templateName
                                  -validFrom validFromString
                                  [-profileName profileName]
```

Where:

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Process Choreographer 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** *server_name*
    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.

**-templateName** *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 must have the following format:
    'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds).
    For example, `2005-01-31T13:40:50`

**-profileName** *profileName*
    The name of a user-defined profile. Specify this option if you are not
    working with the default profile.

**What to do next**

**Note:** The jacl version of the cleanup unused processes script,
`deleteInvalidProcessTemplate.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 human task templates that are not valid

Use the administrative commands to delete, from the Business Process
Choreographer database, human task templates that are no longer valid.

**Before you begin**

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 WebSphere administrative security is enabled. If the
Business Process Choreographer is configured on a cluster, then at least one cluster
member must be running.

**About this task**

Use the deleteInvalidTaskTemplate.py script to remove, from the database, those
templates, and all objects that belong to them, that are not contained in any
corresponding valid application in the WebSphere configuration repository. This
situation can occur if an application installation was canceled or not stored in 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.

**Procedure**

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

   On Windows platforms, enter:

   `cd install_root\ProcessChoreographer\admin`

   On Linux, UNIX, and i5/OS platforms, enter:

   `cd install_root/ProcessChoreographer/admin`

2. Delete, from the database, human task templates that are no longer valid.

   To delete, on Windows systems, 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 –lang jython -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                             [-profileName profileName]
   ```

   ```
   install_root\bin\wsadmin –lang jython -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -node nodeName
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]
   ```

   ```
   install_root\bin\wsadmin –lang jython -f deleteInvalidTaskTemplate.py
                            -cluster cluster_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]
   ```

   To delete, on UNIX and Linux systems, 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 server_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]
   ```

   ```
   install_root/bin/wsadmin.sh –lang jython -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -node nodeName
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]
   ```

   ```
   install_root/bin/wsadmin.sh –lang jython -f deleteInvalidTaskTemplate.py
                            -cluster cluster_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]
   ```

   To delete, on i5/OS systems, 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 —lang jython -f deleteInvalidTaskTemplate.py
                    -server server_name
                    -templateName templateName
                    -validFrom validFromString
                    -nameSpace nameSpace
                    [-profileName profileName]

install_root/bin/wsadmin —lang jython -f deleteInvalidTaskTemplate.py
                    -server server_name
                    -node nodeName
                    -templateName templateName
                    -validFrom validFromString
                    -nameSpace nameSpace
                    [-profileName profileName]

install_root/bin/wsadmin —lang jython -f deleteInvalidTaskTemplate.py
                    -cluster cluster_name
                    -templateName templateName
                    -validFrom validFromString
                    -nameSpace nameSpace
                    [-profileName profileName]
```

Where:

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Process Choreographer 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** *server_name*
> 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.

**-templateName** *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 must 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.

**What to do next**

**Note:** The jacl version of the cleanup unused people 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**

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 WebSphere administrative security is enabled. If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.

**About this task**

A top-level process instance is considered completed if it is in one of the following end states: finished, terminated, end, 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.

**Procedure**

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

   On Windows platforms, enter:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

   On Linux, UNIX, and i5/OS platforms, enter:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. Delete process instances from the database.

   On Windows systems, enter the following command:

   ```
   install_root\bin\wsadmin —lang jython —f deleteCompletedProcessInstances.py
     [([-node nodeName] -server server_name) | (-cluster cluster_name)]
     (-all | -finished | -terminated | -failed )
     [-templateName templateName [-validFrom timestamp]]
     [-startedBy userID ]
     [-completedBefore timestamp]
     [-profileName profileName]
   ```

   On Linux and UNIX systems, enter the following command:

   ```
   install_root/bin/wsadmin.sh —lang jython —f deleteCompletedProcessInstances.py
     [([-node nodeName] -server server_name) | (-cluster cluster_name)]
     (-all | -finished | -terminated | -failed )
     [-templateName templateName [-validFrom timestamp]]
     [-startedBy userID ]
     [-completedBefore timestamp]
     [-profileName profileName]
   ```

   On i5/OS systems, enter the following command:

   ```
   install_root/bin/wsadmin —lang jython —f deleteCompletedProcessInstances.py
     [([-node nodeName] -server server_name) | (-cluster cluster_name)]
     (-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** *server_name*

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** *cluster_name*

The name of the cluster. Required if the Business Process Choreographer 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 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-dd[Thh:mm:ss]' (year, month, day, T, hours, minutes, seconds). For example, 2006-11-20T12:00:00. If you specify only the year, month, and day, the hour, minutes, and seconds are set to 00:00:00.

**-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
```

**Results**

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. Deleting unnecessary data can improve the performance generating reports.

**About this task**

You can delete the observer information for process instances in three ways:

- To delete observer data for process instances that have reached the end state deleted before a specified time, you must provide the following parameters: `-deletedBefore` *timestamp*.
- To delete observer data for process instances of a specific template version regardless of its current state, you must provide the following parameters: `-templateName` *templateName* `-validFrom` *timestamp*.
- To delete observer data for process instances of a specific template version that reached a specified state before a specified time; you must provide the following parameters: `-force -templateName` *template_name* `-validFrom` *timestamp* `-state` *state* `-reachedBefore` *timestamp,* where `-templateName` *template_name* and `-validFrom` *timestamp* are optional.

To use any one of these ways, perform the following:

**Procedure**

1. Make sure that the application server is running. If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.

2. Change to the Business Process Choreographer subdirectory where the administration scripts are located.

   On Windows platforms, enter:

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

   On Linux, UNIX, and i5/OS platforms, enter:

   `cd` *install_root*`/ProcessChoreographer/admin`

3. Enter the command to delete observer data for specific process instances from the database.

   On Windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin -lang jython
       -f observerDeleteProcessInstanceData.py
       [ -user user_ID -password password ]
       ( [-node node_name] -server server_name) | (-cluster cluster_name )
       [ -profileName profile_name ]
       [ -dataSource dataSource_JNDI_name ]
       [ -dbSchemaName dbSchemaName]
       (
         -deletedBefore timestamp
         | ( -templateName template_name  -validFrom timestamp )
         | ( -force [-templateName template_name  -validFrom timestamp]
             -state state -reachedBefore timestamp )
       )
   ```

   On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh -lang jython
       -f observerDeleteProcessInstanceData.py
       [ -user user_ID -password password ]
       ( [-node node_name] -server server_name) | (-cluster cluster_name )
       [ -profileName profile_name ]
   ```

```
[ -dataSource dataSource_JNDI_name ]
[ -dbSchemaName dbSchemaName]
(
  -deletedBefore timestamp
  | ( -templateName template_name  -validFrom timestamp )
  | ( -force [-templateName template_name  -validFrom timestamp]
      -state state -reachedBefore timestamp )
)
```

On i5/OS platforms, enter the following command:

```
install_root/bin/wsadmin -lang jython
    -f observerDeleteProcessInstanceData.py
    [ -user user_ID -password password ]
    ( [-node node_name] -server server_name) | (-cluster cluster_name )
    [ -profileName profile_name ]
    [ -dataSource dataSource_JNDI_name ]
    [ -dbSchemaName dbSchemaName]
    (
      -deletedBefore timestamp
      | ( -templateName template_name  -validFrom timestamp )
      | ( -force [-templateName template_name  -validFrom timestamp]
          -state state -reachedBefore timestamp )
    )
```

Enter the following command:

```
install_root/bin/wsadmin.sh -lang jython
    -f observerDeleteProcessInstanceData.py
    [ -user user_ID -password password ]
    ( [-node node_name] -server server_name) | (-cluster cluster_name )
    [ -profileName profile_name ]
    [ -dataSource dataSource_JNDI_name ]
    [ -dbSchemaName dbSchemaName]
    (
      -deletedBefore timestamp
      | ( -templateName template_name  -validFrom timestamp )
      | ( -force [-templateName template_name  -validFrom timestamp]
          -state state -reachedBefore timestamp )
    )
```

Where:

**-user** *user_ID* **-password** *password*
>   If WebSphere administrative security is enabled on the server, you must
>   provide a user ID that has permission to delete from the database, and the
>   associated password.

**-node** *node_name*
>   This name identifies the node. This parameter is optional. The default is the
>   local node.

**-server** *server_name*
>   The name of the server. The default is the default server. If you specify this
>   parameter, you must not specify the cluster parameter.

**-cluster** *cluster_name*
>   The name of the cluster. If you specify this parameter, you must not specify
>   the server parameter.

**-profileName** *profile_name*
>   The name of a user-defined WebSphere profile. Specify this option if you
>   are not working with the default profile.

**-dataSource** *datasource_JNDI_name*
>   Because a server or cluster can have multiple observer databases, this
>   parameter identifies which database the command will act on. The default
>   is jdbc/BPEDB.

**-dbSchemaName** *dbSchemaName*
>    Use this parameter if the Observer database is set up with a specific
>    schema name.

**-deletedBefore** *timestamp*
>    Deletes all observer data for process instances that reached the state deleted
>    before the specified time.
>
>    *timestamp*
>>        The date and time are expressed in Coordinated Universal Time (UTC),
>>        in the following format: '*yyyy-MM-dd*[T*hh:mm:ss*]' (year, month, day, T,
>>        hours, minutes, seconds). For example, 2008-07-20T12:00:00. If you
>>        specify only the year, month, and day, the hour, minutes, and seconds
>>        are set to 00:00:00.

**-templateName** *template_name*
>    Deletes all observer data for instances that belong to the specified template
>    version.
>
>    **-validFrom** *timestamp*
>>        This is required if you specify the templateName option.
>>
>>        *timestamp*
>>>            The date and time are expressed in Coordinated Universal Time
>>>            (UTC), in the following format: '*yyyy-MM-dd*T*hh:mm:ss*' (year,
>>>            month, day, T, hours, minutes, seconds). For example,
>>>            2008-07-20T12:00:00.

**-force**
>    Forces the deletion of all observer data for process instances either for all
>    templates or for a specified template version that reached a specified state
>    before a specified time. If you use this option, you must also specify the
>    options -state, and -reachedBefore. The options -templateName and
>    -validFrom are optional.
>
>    **-state** *state*
>>        Specifies one of the following states: running, terminated, suspended,
>>        failed, finished, compensated.
>
>    **-reachedBefore** *timestamp*
>>        Specifies a time when the specified state must have been reached.
>>
>>        *timestamp*
>>>            The date and time are expressed in Coordinated Universal Time
>>>            (UTC), in the following format: '*yyyy-MM-dd*[T*hh:mm:ss*]' (year,
>>>            month, day, T, hours, minutes, seconds). For example,
>>>            2008-07-20T12:00:00. If you specify only the year, month, and day,
>>>            the hour, minutes, and seconds are set to 00:00:00.

For example, to delete all observer data for instances of the process template
*my_template*, which is valid from midday on January 2, 2007, that are running
on node *my_node* in server *my_server* that were started before midday on July
20, 2007, run the following command:

```
wsadmin —lang jython —f observerDeleteProcessInstanceData.py
        -node my_node -server my_server
        -force -templateName my_template -validFrom 2007-01-02T12:00:00
              -state running -reachedBefore 2007-07-20T12:00:00
```

**Results**

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.

# 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**

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.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- When WebSphere administrative security is enabled, you must have operator authority.

**About this task**

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:

**Procedure**

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

   On Windows platforms, enter:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

   On Linux, UNIX, and i5/OS platforms, enter:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. Query the number of failed messages on both the retention and hold queues.

   On Windows systems, enter one of the following commands. The differences between the commands are emphasized:

   ```
   install_root\bin\wsadmin —lang jython -f queryNumberOfFailedMessages.py
                            -cluster cluster_name
                            [ -bfm | -htm ]
                            [-profileName profileName]
   ```

   ```
   install_root\bin\wsadmin —lang jython -f queryNumberOfFailedMessages.py
                            -node nodeName
                            -server server_name
                            [ -bfm | -htm ]
                            [-profileName profileName]
   ```

   On Linux and UNIX systems, enter one of the following commands. The differences between the commands are emphasized:

   ```
   install_root/bin/wsadmin.sh —lang jython -f queryNumberOfFailedMessages.py
                            -cluster cluster_name
                            [ -bfm | -htm ]
                            [-profileName profileName]
   ```

   ```
   install_root/bin/wsadmin.sh —lang jython -f queryNumberOfFailedMessages.py
   ```

```
                                   -node nodeName
                                   -server server_name
                                   [ -bfm | -htm ]
                                   [-profileName profileName]
```

On i5/OS systems, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin –lang jython -f queryNumberOfFailedMessages.py
                                   -cluster cluster_name
                                   [ -bfm | -htm ]
                                   [-profileName profileName]


install_root/bin/wsadmin –lang jython -f queryNumberOfFailedMessages.py
                                   -node nodeName
                                   -server server_name
                                   [ -bfm | -htm ]
                                   [-profileName profileName]
```

Where:

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Process Choreographer 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** *server_name*
> 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 Flow Manager hold and retention queues, specify the -bfm option. If you only want to display the number of messages in the Human Task Manager hold queue, specify the -htm option.

**-profileName** *profileName*
> The name of a user-defined profile. Specify this option if you are not working with the default profile.

If you want to check for a server on the local node, enter:

```
wsadmin –lang jython -f queryNumberOfFailedMessages.py -server server_name
```

3. Replay all failed messages on the hold queue, retention queue, or both queues.

On Windows systems, enter one of the following commands. The differences between the commands are emphasized:

```
install_root\bin\wsadmin –lang jython -f replayFailedMessages.py
                                   -cluster cluster_name
                                   -queue replayQueue
                                   [ -bfm | -htm ]
                                   [-profileName profileName]


install_root\bin\wsadmin –lang jython -f replayFailedMessages.py
                                   -node nodeName
                                   -server server_name
                                   -queue replayQueue
                                   [ -bfm | -htm ]
                                   [-profileName profileName]


install_root\bin\wsadmin –lang jython -f replayFailedMessages.py
                                   -server server_name
```

```
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]
```

On Linux and UNIX systems, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin.sh –lang jython -f replayFailedMessages.py
                    -cluster cluster_name
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]


install_root/bin/wsadmin.sh –lang jython -f replayFailedMessages.py
                    -node nodeName
                    -server server_name
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]


install_root/bin/wsadmin.sh –lang jython -f replayFailedMessages.py
                    -server server_name
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]
```

On i5/OS systems, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin –lang jython -f replayFailedMessages.py
                    -cluster cluster_name
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]


install_root/bin/wsadmin –lang jython -f replayFailedMessages.py
                    -node nodeName
                    -server server_name
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]


install_root/bin/wsadmin –lang jython -f replayFailedMessages.py
                    -server server_name
                    -queue replayQueue
                    [ -bfm | -htm ]
                    [-profileName profileName]
```

Where:

**-queue** *replayQueue*
> Optionally specifies the queue to replay. *replayQueue* can have one of the following values:
>
>> holdQueue (this is the default value)
>>
>> retentionQueue (only valid when the -bfm option is specified)
>>
>> both (not valid when the -htm option is specified)

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Process Choreographer 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** *server_name*
> The name of the server. Required if the cluster name is not specified.

Chapter 6. Administering Business Process Choreographer    **305**

**-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 option.

**-profileName** *profileName*

The name of a user-defined profile. Specify this option if you are not working with the default profile.

**What to do next**

**Note:** The jacl version of the cleanup unused people 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.

**Related concepts**

"Recovery from infrastructure failures" on page 30
Business Flow Manager provides a facility for handling temporary infrastructure failures.

# Refreshing people query results, using administrative commands

The results of a people query are static. Use the administrative commands to refresh people queries.

**Before you begin**

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.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- When WebSphere administrative security is enabled, you must have operator authority.

**About this task**

Business Process Choreographer caches the results of people queries, which have been evaluated against a people directory, such as a Lightweight Directory Access Protocol (LDAP) server, in the runtime database. If the people directory changes, you can force the people assignments to be evaluated again.

**Procedure**

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

   On Windows platforms, enter:

   `cd install_root\ProcessChoreographer\admin`

   On Linux, UNIX, and i5/OS platforms, enter:

   `cd install_root/ProcessChoreographer/admin`

2. Force the people assignment to be evaluated again.

On Windows platforms, enter one of the following commands. The differences between the commands are emphasized:

```
install_root\bin\wsadmin —lang jython -f refreshStaffQuery.py
        -server server_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]

install_root\bin\wsadmin —lang jython -f refreshStaffQuery.py
        -node nodeName
         -server server_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]

install_root\bin\wsadmin —lang jython -f refreshStaffQuery.py
        -cluster cluster_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]
```

On Linux and UNIX platforms, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin.sh —lang jython -f refreshStaffQuery.py
        -server server_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]

install_root/bin/wsadmin.sh —lang jython -f refreshStaffQuery.py
        -node nodeName
         -server server_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]

install_root/bin/wsadmin.sh —lang jython -f refreshStaffQuery.py
        -cluster cluster_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]
```

On i5/OS platforms, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin —lang jython -f refreshStaffQuery.py
        -server server_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]

install_root/bin/wsadmin —lang jython -f refreshStaffQuery.py
        -node nodeName
         -server server_name
        [-processTemplate templateName |
        (-taskTemplate templateName [-nameSpace nameSpace]) |
         -userlist username{,username}...]
        [-profileName profileName]

install_root/bin/wsadmin —lang jython -f refreshStaffQuery.py
```

```
 -cluster cluster_name
[-processTemplate templateName |
(-taskTemplate templateName [-nameSpace nameSpace]) |
 -userlist username{,username}...]
[-profileName profileName]
```

Where:

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Process Choreographer 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** *server_name*
> The name of the server. Required if the cluster name is not specified.

**-processTemplate** *templateName*
> The name of the process template. People assignments that belong to this process template are refreshed.

**-taskTemplate** *templateName*
> The name of the task template. People 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. People assignments that contain the specified names are refreshed. The user list can be surrounded by quotes. If the quotes are omitted the user list must not contain blanks between the user names.

**-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 people 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 people 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.

**Related concepts**

"Managing people assignment criteria and people resolution results" on page 86
A people assignment criteria that is associated with a task authorization role is valid throughout the lifetime of a deployed task template or task instance.

# Removing unused people query results, using administrative commands

Use the administrative commands to remove unused people query results from the database.

**Before you begin**

Before you begin this procedure, the following conditions must be met:

- The application server, through which unused people 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.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- When WebSphere administrative security is enabled, you must have operator authority.

**About this task**

Business Process Choreographer maintains lists of user names in the runtime database for people queries that have been evaluated. Although the process instances and human tasks that used the people queries 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 lists of people that are cached in the database tables.

**Procedure**

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

   On Windows platforms, enter:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

   On Linux, UNIX, and i5/OS platforms, enter:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. Remove the unused lists of people.

   On Windows platforms, enter one of the following commands. The differences between the commands are emphasized:

   ```
   install_root\bin\wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -server server_name
                         [-profileName profileName]
   ```

   ```
   install_root\bin\wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -node nodeName
                         -server server_name
                         [-profileName profileName]
   ```

   ```
   install_root\bin\wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -cluster cluster_name
                         [-profileName profileName]
   ```

   On Linux and UNIX platforms, enter one of the following commands. The differences between the commands are emphasized:

   ```
   install_root/bin/wsadmin.sh –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -server server_name
                         [-profileName profileName]
   ```

   ```
   install_root/bin/wsadmin.sh –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -node nodeName
                         -server server_name
                         [-profileName profileName]
   ```

   ```
   install_root/bin/wsadmin.sh –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -cluster cluster_name
                         [-profileName profileName]
   ```

On i5/OS platforms, enter one of the following commands. The differences between the commands are emphasized:

```
install_root/bin/wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -server server_name
                         [-profileName profileName]

install_root/bin/wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -node nodeName
                         -server server_name
                         [-profileName profileName]

install_root/bin/wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                         -cluster cluster_name
                         [-profileName profileName]
```

Where:

**-cluster** *cluster_name*
> The name of the cluster. Required if the Business Process Choreographer 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** *server_name*
> 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.

**Results**

The number of entries deleted from the database is displayed.

**What to do next**

**Note:** The jacl version of the script to cleanup unused people queries, `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.

# Chapter 7. Getting started with Business Process Choreographer Explorer

Depending on your user role, you can use Business Process Choreographer Explorer to manage business processes and human tasks, or work with your assigned tasks.

**About this task**

You can use Business Process Choreographer Explorer to perform the following tasks:

- If you are a business administrator, you can manage the lifecycle of business processes, and you can repair business processes. For example, you can restart or force the completion of single activities, or compensate the business process as a whole. If compensations failed, you can retry, skip or stop the process instances. In addition, you can add and update custom properties for business processes and activities.
- If you are a human task administrator, you can manage the life cycle of human tasks, and manage work assignments. For example, you can assign responsibility to users, or manage absence handling and substitutes for users. You can also change the priority and business category for human tasks, and add or update custom properties.
- If you are a business user, you can use Business Process Choreographer Explorer to work with your assigned tasks. For example, you can initiate business processes, services, and human tasks, and you can work on, edit, save, complete, or release human tasks. In addition, you can flag your absence and define substitutes.

Furthermore, Business Process Choreographer Explorer offers a search function that you can use to discover business processes and their related activities and human tasks that need attention. For example, you can check the status of these instances, navigate between related instances and templates, and retrieve a graphical view of the process states which includes the associated activities and human tasks.

**Related tasks**

"Administering task templates and task instances" on page 344
Use the administrative console or the administrative commands to administer task templates. Use Business Process Choreographer Explorer to work with task instances.

"Managing work assignments" on page 349
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.

"Creating and starting a task instance" on page 346
You can create and start a task instance from any of the task templates that you are authorized to use.

"Working on your tasks" on page 346
To work on a task, you must claim the task and then perform the actions that are needed to complete it.

# 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.

**Taskbar**

**Navigation pane**

**Page title**

**Action bar**

| Start | Stop | Complete |

**Content...**

| | | | | |
| --- | --- | --- | --- | --- |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
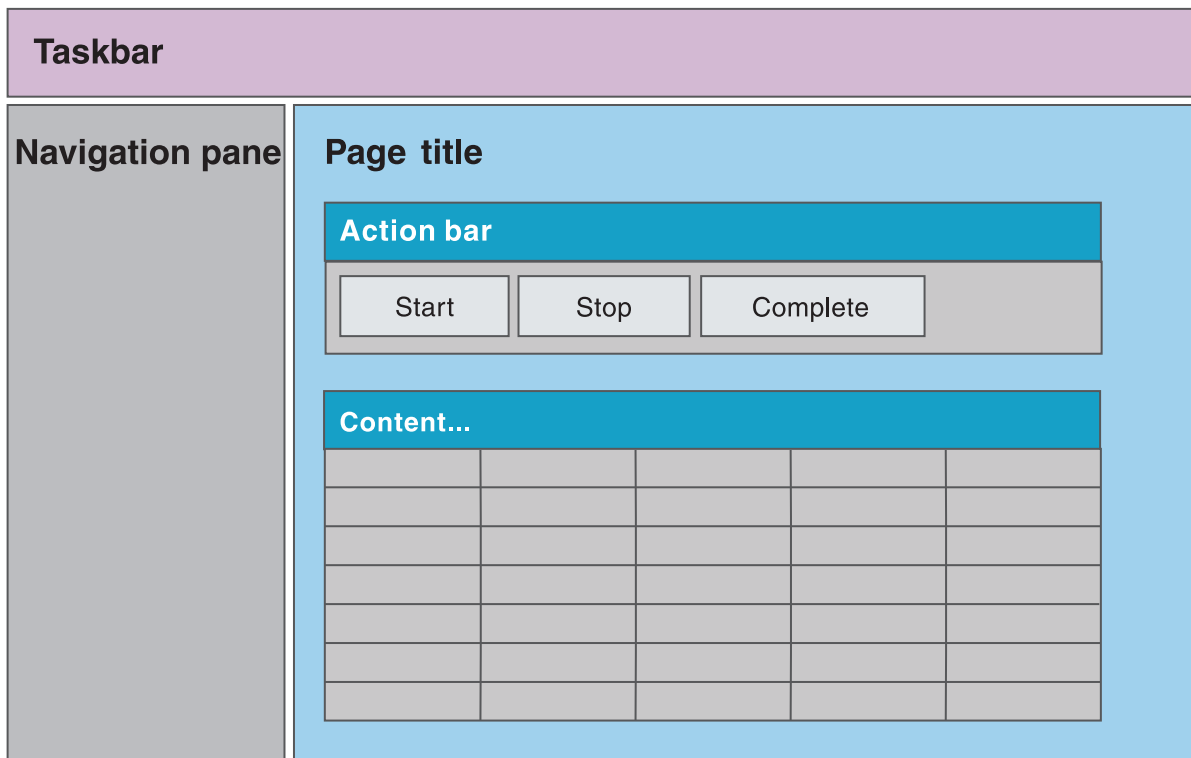
The user interface has the following main areas.

## Taskbar

For all users, the taskbar offers options for logging out of Business Process Choreographer Explorer, specifying absence settings, 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.

**Define Substitutes**
> Select this option to define absence settings for users.

### Navigation pane

The navigation pane 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 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 personalized views, which are added to the navigation pane. In addition, if you are a system administrator, you can define customized views that are available to all users.

### 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 arrow beside an item in the navigation pane to expand or collapse the item.
- Define a new search.

  Click the **New 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 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. If no view is defined for a group of views, the group is not displayed.

**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**
> By default, the activity instances group does not contain any views. Therefore this group is not displayed 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 To-dos**
>> 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.**

Business users can click the view name and navigate to the view. For system administrators, pop-up menus are available.

- 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 are only visible to the user who created the views. The user can delete, edit, copy, and move the views.

# Starting Business Process Choreographer Explorer

Business Process Choreographer Explorer is a Web application that can be 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.

**About this task**

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

**Procedure**

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**.

**Results**

The initial page of the Business Process Choreographer Explorer is displayed. By default, this is the page that shows the My To-dos 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 the interface for a specific Business Process Choreographer Explorer instance to address the business needs of user groups that are assigned to this instance.

**About this task**

You can customize the user interface in various ways.

## Customizing the Business Process Choreographer 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 To-dos 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.

**Before you begin**

To customize the interface, you must have BPCSystemAdministrator authorization.

**About this task**

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 those users who are assigned to the instance.

To customize the default user interface of Business Process Choreographer Explorer, complete the following steps.

**Procedure**

1. Customize the set of views in the navigation pane and the default login view.
   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 Search And Define Customized Views page (see step 2).
   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. 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 Customized Views page, select the type of view that you want to customize, for example, process templates.
   c. In the Search And Define Customized Views page for the view, specify the search criteria.
   d. Use the View Properties tab to select the properties to include in the view, and to specify the list properties.

      If this is a task instance view or a process instance view, click **View Settings** to select a set of actions to add to the action bar in the view, and to specify the items that are included in the view for system administrators and system monitors.
      - Select the view type:
        – To add administrative actions to the action bar in the view, select **Manage Instances**.
        – To add a set of actions to the action bar that enables the logged-on user to work with the instances, select **Work with Instances**.

- For system administrators and system monitors, you can limit the search result to their own instances:
  – To show all items that match the search criteria in the view, select **All Instances**. All of the items are shown regardless of whether the system administrator has work items for these items.
  – To show only the items that the logged-on user has work items for, select **Personal Instances**.
  e. Enter a display name for the view in the **View Name** field and click **Save**.

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

## Defining 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.

**Before you begin**

To create customized views, you must have BPCSystemAdministrator authorization.

**Procedure**

1. Click **Define Views** in the taskbar.
2. In the Search and Define Views page, select **Search For Process Templates And Define Customized Views**.
3. Click **Property Filters → Custom Property Filter**.
   a. Add a custom property with the following settings:
      - In the **Property Name** field, type generatedBy.
      - In the **Property Value** field, type BusinessStateMachine.
   b. Click **Add**.
   c. Add other custom properties as needed.
4. Click **View Properties → List Columns**.
   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 **Display Name** field, type a display name for the column, and click **Add**.
   b. Add other columns to or remove columns from the list of selected columns.
5. Type a display name for the query in the **View Name** field, and click **Save**.

**Results**

By default, a link to the new view is added to the Process Templates group in the navigation pane. Your users see this view the next time they log in to Business Process Choreographer Explorer.

## Defining 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.

**Before you begin**

To create customized views, you must have BPCSystemAdministrator authorization.

**Procedure**

1. Click **Define Views** in the taskbar.
2. In the Search and Define Views page, select **Search For Process Instances And Define Customized Views**.
3. Click **Property Filters** → **Custom Property Filter**.
   a. Add a custom property with the following settings:
      * In the **Property Name** field, type generatedBy.
      * In the **Property Value** field, type BusinessStateMachine.
   b. Click **Add**.
   c. Add other custom properties as needed.
4. Click **View Properties** → **List Columns**.
   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, and click **Add**.
      * 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 **View Name** field, and click **Save**.

**Results**

By default, a link to the new view is added to the Process Instances group in the navigation pane. Your users see this view the next time they log in to 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.

**About this task**

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

**Procedure**

1. In the section of the navigation tree where you want to define the new view, click the **New search** icon ( ⊞ ).
2. In the Search and Define Personalized Views page for the view, specify the search criteria.
3. Use the View Properties tab to select the properties to include in the view, and to specify the list properties.

   If this is a task instance view or a process instance view, click **View Settings** to select a set of actions to add to the action bar in the view. If you are a system administrator and system monitor, specify the items that are included in the view.
   - Select the view type:
     – To add administrative actions to the action bar in the view, select **Manage Instances**.
     – To add a set of actions to the action bar that enables the logged-on user to work with the instances, select **Work with Instances**.
   - If you are a system administrator and system monitor, you can limit the search result to your own instances:
     – To show all items that match the search criteria in the view, select **All Instances**. All of the items are shown regardless of whether the system administrator has work items for these items.
     – To show only the items that the logged-on user has work items for, select **Personal Instances**.
4. Enter a display name for the view in the **View Name** field and click **Save**.

**Results**

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) components. A cascading style sheet (CSS) controls how the Web interface is rendered. You can modify the style sheet to adapt the user interface to fit a certain look and feel without writing any new code.

**Before you begin**

Style sheet modification requires profound knowledge about cascading style sheets.

**About this task**

You can change the CSS, for example, so that the default interface conforms to guidelines for corporate identity.

**Procedure**

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.

> **Related concepts**
>
> "Business Process Choreographer Explorer user interface" on page 312
> 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.

## 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:

- "Banner"
- "Footer" on page 322
- "Menu bar" on page 322
- "Login page" on page 322
- "Navigator" on page 322
- "Content panels" on page 323
- "Command bar" on page 323
- "Lists" on page 323
- "Details panel" on page 323
- "Message data" on page 324
- "Tabbed panes" on page 324
- "Search pages" on page 324
- "Error details" on page 324

This file is in the following directory:

`<profile_root>\installedApps\<node_name>\<explorer_instance>\bpcexplorer.war\theme`

### Banner

| Style name | Description |
|---|---|
| .banner | The division for the banner. |
| .banner_left | A division in the banner. It is used to embed the title image of the application. |
| .banner_right | A division in the banner. You can use it, for example, to display further logos. |

## Footer

| Style name | Description |
|---|---|
| .footer | The division for the footer. |
| .footer_left | A division in the footer, for example, you can use it to display the company logo for the application. |
| .footer_right | A division in the footer, for example, you can use it to display further logos. |

## 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. |

## Login page

| Style name | Description |
|---|---|
| .loginPanel | The panel containing the login form. |
| .loginTitle | The title on the form. |
| .loginText | The instructional text. |
| .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. |

## Navigator

| Style name | Description |
|---|---|
| .pageBodyNavigator | The area that contains the navigator. |
| .navigator | JSF subview for navigator which contains the links to the lists. |
| .navigatorTitle | The title for each navigator box. |
| .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. |

| Style name | Description |
| --- | --- |
| .navigatorFrame | The division for each navigator box, for example, to draw a border. |
| .navigatorLink | A link in the navigator box. |
| .expanded | Used when the navigator boxes are expanded. |
| .collapsed | Used when the navigator boxes are collapsed. |

## Content panels

| Style name | Description |
| --- | --- |
| .pageBodyContent | The area that contains the content. |
| .panelContainer | The division panel that contains the list, details or messages. |
| .panelTitle | The title for the displayed content, for example, My To-dos. |
| .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 |
| --- | --- |
| .commandbar | The division container around the command-bar area. |
| .button | The style that is used for buttons in the command bar. |

## Lists

| Style name | Description |
| --- | --- |
| .list | The table that contains the rows. |
| .listHeader | The style used in the header row of the list. |
| .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. |

## Details panel

| Style name | Description |
| --- | --- |
| .details | The division container around a details panel. |
| .detailsProperty | The label for a property name. |
| .detailsValue | The text for a property value. |

## Message data

| Style name | Description |
| --- | --- |
| .messageData | The division container around a message. |
| .messageDataButton | Button style for Add and Remove buttons in the message form. |
| .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. |
| .selectedTab | The active tab header. |
| .tab | The inactive tab headers. |
| .tabPane | The division container that encloses a tabbed pane. |
| .tabbedPaneNested | The division container around nested tabbed panes used on the search pages. |
| .tabHeaderSimple | The tab header of a nested tabbed pane. |
| tabHeaderProcess | The tab header of a nested tabbed pane for process filters. |
| .tabHeaderTask | The tab header of a nested tabbed pane for task filters. |
| .tabPaneSimple | The division container that encloses a nested tabbed pane. |

## Search pages

| Style name | Description |
| --- | --- |
| .searchPane | 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. |
| .summary | The container that encloses the search summary pane. |
| .summaryTitle | The common style for all titles on the search summary pane. |
| .summaryTitleProcess | A style for the title of process related sections on the search summary pane. |
| .summaryTitleTask | A style for the title of task related sections on the search summary pane. |

## Error details

| Style name | Description |
| --- | --- |
| .errorPage | The tabbed pane for an error page. |
| .errorLink | Styles uses to render the button links on a page. |

| Style name | Description |
|---|---|
| .errorDetails | Tabbed pane with error details. |
| .errorDetailsStack | Tabbed pane with an exception stack. |
| .errorDetailsMessage | Text style for error message. |

# Chapter 8. Getting started with Business Process Choreographer Observer

While business processes and tasks are running, WebSphere® Process Server can emit events that contain information about state changes of process instances and their related activities. Use Business Process Choreographer Observer to retrieve statistical information based on these events and create reports on processes and activities.

**About this task**

You can define your own reports, or use a drill-down approach to get more detailed information on specific process instances, activity instances, or inline human tasks. In addition, you can export the reported results for further external processing.

Business Process Choreographer Observer bridges a gap between IT-level monitoring and business level monitoring. By providing means for reporting on events in the Business Flow Manager component it helps you to understand what is happening in Business Process Choreographer.

### Related concepts

"Business Process Choreographer Observer user interface"
Business Process Choreographer Observer is a stand-alone Web application that provides a set of functions for reporting on process and activity events. The interface consists of a taskbar, a navigation pane, and the workspace.

"Business Process Choreographer Observer navigation pane" on page 329
Use the navigation pane to select the kind of report that you want to create, such as process or activity reports. You can also store your own report definitions and add these to the navigation pane.

## Business Process Choreographer Observer user interface

Business Process Choreographer Observer is a stand-alone Web application that provides a set of functions for reporting on process and activity events. The interface consists of a taskbar, a navigation pane, and the workspace.

The following figure shows the layout of the Business Process Choreographer Observer user interface.
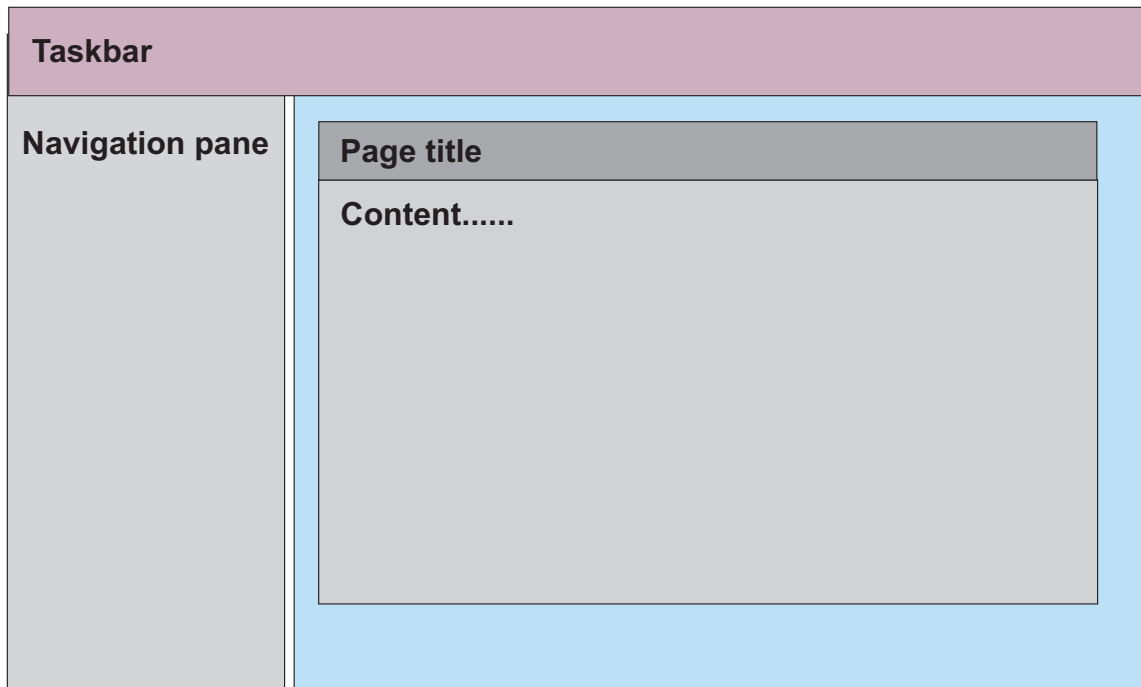
The user interface has the following main areas.

### Taskbar

The taskbar offers options for logging out of Business Process Choreographer Observer, and a link to the general Help page.

### Navigation pane

The navigation pane on the left side of the user interface contains links that you use to select the kind of report that you want to create, for example, you can view the data for an activity instance in a chart.

### Workspace

The workspace on the right side of the user interface contains pages that you use to specify report definitions and view reports. To access these pages click the links in the navigation pane. For information about a page click the **Help** icon  on the respective page.

**Related tasks**

Chapter 8, "Getting started with Business Process Choreographer Observer," on page 327

While business processes and tasks are running, WebSphere® Process Server can emit events that contain information about state changes of process instances and their related activities. Use Business Process Choreographer Observer to retrieve statistical information based on these events and create reports on processes and activities.

# Business Process Choreographer Observer navigation pane

Use the navigation pane to select the kind of report that you want to create, such as process or activity reports. You can also store your own report definitions and add these to the navigation pane.

## Available actions

The following actions are available in the navigation pane:

- 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.
- Navigate to a predefined list or chart.

  Click the kind of instance that you want to report.
- Navigate to the process or activity report wizard.

  Click the **New report** icon ( ) to specify the type of report, the report content and the filter criteria for a report.
- Run a saved process or activity report.

  Click the report name to run the report.
- Open the pop-up menu of a saved process or activity report definition.

  Click the **Show pop-up menu** icon ( ) to work on a saved report definition.

  - To edit the report definition, click the Edit icon ( ).

  - To copy the report definition, click the Copy icon ( ).

  - To delete the report definition, click the Delete icon ( ).

  - To export the report result, click the Export icon ( ).

  - To run a search asynchronously, click the **Asynchronous Search** icon ( ).
    - After the asynchronous search completes successfully, the **Asynchronous Search Completed** icon ( ) is displayed in the navigation pane. Click the name of the report to view your search results.
    - If the asynchronous search does not complete successfully, the **Asynchronous Search Failed** icon ( ) is displayed.

## Predefined lists and charts in the navigation pane

The navigation pane contains the following groups of predefined lists and charts.

**Lists**   This group contains the following lists:

> **Processes**
>
> > Use this list to view processes that emitted a process event during the specified time frame. The processes are listed according to the process state.
>
> **Activities**
>
> > Use this list to view the state that the selected activities reached during the specified time frame. The activities are listed according to the activity state.

**Users**

Use this list to view the activities that the selected users performed during the specified time frame, and the state the activities reached. The activities are displayed according to their state. The corresponding user for each activity is shown.

**Attention:** In countries where gathering data on the work performance of employees violates privacy and data protection laws, the process model must be defined to not emit activity events for individual users.

**Charts** This group contains the following charts:

**Process snapshot**

Use this chart to check how many process instances are in the different states at the specified time. You can view the data in a bar chart, or in a pie chart.

**Processes by period**

Use this chart to check the distribution of the number of process instances that reached the specified state during a specified period. Each instance is shown in the time slice in which it reached the specified state. You can view the data in a line, bar, or pie chart

**Activity snapshot**

Use this chart to check how many activity instances are in the different states at the specified time. You can view the data in a bar chart, or in a pie chart.

**Activities by period**

Use this chart to check the distribution of the number of activity instances that reached the specified state during a specified period. Each instance is shown in the time slice in which it reached the specified state. You can view the data in a line, bar, or pie chart.

## Process and activity reports

The navigation pane links to the following report wizards. The report wizard is indicated by the **New report** icon:  .

**Process reports**

Use process reports to query process instance events. These events describe the state changes of process instances. Use the report wizard to define the data for your reports. You can save and retrieve your report definitions.

**Activity reports**

With an activity report, you query activity instance events. These events describe state changes of activity instances. Use the report wizard to specify individual reports. You can store and retrieve your report definitions.

**Related tasks**

Chapter 8, "Getting started with Business Process Choreographer Observer," on page 327
While business processes and tasks are running, WebSphere® Process Server can emit events that contain information about state changes of process instances

and their related activities. Use Business Process Choreographer Observer to retrieve statistical information based on these events and create reports on processes and activities.

# Chapter 9. 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.

**Related concepts**

Chapter 1, "About business processes," on page 3
A business process is a set of business-related activities that are invoked to achieve a business goal.

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

## 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.

**About this task**

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.

**Related concepts**

Chapter 1, "About business processes," on page 3
A business process is a set of business-related activities that are invoked to achieve a business goal.

"Business process administration—frequently asked questions"
Answers to a set of frequently asked questions about administering business processes.

## 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?" on page 334

- "How do I stop new process instances being created?"
- "What happens to running instances when a newer process template becomes valid?"
- "What happens to a running instance if the template it was created from is stopped?"
- "How can I tell if any process instances are still running?"
- "Why can't I stop a business process application if it has any process instances?"

## 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.

**Related tasks**

# Stopping and starting process templates with the administrative console

You can use the administrative console to start and stop each installed process template individually.

**Before you begin**

If WebSphere administrative security is enabled, verify that your user ID has operator authorization. The server on which the application is installed must be running.

**About this task**

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.

**Procedure**

1. Select the module that you want to manage.

   In the navigation pane of the administrative console, click **Applications** → **SCA modules** → *module_name* .

2. In the Configuration page for the EJB module under **Additional Properties**, click **Business processes**, and then a process template.

3. 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.

4. Start the process template that is in the stopped state.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

# Stopping and starting process templates with administrative commands

Administrative commands provide an alternative to the administrative console for stopping and starting process templates. Use the administrative commands to stop all process templates within an enterprise application.

**Before you begin**

Before you begin this procedure, the following conditions must be met:

- If WebSphere administrative security is enabled, verify that your user ID has operator authorization.
- The application server, on which the process templates are to be stopped or started, must be running. That is, the -conntype none option of wsadmin cannot be used, because a server connection is required.

**About this task**

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 commands to stop and start process templates.

**Procedure**

1. Change to the Business Process Choreographer subdirectory that contains the administration scripts.

   On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

   On UNIX, Linux, and i5/OS systems, enter:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. Stop the process template.

   On Windows systems, enter:

   ```
   install_root\bin\wsadmin -f bpcTemplates.jacl
                               -stop application_name
   ```

   On UNIX, Linux, and i5/OS systems, enter:

   ```
   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.

   On Windows systems, enter:

   ```
   install_root\bin\wsadmin -f bpcTemplates.jacl
                               -start application_name
   ```

   On UNIX, Linux, and i5/OS systems, enter:

   ```
   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.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

## 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.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

## Starting a new process instance

You can start a new process instance from any of the process templates that you are authorized to use.

**About this task**

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.

**Procedure**

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**.

**Results**

The process instance is started. If the business process contains an activity that requires human interaction, a task is generated that can be claimed by any of the potential owners. If you are one of these potential owners, this task appears in the list on the My To-dos page.

If the process instance is a microflow, a process output message is displayed immediately after the process finishes. For long-running processes, ensure that the process instance is not automatically deleted after the process completes. In that case, a process output message is displayed on the process instance view. Not all processes have output messages, for example, if the process implements a one-way operation, an output message is not displayed.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

## 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.

**About this task**

In Business Process Choreographer Explorer, complete the following steps to monitor the progress of a process instance.

**Procedure**

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**.

   The Process State page is displayed. 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.

   The Activity page is displayed, where you can take action so that the process can run to completion.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

## 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.

**Before you begin**

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.

**About this task**

To suspend or resume a process instance, complete the following steps in Business Process Choreographer Explorer.

**Procedure**

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**.

3. Choose one of the options to suspend the process instance.

- To suspend the process until it is manually resumed, select **Suspend**.
- To suspend the process until a certain time, select **Suspend process until**, and specify the date and time.
- To suspend the process for a period of time, select **Suspend process for**, and specify the duration.

4. To confirm your selection, click **Submit**.

   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.

**What to do next**

To resume a process instance that is in the suspended state, select the process instance 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.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

## 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.

**Before you begin**

To terminate a process instance, you must have process administrator authorization.

**About this task**

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.

**Procedure**

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.

**Related concepts**

Chapter 1, "About business processes," on page 3
A business process is a set of business-related activities that are invoked to achieve a business goal.

"Business process administration—frequently asked questions" on page 333
Answers to a set of frequently asked questions about administering business processes.

## 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.

**Before you begin**

To delete a process instance, you must have process administrator authorization. The process instance must be in the finished, failed, terminated, or compensated state.

**About this task**

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.

**Procedure**

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**.

**Results**

This action deletes the selected process instance from the database.

**Related concepts**

Chapter 1, "About business processes," on page 3
A business process is a set of business-related activities that are invoked to achieve a business goal.

"Business process administration—frequently asked questions" on page 333
Answers to a set of frequently asked questions about administering business processes.

# Repairing processes and activities

If the process runs into problems, you can analyze the process and then repair the activities.

**About this task**

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.

**What to do next**

You can now take action to repair the pending activities.

> **Related concepts**
>
> Chapter 1, "About business processes," on page 3
> A business process is a set of business-related activities that are invoked to achieve a business goal.
>
> "Business process administration—frequently asked questions" on page 333
> Answers to a set of frequently asked questions about administering business processes.

## Restarting activities

If you have repaired an activity, you can restart it using new input data.

**Before you begin**

The activity must be in the stopped state and the associated process instance must be in the running state.

**About this task**

To restart an activity, complete the following steps in Business Process Choreographer Explorer.

**Procedure**

1. Navigate to the Activity page for the activity and click **Restart**.

   For example, on the Process Instances Administered By Me page, click the name of a process instance. On the Process Instance page, click the **Activities** tab, and click the name of the activity you want to 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. If an expiration time is set for the activity, specify the expiration behavior for the restarted activity.
4. Click **Restart**.

**Related concepts**

Chapter 1, "About business processes," on page 3
A business process is a set of business-related activities that are invoked to achieve a business goal.

"Business process administration—frequently asked questions" on page 333
Answers to a set of frequently asked questions about administering business processes.

## 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.

**Before you begin**

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.

**About this task**

To force the completion of an activity, complete the following steps in Business Process Choreographer Explorer.

**Procedure**

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.

**Related concepts**

Chapter 1, "About business processes," on page 3
A business process is a set of business-related activities that are invoked to achieve a business goal.

"Business process administration—frequently asked questions" on page 333
Answers to a set of frequently asked questions about administering business processes.

## 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.

**Before you begin**

For microflows to be compensated, the compensation service must be started in the administrative console.

**About this task**

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.

**Procedure**

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.

   **Related concepts**

   Chapter 1, "About business processes," on page 3
   A business process is a set of business-related activities that are invoked to achieve a business goal.

   "Business process administration—frequently asked questions" on page 333
   Answers to a set of frequently asked questions about administering business processes.

## 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.

**Related concepts**

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

**Related tasks**

Chapter 7, "Getting started with Business Process Choreographer Explorer," on page 311
Depending on your user role, you can use Business Process Choreographer Explorer to manage business processes and human tasks, or work with your assigned tasks.

### Stopping and starting task templates with the administrative console

Use the administrative console to start and stop each installed task template individually.

**Before you begin**

If WebSphere administrative security is enabled, verify that user ID has operator authorization.

**About this task**

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.

**Procedure**

1. Select the module that you want to manage.

   In the navigation pane of the administrative console, click **Applications** → **SCA modules** → *module_name* .

2. In the Configuration page for the EJB module under **Additional Properties**, click **Human tasks**, and then a process template.

3. To stop the task template, click **Stop**.

4. To start the task template, click **Start**.

   **Related concepts**

   Chapter 2, "About human tasks," on page 37
   A human task is a component that allows people and services to interact.

# Stopping and starting task templates with the administrative commands

Administrative commands provide an alternative to the administrative console for stopping and starting task templates. Use the administrative commands to stop all task templates within an enterprise application.

**Before you begin**

If WebSphere administrative security is enabled, verify that you are logged with a user ID that has operator authorization.

**About this task**

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.

**Procedure**

1. Change to the Business Process Choreographer subdirectory that contains the administration scripts.

   On Windows systems, enter:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

   On UNIX, Linux, and i5/OS systems, enter:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. Stop the task template.

   On Windows systems, enter:

   ```
   install_root\bin\wsadmin -f bpcTemplates.jacl
                            -stop application_name
   ```

   On UNIX, Linux, and i5/OS systems, enter:

   ```
   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.

   On Windows systems, enter:

   ```
   install_root\bin\wsadmin -f bpcTemplates.jacl
                            -start application_name
   ```

   On UNIX, Linux, and i5/OS systems, enter:

   ```
   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.

   **Related concepts**

   Chapter 2, "About human tasks," on page 37
   A human task is a component that allows people and services to interact.

## 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.

**About this task**

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.

**Procedure**

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**.

**Results**

The task instance is ready to be worked on.

   **Related concepts**

   Chapter 2, "About human tasks," on page 37
   A human task is a component that allows people and services to interact.

   **Related tasks**

   Chapter 7, "Getting started with Business Process Choreographer Explorer," on page 311
   Depending on your user role, you can use Business Process Choreographer Explorer to manage business processes and human tasks, or work with your assigned tasks.

## 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.

**About this task**

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.

**Procedure**

1. Display the tasks that have been assigned to you.

   Click **Task Instances → My To-dos**.

   This action displays the My To-dos 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.

**Results**

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.

**Related concepts**

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

**Related tasks**

Chapter 7, "Getting started with Business Process Choreographer Explorer," on page 311
Depending on your user role, you can use Business Process Choreographer Explorer to manage business processes and human tasks, or work with your assigned tasks.

# Suspending and resuming task instances

You can suspend task instances. You might want to do this, for example, to fix a problem that is causing the task instance to fail. When the prerequisites for the task are met, you can resume running the task instance.

**Before you begin**

To suspend and resume a task instances, you must have task administrator authorization.

To suspend a task instance, the task instance must be in either the running or failing state. To resume a task, the task instance must be in the suspended state.

Suspending tasks is only supported for human tasks that use the WebSphere Application Server simple calendar.

**About this task**

To suspend a task instance, complete the following steps in Business Process Choreographer Explorer.

**Procedure**

1. Display the task instances that you can administer.

   Click **Administered By Me** under Task Instances in the navigation pane.
2. On the Task Instance page, click **Suspend**.
3. Choose one of the options to suspend the task instance.
   - To suspend the task until it is manually resumed, select **Suspend**.
   - To suspend the task until a certain time, select **Suspend task until**, and specify the date and time.
   - To suspend the task for a period of time, select **Suspend task for**, and specify the duration.
4. To confirm your selection, click **Submit**. The task instance is put into the suspended state.

**What to do next**

To resume a task instance that is in the suspended state, click **Resume**.

   **Related concepts**

   Chapter 2, "About human tasks," on page 37
   A human task is a component that allows people and services to interact.

## Managing priorities of human tasks

You can use the priorities of human tasks to filter for tasks, and to sort your list of tasks.

**About this task**

To change the priority of a task instance, complete the following steps in Business Process Choreographer Explorer.

**Procedure**

1. Display a list of task instances.

   For example, click **My To-dos** under Task Instances in the navigation pane.
2. Select the check box next to the task instance, and click **Change Priority**.
3. Enter a value, and click **Submit**.

   The priority of the task instance is set to the new value.

**What to do next**

To sort the list of tasks by priority, click the arrows in the table header.

   **Related concepts**

   Chapter 2, "About human tasks," on page 37
   A human task is a component that allows people and services to interact.

# 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.

**About this task**

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, or specify absence settings for the time you are away. You might also need to create additional work items or delete work items that are not needed anymore.

**Related concepts**

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

**Related tasks**

Chapter 7, "Getting started with Business Process Choreographer Explorer," on page 311
Depending on your user role, you can use Business Process Choreographer Explorer to manage business processes and human tasks, or work with your assigned tasks.

## 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.

**About this task**

In Business Process Choreographer Explorer, complete the following steps to transfer a task that you own.

**Procedure**

1. Display the tasks that you own.

   Click **My To-dos** in the Task Instances group of 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.

**Results**

The transferred task appears on the list of tasks belonging to the new task owner.

**Related concepts**

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

**Related tasks**

"Specifying absence settings" on page 351
If you intend to be away from the office for a certain time, specify a substitute for your tasks.

"Specifying absence settings for users" on page 352
If users are prevented from working on their tasks, for example, if they are on sick leave, specify a substitute for the user's tasks and a date range for which the delegation is applicable.

## 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.

**Before you begin**

To transfer a work item you must have one of the following roles, and according to the assignment reason, the task must be in one of the following states.

| Role | Assignment reason | Task state | Work items can be transferred to the following user roles: |
|---|---|---|---|
| Owner | Owner | Claimed | Potential owner, administrator. |
| Starter | Starter | Expired, terminated, finished, failed, or running | Potential starter, administrator. |
| Originator | Originator | Any task state | Potential instance creator, administrator. |
| Originator | Potential starter | Inactive | Any user role. |
| Administrator | Starter | Expired, terminated, finished, failed, or running | Starter. |
| Administrator | Potential starter | Inactive | Potential starter. |
| Administrator | Reader or administrator | In any but in the inactive state | Reader, administrator. |
| Administrator | Potential owner or editor | Ready, or claimed | Potential owner, or editor. |

**About this task**

In Business Process Choreographer Explorer, complete the following steps to transfer a work item.

**Procedure**

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**.

**Results**

The transferred work item with the new work-item owner appears in the list of work items.

> **Related concepts**
>
> Chapter 2, "About human tasks," on page 37
> A human task is a component that allows people and services to interact.
>
> **Related tasks**
>
> "Specifying absence settings for users" on page 352
> If users are prevented from working on their tasks, for example, if they are on sick leave, specify a substitute for the user's tasks and a date range for which the delegation is applicable.

## Specifying absence settings

If you intend to be away from the office for a certain time, specify a substitute for your tasks.

**Before you begin**

To perform this task the Virtual Member Manager (VMM) people directory provider for substitution is required.

**About this task**

Depending on the applied substitution policy, one or more than one substitute can receive your work assignments while you are absent. The substitution policy can differ for each task templates.

**Procedure**
1. In the taskbar, click **My Substitutes**.
2. On the My Substitutes page, specify the absence settings, and click **Save**.

   a. To enable your absence settings, select the **I am absent** check box.

   b. In the **My substitutes** field, enter the user ID of your substitute, and click **Add**.

   c. Optional: Add further substitutes as needed. Depending on the applied substitution policy, one or more than one substitute can receive your work assignments while you are absent. The substitution policy can differ for each task templates.

   d. Optional: To remove a substitute from the list, select the user ID of the substitute and click **Remove**. To select more than one substitute, hold down the Ctrl key.
3. Ask your BPESystemAdministrator to refresh the people query results.

**Results**

While the **I am absent** check box is selected, your substitutes will receive your work assignments.

**What to do next**

Work assignments that were assigned to you before the **I am absent** check box got selected must be transferred separately.

> **Related concepts**
>
> "Substitution for absentees" on page 83
> The substitution feature allows you to specify absence settings either for yourself, or for members of the group that you administer. A substitution policy defines how to deal with tasks and escalations that are assigned to absent users.
>
> Chapter 2, "About human tasks," on page 37
> A human task is a component that allows people and services to interact.
>
> **Related tasks**
>
> "Transferring tasks that you own" on page 349
> 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.
>
> "Configuring the Virtual Member Manager people directory provider" on page 193
> Configure the Virtual Member Manager (VMM) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task. The default people directory provider is ready to use, and only needs to be configured if you introduce custom people assignment criteria.
>
> "Refreshing people query results, using the administrative console" on page 287
> The results of a people query are static. Use the administrative console to refresh people queries.
>
> "Refreshing people query results, using administrative commands" on page 306
> The results of a people query are static. Use the administrative commands to refresh people queries.

## Specifying absence settings for users

If users are prevented from working on their tasks, for example, if they are on sick leave, specify a substitute for the user's tasks and a date range for which the delegation is applicable.

**Before you begin**

You must have BPESystemAdministrator rights to perform this task. To perform this task, the Virtual Member Manager (VMM) people directory provider for substitution is required.

**Procedure**

1. In the taskbar, click **Define Substitutes**.
2. On the Define Substitutes page, specify the absence settings, and click **Save**.
   a. Enter the user ID of the user for whom you want to specify the absence settings.
   b. To enable the absence settings, select the **User is absent** check box.

c. In the **The user's substitute** field, enter the user ID of the substitute that you want to appoint, and click **Add**.

d. Optional: Add further substitutes as needed. Depending on the applied substitution policy, one or more than one substitute can receive the work assignments while the user is absent. The substitution policy can differ for each task templates.

e. Optional: To remove a substitute from the list, select the user ID of the substitute and click **Remove**. To select more than one substitute, hold down the Ctrl key.

3. Refresh the people query results.

**Results**

While the **User is absent** check box is selected, the substitutes will receive the user's work assignments.

**What to do next**

Work assignments that were assigned to the absent user before the **User is absent** check box got selected must be transferred separately.

**Related concepts**

"Substitution for absentees" on page 83
The substitution feature allows you to specify absence settings either for yourself, or for members of the group that you administer. A substitution policy defines how to deal with tasks and escalations that are assigned to absent users.

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

**Related tasks**

"Transferring work items for which you are the starter, originator, or administrator of the task" on page 350
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.

"Transferring tasks that you own" on page 349
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.

"Configuring the Virtual Member Manager people directory provider" on page 193
Configure the Virtual Member Manager (VMM) people directory provider for Business Process Choreographer to perform people assignment, which determines who can start a process or claim an activity or a task. The default people directory provider is ready to use, and only needs to be configured if you introduce custom people assignment criteria.

"Refreshing people query results, using the administrative console" on page 287
The results of a people query are static. Use the administrative console to refresh people queries.

"Refreshing people query results, using administrative commands" on page 306
The results of a people query are static. Use the administrative commands to refresh people queries.

## 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 people directory 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.

**Before you begin**

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.

**About this task**

In Business Process Choreographer Explorer, complete the following steps to create a work item.

**Procedure**

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**.

**Results**

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.

> **Related concepts**
> Chapter 2, "About human tasks," on page 37
> A human task is a component that allows people and services to interact.

## 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.

**Before you begin**

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 work item if the task instance 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 work item if the task instance is in the terminated or expired state.

**About this task**

In Business Process Choreographer Explorer, complete the following steps to delete a work item.

**Procedure**

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**.

**Results**

The work items are deleted.

> **Related concepts**
>
> Chapter 2, "About human tasks," on page 37
> A human task is a component that allows people and services to interact.

# Viewing task escalations

An escalation notifies the escalation receiver that a user might have problems completing their assigned task on time.

**About this task**

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.

**Procedure**

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.

> **Related concepts**
>
> Chapter 2, "About human tasks," on page 37
> A human task is a component that allows people and services to interact.

## 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.

**Before you begin**

The following rules apply to escalation e-mails:

- Your people directory provider must support the specification of e-mail addresses, such as Lightweight Directory Access Protocol (LDAP) or virtual member manager.
- The **Everybody**, **Nobody**, **Group** and **Users by user ID** people assignment criteria are not supported. For example, use **User records by user ID** instead.

**Procedure**

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 **People directory (JNDI name)** field.

      Set the value of this field to one of the following:
      - `bpe/staff/samplevmmconfiguration`
      - `bpe/staff/samplevmmconfiguration`
      - The people directory configuration name (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.

      To insert a variable to include task specific information into the text, click **Add Variable** and select an appropriate variable from the list. In the editor, the variable will appear between "%" characters, but will be replaced when it is evaluated during execution in the runtime environment when the email is sent.

      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. If authentication is enabled, set the User ID and password for the SMTP host.

      In the administrative console, click **Resources** ⇝ **Mail** ⇝ **Mail Sessions** ⇝ *HTMMailSession_nodeName_serverName* to check this setting, or **Resources** ⇝ **Mail** ⇝ **Mail Sessions** ⇝ *HTMMailSession_clusterName* if Business Process Choreographer is configured on a cluster. The SMTP host is defined on the cell level.
   b. Ensure that the sender e-mail address (**Sender e-mail address**) that you specify when you configure the human task manager is a valid e-mail address.

      In the administrative console, click **Servers** ⇝ **Application servers** ⇝ *server_name* to check this setting, or **Servers** ⇝ **Clusters** ⇝ *cluster_name* if Business Process Choreographer is configured on a cluster. On the **Configuration** tab, in the Business Integration section, click **Business Process Choreographer** ⇝ **Human Task Manager**.

**What to do next**

If a problem occurs with escalation e-mails, check the `SystemOut.log` file for error messages.

**Related concepts**

Chapter 2, "About human tasks," on page 37
A human task is a component that allows people and services to interact.

# Creating and editing custom properties in Business Process Choreographer Explorer

Create new custom properties to specify additional properties for process instances, activity instances, or task instances.

**About this task**

To create custom properties for an instance, complete the following steps in Business Process Choreographer Explorer.

**Procedure**

1. Display a list of process instances, activity instances, or task instances, and click the name of an instance to open the details page.

   For example, to open a list of task instances, click **My To-dos** under Task Instances in the navigation pane.

2. On the Custom Properties tab, click **Add**.

3. Enter a name for the custom property in the **Property Name** field, and a value in the **Property Value** field.

4. Optional: To add additional custom properties, go to step 2.

5. Optional: To remove a new custom property, click the **Delete** icon next to the custom property.

6. Optional: To change the property name or value for a custom property, click the custom property and enter the new value.

7. Click **Save**. After you save a custom property, you cannot change the property name, and you cannot delete the custom property.

# Reporting on business processes and activities

During the processing of business processes and activities, an event can be 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.

**About this task**

You can work with predefined reports or create user-defined reports for processes and activities.

**Related concepts**

"Snapshot reports" on page 358
Use snapshot reports to determine the states of activities or processes at a specific date and time.

"Period reports" on page 359
Use period reports to determine how often specific activity or process events occur over a period of time.

"Time processing" on page 361
In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

# Snapshot reports

Use snapshot reports to determine the states of activities or processes at a specific date and time.

For example, you want to know the number of process instances that are running at midnight. For each process or activity instance, Business Process Choreographer Observer finds the last event before the specified date and time, and evaluates the resulting state. The following state diagram shows how events qualify for a snapshot report.
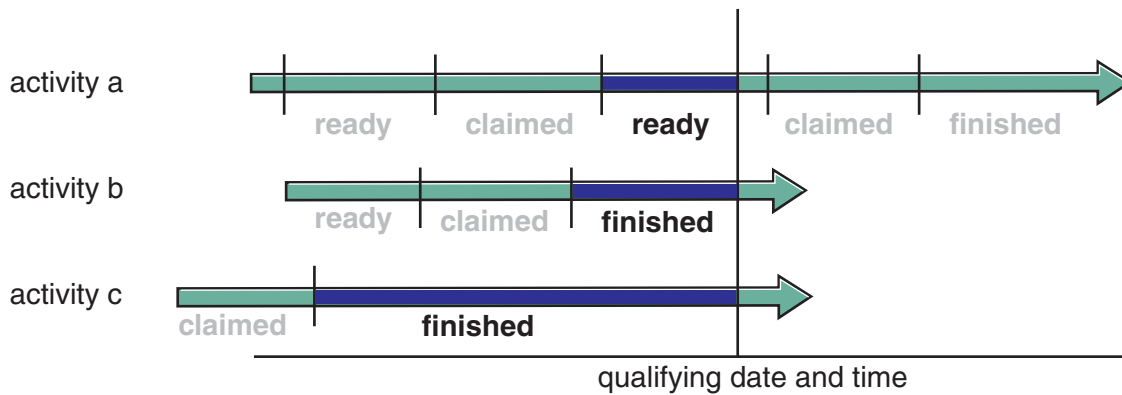


The snapshot includes one activity in the ready state (activity a) and two activities in the finished state (activities b and c).

## Configuration parameter ReportAtSnapshotRange

If the Observer database contains process instance data that covers a long period of time, getting a snapshot can be time consuming. To avoid querying events that are not relevant anymore, use the ReportAtSnapshotRange configuration parameter. Only the events that are newer than the specified date and time minus the value of the ReportAtSnapshotRange configuration parameter are considered in the report. The following state diagram shows how events qualify for a snapshot report when the ReportAtSnapshotRange parameter is set.
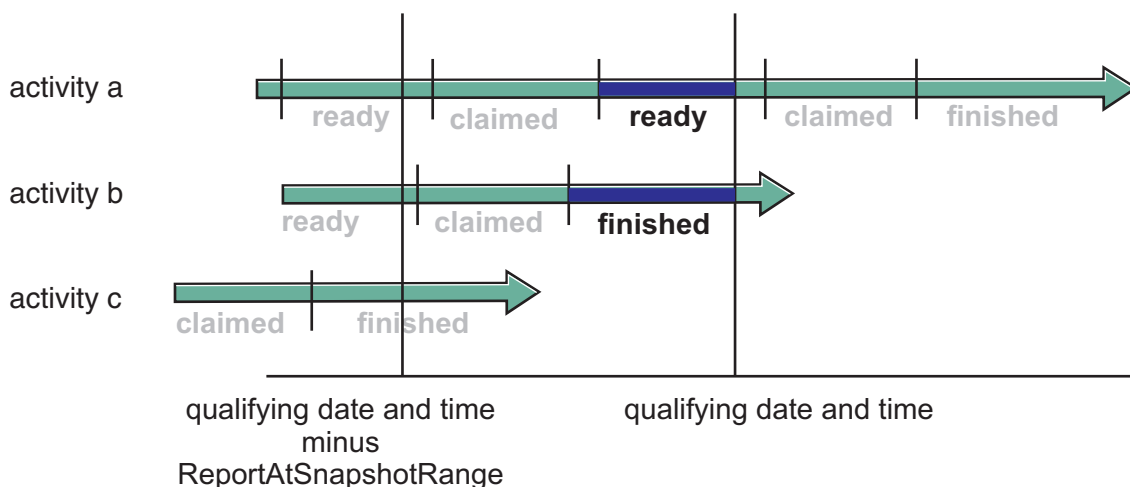


The snapshot includes one activity in the ready state (activity a) and one activity in the finished state (activity b). The report does not return the status of activity c.

## Reporting cycles

You can define reporting cycles for snapshot reports. Use this option to create a report that contains repeating snapshots for multiple dates. For example, you want to report the number of started processes for each day of March. You do not need to report each day separately. Instead, you can define a start date of 1 March, the number of snapshots after the start date as 31, and the time between snapshots as 1 day. The resulting report contains an additional column that includes the time slice number. The value of each time slice indicates the day of the month.

**Related tasks**

"Creating a predefined snapshot chart" on page 364
Use predefined snapshot charts to see the distribution of process instance or activity instance states for a specified date and time.

"Creating user-defined snapshot reports" on page 367
You can define user-defined reports that take a snapshot of the state information at a specified date and time. You can also create reports that contain state snapshots for regular points within a reporting period, for example, the first of each month at midnight.

"Changing configuration parameters for the Business Process Choreographer Observer" on page 256
Tuning the configuration parameters for the Business Process Choreographer Observer and event collector applications is important to enable verification and improve performance.

"Reporting on business processes and activities" on page 357
During the processing of business processes and activities, an event can be 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.

"Using the predefined lists and charts" on page 361
Predefined lists and charts provide a drill-down approach to get you to state and event information for runtime entities. With each step in the drill-down process, you define further the kind of information you are interested in. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

"Creating user-defined reports" on page 366
User-defined process and activity reports are more flexible than the predefined lists and charts. In addition, you can store and reuse your report definitions, and you can export the report results.

# Period reports

Use period reports to determine how often specific activity or process events occur over a period of time.

With a period view, you specify the start and end date for the reporting period. The report covers the interval between these two dates. For example, you want to know how many staff activities have been claimed during the day.

The following state diagram shows how events qualify to a period report. A report that covers the period shown in the following example includes six activity events; four events for activity a and 2 events for activity b. Activity c completed before the start of the reporting period and therefore it does not contribute events to the report.
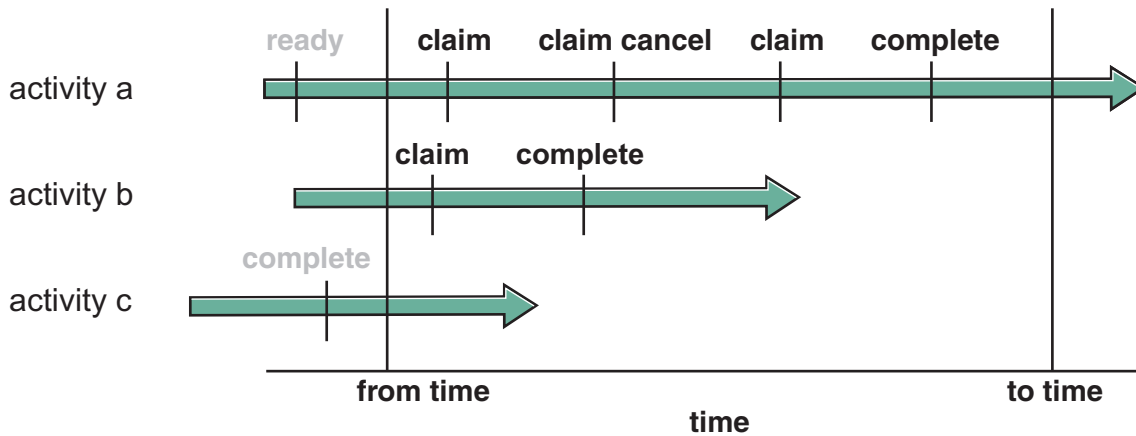
This means that if you query the number of completed events in this period, the result is 2.

## Reporting cycles

You can define reporting cycles for period reports. Use this option to create a report that covers multiple periods. For example, you want to report the number of started processes for each month in the last 12 months. You do not need to report each month separately. Instead, you can define a start date of 1 January, the number of time slices after the start date as 12, and the length of a time slice as 1 month. The resulting report contains an additional column that includes the time slice number. The value of each time slice indicates the month.

**Related tasks**

"Creating a predefined period chart" on page 364
Use the predefined period charts to see the distribution of the number of process instances or activity instances that reached a specified state during a time period. Each instance is shown in the time slice in which it reached the specified state.

"Creating user-defined period reports" on page 369
You can create user-defined reports for process or activity events that occur over a period of time. You can also create reports that cover multiple periods according to a reporting cycle.

"Reporting on business processes and activities" on page 357
During the processing of business processes and activities, an event can be 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.

"Using the predefined lists and charts" on page 361
Predefined lists and charts provide a drill-down approach to get you to state and event information for runtime entities. With each step in the drill-down process, you define further the kind of information you are interested in. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

"Creating user-defined reports" on page 366
User-defined process and activity reports are more flexible than the predefined lists and charts. In addition, you can store and reuse your report definitions, and you can export the report results.

# Time processing

In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

## Timestamps

In the database, timestamps are stored in coordinated universal time (UTC). Timestamps that are entered and displayed are always in the local time of the location where the user interface runs. This means that if you specify a snapshot report with a reporting cycle and the reporting cycle spans a clock adjustment for daylight saving time, the dates and times vary by one hour after the clock change.

For example, if you specify a snapshot report with a reporting cycle that takes the first snapshot at 8:00 a.m. during winter time and the following snapshots are taken every 24 hours, then the snapshots are taken at 9:00 a.m. during daylight saving time.

## Durations of months and years

If you specify a report with a reporting cycle, and, for example, you give the time slice length in units of months or years, the lengths of each individual time slice varies depending on the calendar. This allows you to specify a report where each time slice represents a month of a year.

**Related tasks**

"Reporting on business processes and activities" on page 357
During the processing of business processes and activities, an event can be 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.

"Using the predefined lists and charts"
Predefined lists and charts provide a drill-down approach to get you to state and event information for runtime entities. With each step in the drill-down process, you define further the kind of information you are interested in. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

"Creating user-defined reports" on page 366
User-defined process and activity reports are more flexible than the predefined lists and charts. In addition, you can store and reuse your report definitions, and you can export the report results.

# Using the predefined lists and charts

Predefined lists and charts provide a drill-down approach to get you to state and event information for runtime entities. With each step in the drill-down process, you define further the kind of information you are interested in. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

**About this task**

The following types of predefined lists and charts are available:
- Lists
- Processes and activity snapshot charts

- Process and activity instances by period charts

  **Related concepts**

  "Snapshot reports" on page 358
  Use snapshot reports to determine the states of activities or processes at a specific date and time.

  "Period reports" on page 359
  Use period reports to determine how often specific activity or process events occur over a period of time.

  "Time processing" on page 361
  In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

  **Related tasks**

  "Example: Using the predefined charts" on page 365
  This scenario gives you an example of how you could use the predefined charts.

  "Example: Using the predefined lists" on page 363
  This scenario gives you an example of how you could use the predefined lists.

## Creating a report using the predefined lists

Use the predefined lists to report on the number of process or activity events, that occurred within a specified time period, sorted by states. You can also use the lists to drill down to the events for a particular instance. In addition, you can export the report results for each state.

**Procedure**

1. Select a list type.

   Predefined lists are available for process instances, activity instances, and activities associated with users.

2. Enter the start and end date for the time period in which you are interested, and click **Continue**.

   Depending on the list type, a list of process templates, activity templates, or a list of users and the number of their associated instances is displayed.

3. Select the check boxes of the instances that you are interested in, and click **Instances Snapshot**.

   The events for the selected instances are displayed in a tabbed pane. Each of the pages shows the instances in a particular state.

4. Optional: To see all of the events for, and for more information about a specific instance, click the instance name.

5. Optional: To export the reported data in CSV format, click **Export**. Select whether you want to open or to save the generated export data, and click **OK**. The reported data for the currently displayed state is exported.

   **Related concepts**

   "Time processing" on page 361
   In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

   **Related tasks**

   "Example: Using the predefined lists" on page 363
   This scenario gives you an example of how you could use the predefined lists.

## Example: Using the predefined lists

This scenario gives you an example of how you could use the predefined lists.

**Before you begin**

Your factory produces different items Item1, Item2, and Item3. Your manufacturing and shipment process is modeled and run as a SOA process with WebSphere Process Server. Each customer order is represented by a dedicated process instance of the appropriate process template. After an item has been shipped to the customer, your shipment process reaches the end state finished. If a customer cancels an order, the corresponding process instance is terminated and reaches the terminated state.

To see how many customers have cancelled their order of Item1, Item2, or Item3 within the last month, you are interested in the number of process instances that reached the terminated state. In addition, you want to know how far the order had been processed when the cancellation occurred.

**About this task**

Use the predefined lists to create a view that shows you how many processes have been cancelled, and to see the state the process was in when the cancellation occurred:

**Procedure**

1. In the **Lists** section in the navigation bar, select **Processes**.
2. On the Search Criteria page, enter the start and end date for the time period in which you are interested, and click **Continue**. The Process Templates page lists all of the process templates that generated a process within the observation period. For each process template, you can see the number of process instances that were started and ended.
3. On the Process Template page, select all templates of the list, and click **Instance snapshot**. The Process Instance page lists all of the process instances grouped by the state that they reached within the observation period.
4. On the Process Instance page, select the **Terminated** tab to see the total number of cancellations during the observation period.
5. Sort the list by template name, and evaluate the number of cancellations per process template.
6. For further details, click the name of a terminated process instance to view the Process Instance Detail page. Check the work time and the elapsed time of the instance.

   **Related tasks**

   "Using the predefined lists and charts" on page 361
   Predefined lists and charts provide a drill-down approach to get you to state and event information for runtime entities. With each step in the drill-down process, you define further the kind of information you are interested in. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

   "Creating a report using the predefined lists" on page 362
   Use the predefined lists to report on the number of process or activity events, that occurred within a specified time period, sorted by states. You can also use the lists to drill down to the events for a particular instance. In addition, you can export the report results for each state.

## Creating a predefined snapshot chart

Use predefined snapshot charts to see the distribution of process instance or activity instance states for a specified date and time.

**Before you begin**

Your browser must be enabled to run Macromedia Flash Player to view charts.

**Procedure**

1. Select the type of snapshot.

   Predefined snapshot charts are available for process instances and activity instances.

2. Enter the search criteria and click **Continue**.

   A list of object templates is displayed that meet the search criteria.

3. Select the check boxes of the templates that you are interested in, and click **Continue with selected**.

   You can change the chart type to show the results as a bar chart or a pie chart.

   **Related concepts**

   "Snapshot reports" on page 358
   Use snapshot reports to determine the states of activities or processes at a specific date and time.

   "Time processing" on page 361
   In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

   **Related tasks**

   "Example: Using the predefined charts" on page 365
   This scenario gives you an example of how you could use the predefined charts.

## Creating a predefined period chart

Use the predefined period charts to see the distribution of the number of process instances or activity instances that reached a specified state during a time period. Each instance is shown in the time slice in which it reached the specified state.

**Before you begin**

Your browser must be enabled to run Macromedia Flash Player to view charts.

**About this task**

For example, you can use the predefined charts to see the distribution of finished process instances over the last 12 months.

**Procedure**

1. Select the type of period chart.

   Predefined period charts are available for process instances and activity instances.

2. Enter the search criteria and click **Continue**.

   Enter the start date for the time period, and specify the number of time slices, the length of each time slice, and the state you are reporting on. For example,

to report on the finished instances for each month over the last 12 months, specify 12 as the number of time slices, and 1 month as the length of each time slice.

A list of object templates are displayed that meet the search criteria.

3. Select the check boxes of the templates that you are interested in, and click **Continue with selected**.

You can change the chart type to show the results as a bar chart, line chart, or a pie chart.

**Related concepts**

"Period reports" on page 359
Use period reports to determine how often specific activity or process events occur over a period of time.

"Time processing" on page 361
In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

**Related tasks**

"Example: Using the predefined charts"
This scenario gives you an example of how you could use the predefined charts.

## Example: Using the predefined charts

This scenario gives you an example of how you could use the predefined charts.

**About this task**

Your factory produces different items Item1 and Item2. Your manufacturing and shipment process is modeled and run as a SOA process with WebSphere Process Server. Each customer order is represented by a dedicated process instance of the appropriate process template.

Recently you have expanded your production line by Item3. You have a new Item3 ordering template and you want to know the progress that your production line has made during the last month. As an indicator you want to see the number of production orders within the last 30 days.

To visualize the number of production orders that were processed within the last 30 days, specify a chart view that shows all process instances that are related to the OrderItem3 process template for the period of interest:

**Procedure**

1. In the **Charts** section in the navigation bar, select Process by period to see the statistical distribution of process instances within the last thirty days.
2. Specify the search criteria:
   a. Enter the start date of your observation period
   b. Set the number of time slices to 30.
   c. Set the length of a time slice to one day.
   d. In the **Focus on state** list, select **Running**, and click **Continue**.

   The Select Process Templates page opens, which contains a list of all process templates that are related to a process instances that occurred within the observation period.
3. Select the OrderItem3 template to see all process instances that are related to this process template, and click **Continue with selected**.

4. The Process Instances Snapshot page displays all process instances that are in the different states at the specified time.
5. Use the line chart or bar chart to visualize the progress that your process made within the last month

**What to do next**

Your report shows all process instances that reached the state running within the observation period.

**Related tasks**

"Using the predefined lists and charts" on page 361
Predefined lists and charts provide a drill-down approach to get you to state and event information for runtime entities. With each step in the drill-down process, you define further the kind of information you are interested in. For example, you can specify dates and other filter criteria to view the data for an activity instance in a bar chart.

"Creating a predefined snapshot chart" on page 364
Use predefined snapshot charts to see the distribution of process instance or activity instance states for a specified date and time.

"Creating a predefined period chart" on page 364
Use the predefined period charts to see the distribution of the number of process instances or activity instances that reached a specified state during a time period. Each instance is shown in the time slice in which it reached the specified state.

# Creating user-defined reports

User-defined process and activity reports are more flexible than the predefined lists and charts. In addition, you can store and reuse your report definitions, and you can export the report results.

**About this task**

For process reports, you can get information about the attributes of process instances, and the activities that belong to the process instances. For activity reports, you can get information about the attributes of the activities, and the process instances that the activties are associated with. You can define one-off reports, or save your report definitions so that you can run it when required. Include parameters to change the values of your report definition every time you run the report.

**Related concepts**

"Snapshot reports" on page 358
Use snapshot reports to determine the states of activities or processes at a specific date and time.

"Period reports" on page 359
Use period reports to determine how often specific activity or process events occur over a period of time.

"Time processing" on page 361
In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

**Related tasks**

"Example: Using the user-defined reports" on page 372
This scenario gives you an example of how you could use the user-defined reports.

**Related reference**

"Report attributes" on page 374
Use attributes to define the content of your report, and to filter the results. The attributes that are available depend on the report type.

"Business process events for Business Process Choreographer Observer" on page 375
Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer. A subset of these events is available for Business Process Choreographer Observer.

"Performance-relevant attributes" on page 377
The time that is necessary to run a report definition can vary. To improve the performance of the report generation, you can optimize your report definition. Some general rules can help you to evaluate the influence that the report attributes have on the performance.

## Creating user-defined snapshot reports

You can define user-defined reports that take a snapshot of the state information at a specified date and time. You can also create reports that contain state snapshots for regular points within a reporting period, for example, the first of each month at midnight.

**About this task**

The report wizard guides you through the definition of the report.

**Procedure**

1. In the navigation pane, click the **New Report** icon ( ) either for process reports or for activity reports.
2. On the Select Report Type page, click **Snapshot Report**, and click **Next**.
3. On the Select Snapshot Type page, specify when you want the snapshot to be taken and click **Next**.
   - To see the current status, click **Take a snapshot now**. The snapshot date and time is evaluated every time you run the report.

     The Specify Content page is displayed. Continue with step 5.
   - To see the status of processes or activities on a particular date and time, for example, 10 June at 8:00 a.m., click **Take a snapshot at a specific date and time**.

     The Specify Snapshot Settings page is displayed. Continue with step 4.
   - To see the status at regular points within a reporting period, click **Take repeated snapshots according to a reporting cycle**.

     The Specify Snapshot Settings page is displayed. Continue with step 4.
4. Specify the snapshot settings, and click **Next**.

   If the snapshot is to be taken at a specific date and time, specify the date and time settings. You can specify a date and time that is in the future. To change the settings each time you run the report, select the **Use these settings as a parameter** check box.

   For reports with a reporting cycle:

   a. Select whether you want to set the start date, or the end date of the reporting cycle, and click **Next**.

b. To set the start date of the reporting cycle, specify when the first snapshot is to be taken. To set the end date of the reporting cycle, specify when the last snapshot is to be taken.

c. To define the duration of the reporting cycle, set the number of snapshots and the time between each snapshot.

d. To change the settings for the reporting cycle each time you run the report, select the **Use these settings as a parameter** check box.

5. On the Specify Report Content page, specify the information that you want the report to contain, and click **Next**.

   For reports with a reporting cycle, the list of attributes already contains the snapshot number attribute. You cannot delete this attribute.

   a. Click **Add** to see a list of attributes that you can include in the report; these attributes become the column headings of your report. The position of the attributes determines the order of the columns in the report. For each attribute you can also specify how the results are sorted within the column. If you specify a sort order for more than one attribute, the results are sorted in the order of the attributes. Consider to rearrange the order of the attributes to change the sort order of the results in the report.

      - To modify an attribute, click the **Edit** icon ( ).

      - To delete an attribute, click the **Delete** icon ( ).

      - To change the position of an attribute in the report, click the **Up** icon ( ) or the **Down** icon ( ).

   b. To limit the number of entries in the result, for example, for performance reasons, enter a value in the **Threshold** field to specify the maximum number of results.

      The default threshold value is 20. If you do not want to limit the result, set the value to -1.

      To change the threshold value each time you run the report, select the **Use the threshold as a parameter** check box.

6. Optional: On the Specify Filter Content page, set the filter criteria for the attributes.

   Use filter criteria to restrict the values that the attributes can take thus making your report more specific. The report includes only those processes and activities that fulfill all of the specified filter criteria. If you specified an attribute on the Specify Report Content page that is an aggregate, the list of filter criteria already contains filter criteria for this attribute. You cannot delete this filter.

   a. Click **Add** to see a list of attributes for which you can specify filter criteria.

      - For more complex value types, such as timestamps, click the **Input Helper** icon ( ) to complete the field.

      - To change the value for a filter criterion each time you run the report, select the **Parameter** check box.

   b. Click **Next**.

   The Summary page is displayed. This page shows the report definition.

7. On the Summary page, do one of the following:

   - If your report definition does not contain parameters, click **Run**.

      The resulting report is displayed.

- If your report definition contains parameters, click **Next**.

  You can change the values of the parameter, then click **Run**. The resulting report is displayed.

  If the report results are not as you expected, you can click **Edit** to change the settings for the report.

8. Optional: Export the report result.

   To export the reported data in CSV format, click **Export**. Select whether you want to open the generated export data, or to save the data on your hard disk, and click **OK**.

   The **Export** button is only displayed if the report list contains items.

9. Optional: Save the report definition.

   If this is a report that you want to run more than once, for example, a monthly report that shows the completed process instances on the 10th of every month, click **Save** and enter a report name. The report appears in the navigation pane.

   **Related concepts**

   "Snapshot reports" on page 358
   Use snapshot reports to determine the states of activities or processes at a specific date and time.

   "Time processing" on page 361
   In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

   **Related tasks**

   "Example: Using the user-defined reports" on page 372
   This scenario gives you an example of how you could use the user-defined reports.

   **Related reference**

   "Report attributes" on page 374
   Use attributes to define the content of your report, and to filter the results. The attributes that are available depend on the report type.

   "Business process events for Business Process Choreographer Observer" on page 375
   Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer. A subset of these events is available for Business Process Choreographer Observer.

   "Performance-relevant attributes" on page 377
   The time that is necessary to run a report definition can vary. To improve the performance of the report generation, you can optimize your report definition. Some general rules can help you to evaluate the influence that the report attributes have on the performance.

## Creating user-defined period reports

You can create user-defined reports for process or activity events that occur over a period of time. You can also create reports that cover multiple periods according to a reporting cycle.

**About this task**

The report wizard guides you through the definition of the report.

**Procedure**

1. In the navigation pane, click the **New Report** icon (  ) either for process reports or for activity reports.
2. On the Select Report Type page, click **Period Report**, and click **Next**.
3. On the Select Period Type page, specify the type of period, and click **Next**.

   For example, for processes, you can select one of the following period types:
   - To see the events from a specified date to the present, click **Report on all processes up to now**.
   - To see the events for a specified period, click **Report on processes in a specified period**.
   - To see the events for regular intervals in a reporting period, click **Report on processes according to a reporting cycle**.

   The Specify Date and Time page is displayed.
4. Specify the date and time settings, and click **Next**.

   For reports on all processes up to now, specify the start date. The end date is generated every time you run the report. For reports on processes in a specified period, specify the start and the end date. You can specify dates that are in the future. To change the settings each time you run the report, select the **Use these settings as a parameter** check box.

   For reports with a reporting cycle:
   a. Select whether you want to set the start date, or the end date of the reporting cycle, and click **Next**.
   b. To set the start date of the reporting cycle, specify the start date of the first time slice. To set the end date of the reporting cycle, specify the end date of the last time slice.
   c. To define the duration of the reporting cycle, set the total number of time slices, and the length of each time slice.
   d. To change the settings for the reporting cycle each time you run the report, select the **Use these settings as a parameter** check box.
5. On the Specify Report Content page, specify the information that you want the report to contain, and click **Next**.

   For reports with a reporting cycle, the list of attributes already contains the time slice number attribute. You cannot delete this attribute.
   a. Click **Add** to see a list of attributes that you can include in the report; these attributes become the column headings of your report. The position of the attributes determines the order of the columns in the report. For each attribute you can also specify how the results are sorted within the column. If you specify a sort order for more than one attribute, the results are sorted in the order of the attributes. Consider to rearrange the order of the attributes to change the sort order of the results in the report.

      - To modify an attribute, click the **Edit** icon (  ).
      - To delete an attribute, click the **Delete** icon (  ).
      - To change the position of an attribute in the report, click the **Up** icon (  ) or the **Down** icon (  ).
   b. To limit the number of entries in the result, for example, for performance reasons, enter a value in the **Threshold** field to specify the maximum number of results.

The default threshold value is 20. If you do not want to limit the result, set the value to -1.

To change the threshold value each time you run the report, select the **Use the threshold as a parameter** check box.

6. Optional: On the Specify Filter Content page, set the filter criteria for the attributes.

   Use filter criteria to restrict the values that the attributes can take thus making your report more specific. If you specified an attribute on the Specify Report Content page that is an aggregate, the list of filter criteria already contains filter criteria for this attribute. You cannot delete this filter.

   a. Click **Add** to see a list of attributes for which you can specify filter criteria.

      - For more complex value types, such as timestamps, click the **Input Helper** icon ( 💡 ) to complete the field.

      - To change the value for a filter criterion each time you run the report, select the **Parameter** check box.

   b. Click **Next**.

   The Summary page is displayed. This page shows the report definition.

7. On the Summary page, do one of the following:

   - If your report definition does not contain parameters, click **Run**.

     The resulting report is displayed.

   - If your report definition contains parameters, click **Next**.

     You can change the values of the parameter, then click **Run**. The resulting report is displayed.

   If the report results are not as you expected, you can click **Edit** to change the settings for the report.

8. Optional: Export the report result.

   To export the reported data in CSV format, click **Export**. Select whether you want to open the generated export data, or to save the data on your hard disk, and click **OK**.

   The **Export** button is only displayed if the report list contains items.

9. Optional: Save the report definition.

   If this is a report that you want to run regularly, for example, for monthly reporting, click **Save** and enter a report name. The report appears in the navigation pane.

   **Related concepts**

   "Period reports" on page 359
   Use period reports to determine how often specific activity or process events occur over a period of time.

   "Time processing" on page 361
   In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

   **Related tasks**

   "Example: Using the user-defined reports" on page 372
   This scenario gives you an example of how you could use the user-defined reports.

   **Related reference**

"Report attributes" on page 374
Use attributes to define the content of your report, and to filter the results. The attributes that are available depend on the report type.

"Business process events for Business Process Choreographer Observer" on page 375
Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer. A subset of these events is available for Business Process Choreographer Observer.

"Performance-relevant attributes" on page 377
The time that is necessary to run a report definition can vary. To improve the performance of the report generation, you can optimize your report definition. Some general rules can help you to evaluate the influence that the report attributes have on the performance.

## Example: Using the user-defined reports

This scenario gives you an example of how you could use the user-defined reports.

**Before you begin**

Your factory produces different items Item1, Item2, and Item3. Your manufacturing and shipment process is modeled and run as a SOA process with WebSphere Process Server. Each customer order is represented by a dedicated process instance of the appropriate process template. After an item has been shipped to the customer, your shipment process reaches the end state, finished. If a customer cancels an order, the corresponding process instance is terminated and reaches the terminated state.

One of the customers who cancelled his order complains about the long response time he experienced. You want to know why this order took so long to process.

**About this task**

Create a user-defined report for process instances that are in the terminated state and that have a work time of more than two days. In addition, your report should reveal what went wrong with the terminated process instances.

**Procedure**

1. Retrieve the process instance data that belong to the customer's order.

   The customer name, address, and the order number are part of the business data and therefore are contained in the process message. However, Business Process Choreographer Observer cannot make use of the content of a business object because it is not part of a Common Event Infrastructure (CEI) event. However, you know that you are looking for a process instance that is in the terminated state and that has a work time of more than two days.

   a. In the **Process Reports** section in the navigation bar, select **Create a new report**.

   b. Because you are focused on the state of a process instance, select the report type **Snapshot Report**.

   c. On the Select Snapshot Type page, select **Take a snapshot at a specific date and time**. Specify the date and time immediately after the order cancellation as the qualifying snapshot date.

   d. On the Report Content page, add **Process instance ID**, **Process work time**, **Process started** and **Process completed** to your report content.

e. On the Filter Content page, specify **Process work time greater 2 days** and **Process state equal Terminated** as filter content, and run the report.

f. On the Report Result page, check the process instance ID, start date, and completion date to find the process instance that corresponds to your customer's order. If the report result does not meet your expectations, for example, if the list of process instances is too long, click **Edit** to modify your search criteria.

g. Copy the process instance ID to the clipboard because you will need the ID in step 2.

2. Get the information that reveals what went wrong with a specific process instance.

a. In the **Process Reports** section in the navigation bar, select **Create a new report**.

b. Select the report type **Snapshot Report**.

Do not use Period Report type. You are interested in attributes that are related to a snapshot report. To see the difference, define and run a period report with exactly the same attributes.

c. On the Select Snapshot Type page, select **Take a snapshot at a specific date and time**. Specify the date and time immediately after the order cancellation as the qualifying snapshot date.

d. On the Report Content page, add **Process instance ID**, **Activity name**, **Activity started** and **Activity completed** to your report content.

e. On the Filter Content page, specify **Process instance ID equal** *your_customer's_process_instance_ID* as filter content, and run the report. The report reveals in which activity most time has been spent.

f. Optional: If you need further information to evaluate what exactly was the root cause for the delay, edit and rerun your report.

g. Save your report definition.

3. Finally you want to avoid such a situation in the future. You want to have a report at the end of each working day that lists all of the active order processes that are in danger of exceeding the time limit because of resource constraints or failures.

a. Edit your saved report definition. On the Select Snapshot Type page, change the snapshot type to **Take a snapshot now**, delete the filter content **Process instance ID equal** *your_customer's_process_instance_ID* and add the expression **Process work time greater 1 day**.

b. Run your modified report and check that there are no process instances that meet the new filter criteria.

c. Save the report so that you can run it at the end of every working day.

**Related tasks**

"Creating user-defined reports" on page 366
User-defined process and activity reports are more flexible than the predefined lists and charts. In addition, you can store and reuse your report definitions, and you can export the report results.

"Creating user-defined snapshot reports" on page 367
You can define user-defined reports that take a snapshot of the state information at a specified date and time. You can also create reports that contain state snapshots for regular points within a reporting period, for example, the first of each month at midnight.

You can create user-defined reports for process or activity events that occur over a period of time. You can also create reports that cover multiple periods according to a reporting cycle.

## Report attributes

Use attributes to define the content of your report, and to filter the results. The attributes that are available depend on the report type.

Each attribute defined as report content is the name of a column in the report. In addition, use attributes to filter the results of your query. You can also define filter criteria for attributes that you have not included in your report.

| Attribute | Description | Snapshot reports | Period reports |
|---|---|---|---|
| Activity completed | The time when the activity instance reached one of the following end states: failed, finished, skipped, terminated, or expired. | X | X |
| Activity event | The event code of the activity event. | X | X |
| Activity event count | The number of activity events emitted by the activity instance. | X | X |
| Activity instance ID | The activity instance ID. | X | X |
| Activity kind | The kind of the activity instance. | X | X |
| Activity last user name | The name of the last user who initiated an action with this activity. | X | X |
| Activity name | The name of the activity instance. | X | X |
| Activity started | The time when the activity instance was started. | X | X |
| Activity state | The state the activity instance is in after the event. | X | X |
| Activity template ID | The activity template ID. | X | X |
| Average duration of activities | The average duration of all of the activity instances in seconds. | X | X |
| Average duration of processes | The average duration of all of the process instances in seconds. | X | X |
| Event time | The time when the event occurred. | X | X |
| Exception text | If an exception triggered the activity event, the exception message can be part of the event data and is then stored in this field. | X | X |
| Number of activities in state | The number of activity instances that are in the specified state. | X | |
| Number of activity events | The number of activity events that occurred during the specified period. | | X |
| Number of process events | The number of process events that occurred during the specified period. | | X |
| Number of processes in state | The number of process instances that are in the specified state. | X | |
| Process activity count | The number of activities of a process instance that emitted at least one event. | X | X |
| Process activity event count | The number of activity events that belong to a process instance. | X | X |

| Attribute | Description | Snapshot reports | Period reports |
|---|---|---|---|
| Process completed | The time when the process instance reached one of the following end states: compensated, compensation failed, failed, finished, or terminated. | X | X |
| Process deletion time | The time when the process was deleted from the Business Process Choreographer database. | X | X |
| Process event | The event code of the process instance event. | X | X |
| Process event count | The number of process events emitted by the process instance. | X | X |
| Process instance ID | The process instance ID. | X | X |
| Process last user name | The name of the last user who initiated an action with this process. | X | X |
| Process started | The time when the process instance was started. | X | X |
| Process state | The state that the process instance is in after the event. | X | X |
| Process template ID | The process template ID. | X | X |
| Process template name | The process template that is associated with the process instance. | X | X |
| Process work time | The duration of the process instance. This value is the sum of the work times of all of the basic activities that are contained in the process. Basic activities are activities that have no structure and do not contain other activities. | X | X |
| Snapshot number | In a snapshot report with a reporting cycle, this attribute identifies a specific snapshot in the reporting cycle. | X | |
| Time slice number | In a period report with a reporting cycle, this attribute identifies a specific time slice in the reporting cycle. | | X |
| User name | The user ID of a user who is associated to the event. | X | X |
| Valid from | The time when the process template became valid. | X | X |

## Business process events for Business Process Choreographer Observer

Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer. A subset of these events is available for Business Process Choreographer Observer.

The following types of events can be caused by business process:

- "Process events" on page 376
- "Activity events" on page 376

Depending on the settings in WebSphere Integration Developer, 6.0.2-events and 6.1-events can occur.

Business Process Choreographer Observer does not require business data in events.

## Process events

The following table describes all process events that you can report on using Business Process Choreographer Observer.

| Code | Description |
| --- | --- |
| 21000 | Process started |
| 21001 | Process suspended |
| 21002 | Process resumed |
| 21004 | Process completed |
| 21005 | Process terminated |
| 21019 | Process restarted |
| 42001 | Process failed |
| 42003 | Process compensating |
| 42004 | Process compensated |
| 42046 | Process compensation failed |
| 42009 | Process terminating |
| 42010 | Process failing |

## Activity events

The following table describes all activity events that you can report on using Business Process Choreographer Observer.

| Code | Description |
| --- | --- |
| 21006 | Activity ready |
| 21007 | Activity started |
| 21011 | Activity completed |
| 21021 | Claim canceled |
| 21022 | Activity claimed |
| 21027 | Activity terminated |
| 21080 | Activity failed |
| 21081 | Activity expired |
| 42005 | Activity skipped |
| 42015 | Activity stopped |
| 42031 | Activity force retried |
| 42032 | Activity force completed |
| 42036 | Activity has message received |

**Related reference**

"Business process events" on page 572
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.

## Performance-relevant attributes

The time that is necessary to run a report definition can vary. To improve the performance of the report generation, you can optimize your report definition. Some general rules can help you to evaluate the influence that the report attributes have on the performance.

**Specify filters**

Use appropriate filters to restrict the amount of retrieved data. Consider limiting the report results by date, or other activity or process instances properties. For snapshot reports, set the ReportAtSnapshotRange configuration parameter to an appropriate value.

**Period reports versus snapshot reports**

Snapshot reports tend to decrease the performance more than period reports.

**Reports with a reporting cycle**

Reports that are defined with a reporting cycle tend to decrease the performance, in particular if many periods or snapshots are defined for the query.

**Aggregates**

Aggregates such as the total number of events, or the average durations of instances can necessitate the processing of large amounts of data, and therefore decrease the performance.

**Number of results shown**

If you are interested only in some of the results of a report, specify a threshold to limit the number of entries in the result. This reduces the amount of data transferred between the database and the user interface.

However, if you define a sort order, before the data can be sorted, all of the resulting data must be collected in the database. In this case, reducing the number of results shown does not improve the performance. Instead, you should set up appropriate filter expressions.

**Event and instance information**

In the Observer database, information related to events is stored in the event database table, whereas information related to activity and process instances is stored in the instance database table. If you create a report that contains both instance-related and event-specific information, the tables are joined to get the required information. If you create a report that contains only one type of information, the tables are not joined. Therefore reports that contain only one type of information usually have a better performance than a report that queries both instance-related and event-specific information.

**Related reference**

"Changing configuration parameters for the Business Process Choreographer Observer" on page 256
Tuning the configuration parameters for the Business Process Choreographer Observer and event collector applications is important to enable verification and improve performance.

# Using saved user-defined report definitions

If you saved your report definitions, you can run your reports when required, edit your report definitions, or use a copy of your report definition to create similar reports. In addition, you can run your reports asynchronously, and export the report results.

## Running saved user-defined report definitions

You can run your saved report definitions when required. If your report contains parameters, you can set the values that you are interested in each time you run the report.

**Procedure**

1. To run a saved report definition, click the name of the report in the navigation pane.
   - If your report definition does not contain parameters, the resulting report is displayed.
   - If your report definition contains parameters, the Run Report page is displayed. You can change the values of the parameter, then click **Run**.
     
     The resulting report is displayed.
2. Optional: Export the report result.
   
   To export the reported data in CSV format, click **Export**. Select whether you want to open the generated export data, or to save the data on your hard disk, and click **OK**.

## Running saved user-defined report definitions asynchronously

You can run the saved report asynchronously to continue working while the query is running.

**About this task**

To run a saved report definition asynchronously, click the **Show pop-up menu** icon (), and click the Asynchronous Search icon ().

**Procedure**

If your report definition contains parameters, the Run Report page is displayed. You can change the values of the parameter, then click Run.

- After the asynchronous search completes successfully, the Asynchronous Search Completed icon () is displayed in the navigation pane. Click the name of the report to view your search results.
- If the asynchronous search does not complete successfully, the Asynchronous Search Failed icon () is displayed.

## Exporting report results using the pop-up menu

For saved user-defined reports, you can export the report results for further external processing without running the report.

**About this task**

This option is only available for saved user-defined report definitions that do not contain parameters.

**Procedure**

1. To export the report results of a saved report definition, click the **Show pop-up menu** icon (), and click the Export icon ().
2. Select whether you want to open or to save the generated export data, and click **OK**. The reported data is exported.

## Exporting report results using the export client

For saved user-defined reports, you can use the export client command line tool to run reports and export the report results for further external processing.

**Before you begin**

This option is only available for saved user-defined report definitions that do not contain parameters.

The export client tool *wps_install_root*/ProcessChoreographer/client/exportclient.jar has to be installed on your local workstation.

**Procedure**

To run a report and export the report result, use the command line to start the export client.

On Windows platforms, enter: java -jar exportclient.jar*options*

On Linux, UNIX, and i5/OS platforms, enter: java -jar exportclient.jar*options*

You can specify options directly on the command line in the format -option value, and specify the name of the properties file. The options have the format option=value. Options that are specified on the command line take precedence over those that are specified in a properties file.

Following options are valid:

*Table 15. Valid options for the export client*

| Option | Description |
|--------|-------------|
| help | Shows usage information. |
| verbose | Shows additional information when the result is exported that you can use for debugging. |
| unicode | Exports the result in UTF-8 encoding. The default is the local operating system encoding. |
| o | Overwrites any existing file. The default is an error if the file already exists. |
| properties | This defines a fully qualified file name that contains additional options. |
| url | Complete URL where the Business Process Choreographer Observer is running. The default is *http://localhost:9080* |
| out | This defines a fully qualified filename to store the export result. The default is *report name.csv*. |
| userid | When security is enabled, a valid user ID is required. |
| password | When security is enabled, a valid password is required. |
| reportname | The name of a saved report definition is required. |

## Editing and copying saved user-defined report definitions

You can change the settings of your saved report definitions, or use a copy of your report definition to create similar reports.

**Procedure**

1. Click the **Show pop-up menu** icon (📝 ), and do one of the following:

   - To edit the report definition, click the **Edit** icon (✏️ ) .

   - To copy the report definition, click the **Copy** icon (📋 ).

   The Summary page opens. This page shows the time settings, the report content, and the filter settings of the report.

   Click the links below each summary section to change the corresponding settings. You cannot change the report type.

2. Optional: To edit the time settings, click **Modify the date and reporting cycle settings of your report**.

   According to the type of report you defined, either the Select Snapshot Type page, or the Select Period Type page opens.

3. Optional: To modify the report content, click **Modify the result content**.

   The Specify Report Content page opens.

   For reports with a reporting cycle, the list of attributes contains either the snapshot number attribute, or the time slice number attribute, depending on the type of report you defined. You cannot delete this attribute.

4. Optional: To modify the filter settings, click **Modify the filter settings**.

   The Specify Filter Content page opens.

5. On the Summary page, do one of the following:
   - If your report definition does not contain parameters, click **Run**.

     The resulting report is displayed.

   - If your report definition contains parameters, click **Next**.

     You can change the values of the parameter, then click **Run**. The resulting report is displayed.

   If the report results are not as you expected, you can click **Edit** to change the settings for the report.

6. On the Report Result page, click **Save**. If you are going to create a copy of a report definition, enter a name for the new report, and click **Save** again.

   The new report appears in the navigation pane.

   **Related concepts**

   "Time processing" on page 361
   In your report, consider the way that Business Process Choreographer Observer processes timestamps and durations.

   **Related reference**

   "Report attributes" on page 374
   Use attributes to define the content of your report, and to filter the results. The attributes that are available depend on the report type.

   "Business process events for Business Process Choreographer Observer" on page 375
   Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer. A subset of these events is available for Business Process Choreographer Observer.

The time that is necessary to run a report definition can vary. To improve the performance of the report generation, you can optimize your report definition. Some general rules can help you to evaluate the influence that the report attributes have on the performance.

## Deleting saved user-defined report definitions

To keep the navigation pane clear and manageable, delete outdated and redundant report definitions.

**Before you begin**

You cannot restore deleted report definitions.

**Procedure**

To delete a report definition, click the **Show pop-up menu** icon (), and click the **Delete** icon ().

**Results**

The report name disappears from the navigation pane.

# Part 4. Developing and deploying modules

# Chapter 10. 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.

**About this task**

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.

### Related concepts

"Invocation scenarios for business processes" on page 16
A business process is an SCA (Service Component Architecture) component implementation type. It can expose services to other partners and consume services provided by other partners. A business process can be a service provider that is made available by the Business Process Choreographer APIs, an SCA service provider for other SCA service components, or an SCA client that invokes other SCA service components, including other business processes.

# Chapter 11. 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.

**About this task**

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.

**Procedure**

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.

**About this task**

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 interface of the session bean

An EJB client application accesses the remote interface of the session bean through the remote home interface of the bean.

**About this task**

The session bean can be either the BusinessFlowManager session bean for process applications or the HumanTaskManager session bean for task applications.

**Procedure**

1. Add a reference to the remote interface of the 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. 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.

d. 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.

3. Locate the remote home interface of the session bean through the 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 remote 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 lock conflicts, avoid running statements similar to the following in parallel:

```
// Obtain user transaction interface
UserTransaction transaction=
        (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//read the activity instance
process.getActivityInstance(aiid);
//claim the activity instance
process.claim(aiid);

transaction.commit();
```

The getActivityInstance method and other read operations set a read lock. In this example, a read lock on the activity instance is upgraded to an update lock on the activity instance. This can result in a database deadlock when these transactions are run in parallel.

## 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 interface of the session bean

An EJB client application accesses the local interface of the session bean through the local home interface of the bean.

**About this task**

The session bean can be either the BusinessFlowManager session bean for process applications or the HumanTaskManager session bean for human task applications.

**Procedure**

1. Add a reference to the local interface of the 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. Locate the local home interface of the session bean through the 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 BusinessFlowManager bean

     LocalBusinessFlowManagerHome processHome =
         (LocalBusinessFlowManagerHome)initialContext.lookup
         ("java:comp/env/ejb/LocalBusinessFlowManagerHome");
   ```

   The local home interface of the 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 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:

```
// Obtain user transaction interface
UserTransaction transaction=
        (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//read the activity instance
process.getActivityInstance(aiid);
//claim the activity instance
process.claim(aiid);

transaction.commit();
```

The getActivityInstance method and other read operations set a read lock. In this example, a read lock on the activity instance is upgraded to an update lock on the activity instance. This can result in a database deadlock when these transactions are run in parallel

## 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 HumanTaskManager bean
LocalHumanTaskManagerHome taskHome =
        (LocalHumanTaskManagerHome)initialContext.lookup
        ("java:comp/env/ejb/LocalHumanTaskManagerHome");

...
//Access the local interface of the session bean
LocalHumanTaskManager task = taskHome.create();

...
//Call the business functions exposed by the service interface
task.callTask(tkiid,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.

**About this task**

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 can be called by all users, and it returns the properties of the objects for which work items exist. The queryAll method can be called only by users who have one of the following J2EE roles: BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor, or TaskSystemMonitor. This method returns the properties of all the objects that are stored in the database.

All API queries are mapped to SQL queries. The form of the resulting SQL query depends on the following aspects:

- Whether the query was invoked by someone with one of the J2EE roles.
- The objects that are queried. Predefined database views are provided for you to query the object properties.
- The insertion of a from clause, join conditions, and user-specific conditions for access control.

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.

## Syntax of the API query method

The syntax of the Business Process Choreographer API queries is similar to SQL queries. A query can include a select clause, a where clause, an order-by clause, a skip-tuples parameter, a threshold parameter and a 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 16.*

| Object | Syntax |
|---|---|
| Process template | `ProcessTemplateData[] queryProcessTemplates`<br>                      `(java.lang.String whereClause,`<br>                       `java.lang.String orderByClause,`<br>                       `java.lang.Integer threshold,`<br>                       `java.util.TimeZone timezone);` |
| Task template | `TaskTemplate[]        queryTaskTemplates`<br>                      `(java.lang.String whereClause,`<br>                       `java.lang.String orderByClause,`<br>                       `java.lang.Integer threshold,`<br>                       `java.util.TimeZone timezone);` |
| Business-process and task-related data | `QueryResultSet query (java.lang.String selectClause,`<br>                       `java.lang.String whereClause,`<br>                       `java.lang.String orderByClause,`<br>                       `java.lang.Integer skipTuples`<br>                       `java.lang.Integer threshold,`<br>                       `java.util.TimeZone timezone);` |

**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 similar to the syntax of an SQL SELECT clause; use commas to separate parts of the clause. Each part of the clause must specify a column from one of the predefined views. The columns must be fully specified by view name and column name. The columns returned in the QueryResultSet object appear in the same order as the columns 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 similar to the syntax of an SQL WHERE clause. You do not need to explicitly add an SQL from clause or join predicates to the API 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

  The LIKE operation supports the wildcard characters that are defined for the queried database.
- Set operation: IN

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.

You can specify a list of columns from the views by which the result is sorted. These columns must be fully qualified by the name of the view and the column. It is a best practice to specify columns that are in the select clause.

The order-by clause syntax is similar to the syntax of an SQL order-by clause; use commas to separate each part of the clause. You can also specify `ASC` to sort the columns in ascending order, and `DESC` to sort the columns in descending order. If you do not want to sort the query result set, you must specify `null` for the order-by clause.

Sort criteria are applied on the server, that is, the locale of the server is used for sorting. If you specify more than one column, the query result set is ordered by the values of the first column, then by the values of the second column, and so on. You cannot specify the columns in the order-by clause by position as you can with an SQL query.

**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 in the query result set.

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')",
                (String)null,
                (Integer)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')",
                (String)null, (Integer)null, (TimeZone)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 properties of the objects that satisfy the where clause given by the caller, and that the caller is authorized to see. You can read elements in a relative fashion using the API next method or in an absolute fashion using the first and last methods. Because the implicit cursor of a query result set is initially positioned before the first element, you must call either the first or next methods before reading an element. You can use the size method to determine the number of elements in the set.

An element of the query result set comprises the selected attributes of work items and their associated referenced objects, such as activity instances and process instances. The first attribute (column) of a QueryResultSet element specifies the value of the first attribute specified in the select clause of the query request. The second attribute (column) of a QueryResultSet element specifies the value of the second attribute specified in the select clause of the query request, and so on.

You can retrieve the values of the attributes by calling a method that is compatible with the attribute type and by specifying the appropriate column index. The numbering of the column indexes starts with 1.

| Attribute type | Method |
|---|---|
| String | getString |
| OID | getOID |
| Timestamp | getTimestamp<br>getString<br>getTimestampAsLong |
| Integer | getInteger<br>getShort<br>getLong<br>getString<br>getBoolean |
| Boolean | getBoolean<br>getShort<br>getInteger<br>getLong<br>getString |
| byte[] | getBinary |

**Example:**

The following query is run:
```
QueryResultSet resultSet = process.query("ACTIVITY.STARTED,
                                          ACTIVITY.TEMPLATE_NAME AS NAME,
                                          WORK_ITEM.WIID, WORK_ITEM.REASON",
                                          (String)null, (String)null,
                                          (Integer)null, (TimeZone)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"
```

## User-specific access conditions

User-specific access conditions are added when the SQL SELECT statement is
generated from the API query. These conditions guarantee that only those objects
are returned to the caller that satisfy the condition specified by the caller and to
which the caller is authorized.

The access condition that is added depends on whether the user is a system
administrator.

### Queries invoked by users who are not system administrators

The generated SQL WHERE clause combines the API where clause with an access
control condition that is specific to the user. The query retrieves only those objects
that the user is authorized to access, that is, only those objects for which the user
has a work item. A work item represents the assignment of a user or user group to
an authorization role of a business object, such as a task or process. If, for example,
the user, John Smith, is a member of the potential owners role of a given task, a
work item object exists that represents this relationship.

For example, if a user, who is not a system administrator, queries tasks, the
following access condition is added to the WHERE clause if group work items are
not enabled:

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'user'
     OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

So, if John Smith wants to get a list of tasks for which he is the potential owner,
the API where clause might look as follows:

```
"WORK_ITEM.REASON == WORK_ITEM.REASON.REASON_POTENTIAL_OWNER"
```

This API where clause results in the following access condition in the SQL
statement:

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'JohnSmith'
     OR WI.OWNER_ID = null AND WI.EVERYBODY = true)
AND WI.REASON = 1
```

This also means that if John Smith wants to see the activities and tasks for which
he is a process reader or a process administrator and for which he does not have a

work item, then a property from the PROCESS_INSTANCE view must be added to the select, where, or order-by clause of the query, for example, PROCESS_INSTANCE.PIID.

If group work items are enabled, an additional access condition is added to the WHERE clause that allows a user to access objects that the group has access to.

### Queries invoked by system administrators

System administrators can invoke the query method to retrieve objects that have associated work items. In this case, a join with the WORK_ITEM view is added to the generated SQL query, but no access control condition for the WORK_ITEM.OWNER_ID.

In this case, the SQL query for tasks contains the following:
```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
```

### queryAll queries

This type of query can be invoked only by system administrators or system monitors. Neither conditions for access control nor a join to the WORK_ITEM view are added. This type of query returns all of the data for all of the objects.

## Examples of the query and queryAll methods

These examples show the syntax of various typical API queries and the associated SQL statements that are generated when the query is processed.

**Example: querying tasks in the ready state:**

This example shows how to use the query method to retrieve tasks that the logged-on user can work with.

John Smith wants to get a list of the tasks that have been assigned to him. For a user to be able to work on a task, the task must be in the ready state. The logged-on user must also have a potential owner work item for the task. The following code snippet shows the query method call for this query:
```
query( "DISTINCT TASK.TKIID",
        "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
          AND " +
        "TASK.STATE = TASK.STATE.STATE_READY AND " +
        "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
         (String)null, (String)null, (Integer)null, (TimeZone)null )
```

The following actions are taken when the SQL SELECT statement is generated:
- A condition for access control is added to the where clause. This example assumes that group work items are not enabled.
- Constants, such as `TASK.STATE.STATE_READY`, are replaced by their numeric values.
- A FROM clause and join conditions are added.

The following code snippet shows the SQL statement that is generated from the API query:
```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  WI.OBJECT_ID = TA.TKIID
```

```
AND    TA.KIND IN ( 101, 105 )
AND    TA.STATE = 2
AND    WI.REASON = 1
AND    ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

To restrict the API query to tasks for a specific process, for example, sampleProcess, the query looks as follows:

```
query( "DISTINCT TASK.TKIID",
          "PROCESS_TEMPLATE.NAME = 'sampleProcess' AND "+
          "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
            AND " +
          "TASK.STATE = TASK.STATE.STATE_READY AND " +
          "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

**Example: querying tasks in the claimed state:**

This example shows how to use the query method to retrieve tasks that the logged-on user has claimed.

The user, John Smith, wants to search for tasks that he has claimed and are still in the claimed state. The condition that specifies "claimed by John Smith" is TASK.OWNER = 'JohnSmith'. The following code snippet shows the query method call for the query:

```
query( "DISTINCT TASK.TKIID",
          "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
          "TASK.OWNER = 'JohnSmith'",
          (String)null, (String)null, (Integer)null, (TimeZone)null )
```

The following code snippet shows the SQL statement that is generated from the API query:

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  WI.OBJECT_ID = TA.TKIID
   AND    TA.STATE = 8
   TA.OWNER = 'JohnSmith'
   AND    ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

When a task is claimed, work items are created for the owner of the task. So, an alternative way of forming the query for John Smith's claimed tasks is to add the following condition to the query instead of using TASK.OWNER = 'JohnSmith':

```
WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER
```

The query then looks like the following code snippet:

```
query( "DISTINCT TASK.TKIID",
          "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
          "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER",
          (String)null, (String)null, (Integer)null, (TimeZone)null )
```

The following actions are taken when the SQL SELECT statement is generated:

- A condition for access control is added to the where clause. This example assumes that group work items are not enabled.
- Constants, such as TASK.STATE.STATE_READY, are replaced by their numeric values.
- A FROM clause and join conditions are added.

The following code snippet shows the SQL statement that is generated from the API query:

```
SELECT DISTINCT TASK.TKIID
   FROM    TASK TA, WORK_ITEM WI,
   WHERE   WI.OBJECT_ID = TA.TKIID
   AND     TA.STATE = 8
   AND     WI.REASON = 4
   AND   ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

John is about to go on vacation so his team lead, Anne Grant, wants to check on his current work load. Anne has system administrator rights. The query she invokes is the same as the one John invoked. However, the SQL statement that is generated is different because Anne is an administrator. The following code snippet shows the generated SQL statement:

```
SELECT DISTINCT TASK.TKIID
   FROM    TASK TA, WORK_ITEM WI,
   WHERE   TA.TKIID = WI.OBJECT_ID =
   AND     TA.STATE = 8
   AND    TA.OWNER = 'JohnSmith')
```

Because Anne is an administrator, an access control condition is not added to the WHERE clause.

**Example: querying escalations:**

This example shows how to use the query method to retrieve escalations for the logged-on user.

When a task is escalated, and escalation receiver work item is created. The user, Mary Jones wants to see a list of tasks that have been escalated to her. The following code snippet shows the query method call for the query:

```
query( "DISTINCT ESCALATION.ESIID, ESCALATION.TKIID",
        "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_ESCALATION_RECEIVER",
          (String)null, (String)null, (Integer)null, (TimeZone)null )
```

The following actions are taken when the SQL SELECT statement is generated:
- A condition for access control is added to the where clause. This example assumes that group work items are not enabled.
- Constants, such as TASK.STATE.STATE_READY, are replaced by their numeric values.
- A FROM clause and join conditions are added.

The following code snippet shows the SQL statement that is generated from the API query:

```
 SELECT DISTINCT ESCALATION.ESIID, ESCALATION.TKIID
   FROM    ESCALATION ESC, WORK_ITEM WI
   WHERE   ESC.ESIID = WI.OBJECT_ID
   AND     WI.REASON  = 10
   AND
  ( WI.OWNER_ID = 'MaryJones' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

**Example: using the queryAll method:**

This example shows how to use the queryAll method to retrieve all of the activities that belong to a process template.

The queryAll method is available only to users with system administrator or system monitor rights. The following code snippet shows the queryAll method call for the query to retrieve all of the activities that belong to the process template, sampleProcess:

```
queryAll( "DISTINCT ACTIVITY.AIID",
       "PROCESS_TEMPLATE.NAME = 'sampleProcess'",
       (String)null, (String)null, (Integer)null, (TimeZone)null )
```

The following code snippet shows the SQL query that is generated from the API query:

```
SELECT DISTINCT ACTIVITY.AIID
  FROM   ACTIVITY AI, PROCESS_TEMPLATE PT
  WHERE  AI.PTID = PT.PTID
  AND    PT.NAME = 'sampleProcess'
```

Because the call is invoked by an administrator, an access control condition is not added to the generated SQL statement. A join with the WORK_ITEM view is also not added. This means that the query retrieves all of the activities for the process template, including those activities without work items.

**Example: including query properties in a query:**

This example shows how to use the query method to retrieve tasks that belong to a business process. The process has query properties defined for it that you want to include in the search.

For example, you want to search for all of the human tasks in the ready state that belong to a business process. The process has a query property, **customerID**, with the value CID_12345, and a namespace. The following code snippet shows the query method call for the query:

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                  PROCESS_INSTANCE.NAME",
         " QUERY_PROPERTY.NAME = 'customerID' AND " +
         " QUERY_PROPERTY.STRING_VALUE = 'CID_12345' AND " +
         " QUERY_PROPERTY.NAMESPACE =
           'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
         " TASK.KIND IN
           ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
         " TASK.STATE = TASK.STATE.STATE_READY ",
         (String)null, (String)null, (Integer)null, (TimeZone)null );
```

If you now want to add a second query property to the query, for example, **Priority**, with a given namespace, the query method call for the query looks as follows:

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                  PROCESS_INSTANCE.NAME",
         " QUERY_PROPERTY1.NAME = 'customerID' AND " +
         " QUERY_PROPERTY1.STRING_VALUE = 'CID_12345' AND " +
         " QUERY_PROPERTY1.NAMESPACE =
                 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
         " QUERY_PROPERTY2.NAME = 'Priority' AND " +
         " QUERY_PROPERTY2.NAMESPACE =
                 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
         " TASK.KIND IN
           ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
         " TASK.STATE = TASK.STATE.STATE_READY ",
         (String)null, (String)null, (Integer)null, (TimeZone)null );
```

If you add more than one query property to the query, you must number each of the properties that you add as shown in the code snippet. However, querying custom properties affects performance; performance decreases with the number of custom properties in the query.

**Example: including custom properties in a query:**

This example shows how to use the query method to retrieve tasks that have custom properties.

For example, you want to search for all of the human tasks in the ready state that have a custom property, **customerID**, with the value CID_12345. The following code snippet shows the query method call for the query:

```
query ( " DISTINCT TASK.TKIID ",
        " TASK_CPROP.NAME = 'customerID' AND " +
        " TASK_CPROP.STRING_VALUE = 'CID_12345' AND " +
        " TASK.KIND IN
          ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
        " TASK.STATE = TASK.STATE.STATE_READY ",
        (String)null, (String)null, (Integer)null, (TimeZone)null );
```

If you now want to retrieve the tasks and their custom properties, the query method call for the query looks as follows:

```
query ( " DISTINCT TASK.TKIID, TASK_CPROP.NAME, TASK_CPROP.STRING_VALUE",
        " TASK.KIND IN
          ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
        " TASK.STATE = TASK.STATE.STATE_READY ",
        (String)null, (String)null, (Integer)null, (TimeZone)null );
```

The SQL statement that is generated from this API query is shown in the following code snippet:

```
SELECT DISTINCT TA.TKIID , TACP.NAME , TACP.STRING_VALUE
   FROM  TASK TA LEFT JOIN TASK_CPROP TACP ON (TA.TKIID = TACP.TKIID),
         WORK_ITEM WI
   WHERE WI.OBJECT_ID = TA.TKIID
   AND    TA.KIND IN ( 101, 105 )
   AND    TA.STATE = 2
   AND   (WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID IS NULL AND WI.EVERYBODY = 1 )
```

This SQL statement contains an outer join between the TASK view and the TASK_CPROP view. This means that tasks that satisfy the WHERE clause are retrieved even if they do not have any custom properties.

## 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.

**Note:** The views might contain columns that are not described. These columns are for internal use only.

**ACTIVITY view:**

Use this predefined database view for queries on activities.

*Table 17. 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. |
| 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) |

*Table 17. Columns in the ACTIVITY view (continued)*

| Column name | Type | Comments |
|---|---|---|
| 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>**TRUE** The activity is business relevant. You can view the activity status in Business Process Choreographer Explorer.<br><br>**FALSE** 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 18. 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. |
| VALUE | String | The value of the custom property. |

**ACTIVITY_SERVICE view:**

Use this predefined database view for queries on activity services.

*Table 19. 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 instance that is 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. |

*Table 19. Columns in the ACTIVITY_SERVICE view (continued)*

| Column name | Type | Comments |
|---|---|---|
| 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 20. 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. Possible values are:<br><br>**TRUE** The task is business relevant and it is audited.<br><br>**FALSE** 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. |
| 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 21. 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>**ACTION_CREATE_WORK_ITEM (1)**<br>Creates a work item for each escalation receiver.<br><br>**ACTION_SEND_EMAIL (2)**<br>Sends an e-mail to each escalation receiver.<br><br>**ACTION_CREATE_EVENT (3)**<br>Creates and publishes an event. |
| ACTIVATION_STATE | Integer | An escalation instance is created if the corresponding task reaches one of the following states:<br><br>**ACTIVATION_STATE_READY (2)**<br>Specifies that the human or participating task is ready to be claimed.<br><br>**ACTIVATION_STATE_RUNNING (3)**<br>Specifies that the originating task is started and running.<br><br>**ACTIVATION_STATE_CLAIMED (8)**<br>Specifies that the task is claimed.<br><br>**ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)**<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>**AT_LEAST_EXPECTED_STATE_CLAIMED (8)**<br>Specifies that the task is claimed.<br><br>**AT_LEAST_EXPECTED_STATE_ENDED (20)**<br>Specifies that the task is in a final state (FINISHED, FAILED, TERMINATED or EXPIRED).<br><br>**AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)**<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. |

*Table 21. Columns in the ESCALATION view  (continued)*

| Column name | Type | Comments |
|---|---|---|
| INCREASE_PRIORITY | Integer | Indicates how the task priority will be increased. Possible values are:<br><br>**INCREASE_PRIORITY_NO (1)**<br>    The task priority is not increased.<br><br>**INCREASE_PRIORITY_ONCE (2)**<br>    The task priority is increased once by one.<br><br>**INCREASE_PRIORITY_REPEATED (3)**<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) |
| 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 22. 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 23. 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. |

**ESC_TEMPL view:**

Use this predefined database view to query data for escalation templates.

*Table 24. Columns in the ESC_TEMPL view*

| Column name | Type | Comments |
|---|---|---|
| ESTID | String | The ID of the escalation template. |
| ACTION | Integer | The action triggered by the escalation. Possible values are:<br><br>**ACTION_CREATE_WORK_ITEM (1)**<br>    Creates a work item for each escalation receiver.<br><br>**ACTION_SEND_EMAIL (2)**<br>    Sends an e-mail to each escalation receiver.<br><br>**ACTION_CREATE_EVENT (3)**<br>    Creates and publishes an event. |
| ACTIVATION_STATE | Integer | An escalation instance is created if the corresponding task reaches one of the following states:<br><br>**ACTIVATION_STATE_READY (2)**<br>    Specifies that the human or participating task is ready to be claimed.<br><br>**ACTIVATION_STATE_RUNNING (3)**<br>    Specifies that the originating task is started and running.<br><br>**ACTIVATION_STATE_CLAIMED (8)**<br>    Specifies that the task is claimed.<br><br>**ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)**<br>    Specifies that the task is waiting for the completion of subtasks. |
| 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>**AT_LEAST_EXPECTED_STATE_CLAIMED (8)**<br>    Specifies that the task is claimed.<br><br>**AT_LEAST_EXPECTED_STATE_ENDED (20)**<br>    Specifies that the task is in a final state (FINISHED, FAILED, TERMINATED or EXPIRED).<br><br>**AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)**<br>    Specifies that all of the subtasks of the task are complete. |
| CONTAINMENT_ CTX_ID | String | If the escalation template belongs to an inline task template, the containment context is the process template. If the escalation template context belongs to a stand-alone task template, the containment context is the task template. |
| FIRST_ESTID | String | The ID of the first escalation template in a chain of escalation templates. |

*Table 24. Columns in the ESC_TEMPL view (continued)*

| Column name | Type | Comments |
|---|---|---|
| INCREASE_PRIORITY | Integer | Indicates how the task priority will be increased. Possible values are:<br><br>**INCREASE_PRIORITY_NO (1)**<br>　　The task priority is not increased.<br><br>**INCREASE_PRIORITY_ONCE (2)**<br>　　The task priority is increased once by one.<br><br>**INCREASE_PRIORITY_REPEATED (3)**<br>　　The task priority is increased by one each time the escalation repeats. |
| NAME | String | The name of the escalation template. |
| PREVIOUS_ESTID | String | The ID of the previous escalation template in a chain of escalation templates. |
| TKTID | String | The task template ID to which the escalation template belongs. |

**ESC_TEMPL_CPROP view:**

Use this predefined database view to query custom properties for escalation templates.

*Table 25. Columns in the ESC_TEMPL_CPROP view*

| Column name | Type | Comments |
|---|---|---|
| ESTID | String | The ID of the escalation template. |
| NAME | String | The name of the property. |
| TKTID | String | The task template ID to which the escalation template belongs. |
| DATA_TYPE | String | The type of the class for non-string custom properties. |
| VALUE | String | The value for custom properties of type String. |

**ESC_TEMPL_DESC view:**

Use this predefined database view to query multilingual descriptive data for escalation templates.

*Table 26. Columns in the ESC_TEMPL_DESC view*

| Column name | Type | Comments |
|---|---|---|
| ESTID | String | The ID of the escalation template. |
| LOCALE | String | The name of the locale associated with the description or display name. |
| TKTID | String | The task template ID to which the escalation template belongs. |
| 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 27. 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 28. Columns in the PROCESS_INSTANCE view*

| Column name | Type | Comments |
|---|---|---|
| PTID | ID | The process template ID. |
| PIID | ID | The process instance ID. |
| NAME | String | The name of the process instance. |
| STATE | Integer | The state of the process instance. Possible values 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_PIID | ID | The ID of the parent process instance. |
| PARENT_NAME | String | The name of the parent process instance. |
| TOP_LEVEL_PIID | ID | The process instance ID of the top-level process instance. If there is no top-level process instance, this is the process instance ID of the current 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. |

*Table 28. Columns in the PROCESS_INSTANCE view (continued)*

| Column name | Type | Comments |
|---|---|---|
| TEMPLATE_NAME | String | The name of the associated process template. |
| TEMPLATE_DESCR | String | Description of the associated process template. |
| RESUMES | Timestamp | The time when the process instance is to be resumed automatically. |

## PROCESS_TEMPLATE view:

Use this predefined database view for queries on process templates.

*Table 29. 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. |
| 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_SUPPORTS (4) |
| DISPLAY_NAME | String | The descriptive name of the process. |

## QUERY_PROPERTY view:

Use this predefined database view for queries on process-level variables.

*Table 30. 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. |
| 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 31. 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>**TRUE** The task is business relevant and it is audited.<br>**FALSE** The task is not business relevant and it is not audited. |
| COMPLETED | Timestamp | The time when the task completed. |

*Table 31. Columns in the TASK view  (continued)*

| Column name | Type | Comments |
|---|---|---|
| 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>**AUTH_NONE**<br>No authorization rights for the associated context object.<br><br>**AUTH_READER**<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. |
| HIERARCHY_ POSITION | Integer | Possible values are:<br><br>**HIERARCHY_POSITION_TOP_TASK (0)**<br>The top-level task in the task hierarchy.<br><br>**HIERARCHY_POSITION_SUB_TASK (1)**<br>The task is a subtask in the task hierarchy.<br><br>**HIERARCHY_POSITION_FOLLOW_ON_TASK (2)** 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. |

*Table 31. Columns in the TASK view  (continued)*

| Column name | Type | Comments |
|---|---|---|
| KIND | Integer | The kind of task. Possible values are:<br><br>**KIND_HUMAN (101)**<br>　　States that the task is a *collaboration task* that is created and processed by a human.<br><br>**KIND_WPC_STAFF_ACTIVITY (102)**<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><br>**KIND_ORIGINATING (103)**<br>　　States that the task is an *invocation task* that supports person-to-computer interactions, which enables people to create, initiate, and start services.<br><br>**KIND_PARTICIPATING (105)**<br>　　States that the task is a *to-do task* that supports computer-to-person interactions, which enable a person to implement a service.<br><br>**KIND_ADMINISTRATIVE (106)**<br>　　States that the task is an administration 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. |
| 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. |
| RESUMES | Timestamp | The time when the task is to be resumed automatically. |
| STARTED | Timestamp | The time when the task was started (STATE_RUNNING, STATE_CLAIMED). |
| STARTER | String | The principal ID of the task starter. |

*Table 31. Columns in the TASK view  (continued)*

| Column name | Type | Comments |
|---|---|---|
| STATE | Integer | The state of the task. Possible values are:<br><br>**STATE_READY (2)**<br>    States that the task is ready to be claimed.<br><br>**STATE_RUNNING (3)**<br>    States that the task is started and running.<br><br>**STATE_FINISHED (5)**<br>    States that the task finished successfully.<br><br>**STATE_FAILED (6)**<br>    States that the task did not finish successfully.<br><br>**STATE_TERMINATED (7)**<br>    States that the task has been terminated because of an external or internal request.<br><br>**STATE_CLAIMED (8)**<br>    States that the task is claimed.<br><br>**STATE_EXPIRED (12)**<br>    States that the task ended because it exceeded its specified duration.<br><br>**STATE_FORWARDED (101)**<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 32. Columns in the TASK_CPROP view*

| Column name | Type | Comments |
|---|---|---|
| TKIID | String | The task instance ID. |
| NAME | String | The name of the property. |

*Table 32. Columns in the TASK_CPROP view (continued)*

| Column name | Type | Comments |
|---|---|---|
| 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_DESC view:**

Use this predefined database view to query multilingual descriptive data for task objects.

*Table 33. 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 34. 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: **TRUE** The task is business relevant and it is audited. **FALSE** 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 34. 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>**AUTH_NONE**<br>No authorization rights for the associated context object.<br><br>**AUTH_READER**<br>Operations on the associated context object require reader authority, for example, reading the properties of a process instance. |
| DEFINITION_NAME | String | The name of the task template definition in the Task Execution Language (TEL) file. |
| DEFINITION_NS | String | The namespace of the task template definition in the TEL file. |
| 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>**KIND_HUMAN (101)**<br>States that the task is a *collaboration task* that is created and processed by a human.<br><br>**KIND_ORIGINATING (103)**<br>States that the task is an *invocation task* that supports person-to-computer interactions, which enables people to create, initiate, and start services.<br><br>**KIND_PARTICIPATING (105)**<br>States that the task is a *to-do task* that supports computer-to-person interactions, which enable a person to implement a service.<br><br>**KIND_ADMINISTRATIVE (106)**<br>States that the task is an administration 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>**STATE_STARTED (1)**<br>Specifies that the task template is available for creating task instances.<br><br>**STATE_STOPPED (2)**<br>Specifies that the task template is stopped. Task instances cannot be created from the task template in this state. |

*Table 34. Columns in the TASK_TEMPL view (continued)*

| Column name | Type | Comments |
|---|---|---|
| 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. |
| 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 35. 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 36. 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 37. Columns in the WORK_ITEM view*

| Column name | Type | Comments |
|---|---|---|
| WIID | ID | The work item ID. |

*Table 37. Columns in the WORK_ITEM view  (continued)*

| Column name | Type | Comments |
|---|---|---|
| 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. |
| OBJECT_TYPE | Integer | The type of the associated object. Possible values are:<br><br>**OBJECT_TYPE_ACTIVITY (1)**<br>    Specifies that the work item was created for an activity.<br><br>**OBJECT_TYPE_PROCESS_INSTANCE (3)**<br>    Specifies that the work item was created for a process instance.<br><br>**OBJECT_TYPE_TASK_INSTANCE (5)**<br>    Specifies that the work item was created for a task.<br><br>**OBJECT_TYPE_TASK_TEMPLATE (6)**<br>    Specifies that the work item was created for a task template.<br><br>**OBJECT_TYPE_ESCALATION_ INSTANCE (7)**<br>    Specifies that the work item was created for an escalation instance.<br><br>**OBJECT_TYPE_APPLICATION_ COMPONENT (9)**<br>    Specifies that the work item was created for an application component. |
| 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. |

*Table 37. Columns in the WORK_ITEM view (continued)*

| Column name | Type | Comments |
|---|---|---|
| REASON | Integer | The reason for the assignment of the work item. Possible values are:<br><br>REASON_POTENTIAL_STARTER (5)<br>REASON_POTENTIAL_INSTANCE_<br>    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) |
| 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.

**About this task**

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 the find the claim.

**Procedure**

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%'",
         (String)null, (Integer)null,
         (Integer)null, (TimeZone)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).

**About this task**

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.

**About this task**

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.

**Procedure**

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",
                (Integer)null, (TimeZone)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.

**About this task**

As the system administrator, you can manage private stored queries that belong to
a specific user.

**Procedure**

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",
                 (Integer)null, (TimeZone)null,
                 (List)null, (String)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",
                          (Integer)null, (Integer)null, (List)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.

**Procedure**

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",
                (Integer)null, (TimeZone)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.

**About this task**

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.

## Required roles for actions on process instances

Access to 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 |
| getActiveEventHandlers | x | | x |
| getActivityInstance | x | | x |
| getAllActivities | x | | x |
| getAllWorkItems | x | | x |
| getClientUISettings | x | x | x |
| getCustomProperties | x | x | x |
| getCustomProperty | x | x | x |
| getCustomPropertyNames | x | x | x |
| getFaultMessage | x | x | x |
| getInputClientUISettings | x | x | x |
| getInputMessage | x | x | x |
| getOutputClientUISettings | x | x | x |
| getOutputMessage | x | x | x |
| getProcessInstance | x | x | x |
| getVariable | x | x | x |
| getWaitingActivities | x | x | x |
| getWorkItems | x | | x |
| restart | | | x |
| resume | | | x |
| setCustomProperty | | x | x |

| Action | Caller's principal role | | |
|---|---|---|---|
| | Reader | Starter | Administrator |
| setVariable | | | x |
| suspend | | | x |
| transferWorkItem | | | x |

## Required roles for actions on business-process activities

Access to 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 |
| getVariableNames | x | x | x | x | x |
| getInputVariableNames | x | x | x | x | x |
| getOutputVariableNames | x | x | x | x | x |
| getWorkItems | x | x | x | x | x |
| setCustomProperty | | x | | x | x |
| setFaultMessage | | x | | x | x |
| setOutputMessage | | x | | x | x |
| setVariable | | | | | x |

| Action | Caller's principal role | | | | |
|---|---|---|---|---|---|
| | Reader | Editor | Potential owner | Owner | Administrator |
| transferWorkItem | | | | x<br><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.

**About this task**

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.

**About this task**

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.

**About this task**

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.

**Procedure**

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.EXCECUTION_MODE_MICROFLOW",
 "PROCESS_TEMPLATE.NAME",
  new Integer(50),
  (TimeZone)null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started by the call method.

2. Start the process with an input message of the appropriate type.

When you create the message, you must specify its message type name so that the message definition is contained.

```
ProcessTemplateData template = processTemplates[0];
//create a message for the single starting receive activity
ClientObjectWrapper input = process.createMessage
                            (template.getID(),
                             template.getInputMessageTypeName());
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.

**About this task**

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.

**Procedure**

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.EXCECUTION_MODE_MICROFLOW",
 "PROCESS_TEMPLATE.NAME",
  new Integer(50),
  (TimeZone)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.getInputMessageTypeName());
   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.

**Procedure**

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),
    (TimeZone)null);
```

The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started by the initiate method.

2. Start the process with an input message of the appropriate type.

   When you create the message, you must specify its message type name so that the message definition is contained. If you specify a process-instance name, it must not start with an underscore. If a process-instance name is not specified, the process instance ID (PIID) in String format is used as the name.

```
ProcessTemplateData template = processTemplates[0];
//create a message for the single starting receive activity
ClientObjectWrapper input = process.createMessage
                            (template.getID(),
                             template.getInputMessageTypeName());
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 human task, 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.

**Procedure**

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),
    (TimeZone)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.getInputMessageTypeName());
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 human task, 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.

**Before you begin**

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.

**About this task**

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.

**Procedure**

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; stand-alone 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 human task 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.

**Before you begin**

The caller must be an administrator of the process instance or a business process administrator.

**About this task**

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.

**Procedure**

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.

**Procedure**

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 of the process is run as if a fault had occurred on the top-level scope. If you terminate the process instance without compensation, the process instance is terminated immediately without waiting for activities, to-do tasks, or inline invocation 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.

**About this task**

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.

**Procedure**

1. List the process instances that are finished.

   ```
   QueryResultSet result =
       process.query("DISTINCT PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                           PROCESS_INSTANCE.STATE.STATE_FINISHED",
                   (String)null, (Integer)null, (TimeZone)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 human task activities

Human task 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.

**About this task**

When a human task activity is activated, both an activity instance and an associated to-do task are created. Handling of the human task activity and the work item management is delegated to Human Task Manager. Any state change of the activity instance is reflected in the task instance and vice versa.

A potential owner claims the activity. This person is responsible for providing the relevant information and completing the activity.

**Procedure**

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",
                   (String)null, (Integer)null, (TimeZone)null);
```

   This action returns a query result set that contains the activities that can be worked on by the logged-on person.

2. Claim the activity to be worked on:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 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.getOutputMessageTypeName());
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 human task 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 complete 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.

**About this task**

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 human task activities (to-do tasks). If the purchaser decides to order several books, this is equivalent to claiming the next human task 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 human task 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 behavior of all the human task activities in the process model must be set to participates.

Compare this example with the example that uses both the Business Flow Manager API and the Human Task Manager API.

**Procedure**

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",
                    (String)null, (Integer)null, (TimeZone)null);
...
//
//Claim the first activity
//
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 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(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
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 order number and claims the next activity in the sequence. 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 there are no more successor activities that can be assigned to this user, `Null` is returned.

If the process contains parallel paths that can be followed and these paths contain human task 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
  ...
}

aiid = successor.getAIID();
```

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.

**About this task**

You can use originating tasks to send the message to the activity.

**Procedure**

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].getInputMessageTypeName();

// create a message for the service to be called
  ClientObjectWrapper message =
        process.createMessage(vtid,atid,inputMessageTypeName);
  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.

**About this task**

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.

**Procedure**

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.getInputMessageTypeName());

    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. The results of long-running processes with one-way interfaces cannot be retrieved using the getOutputMessage method, because the process has no output. However, you can query the contents of variables, instead.

**About this task**

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.

**Procedure**

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",
                    (String)null, (Integer)null, (TimeZone)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.

**About this task**

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
**About this task**

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.

**Human task 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.

**Procedure**

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'",
                  (String)null, (Integer)null, (TimeZone)null);
```

This action returns the stopped activities for the CustomerOrder process instance.

2. Complete the activity, for example, a stopped human task activity.

In this example, an output message is passed.

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper output =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  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(aiid, 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
**About this task**

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.

**Procedure**

1. List the stopped activities.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                   (String)null, (Integer)null, (TimeZone)null);
```

This action returns the stopped activities for the CustomerOrder process instance.

2. Retry the execution of the activity, for example, a stopped human task activity.

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper input =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  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 a fault handler on the process level is run before the process state ends in 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 38. API methods for process templates*

| Method | Description |
|---|---|
| getProcessTemplate | Retrieves the specified process template. |
| queryProcessTemplates | Retrieves process templates that are stored in the database. |

## Process instances

The following API methods are related to starting process instances.

*Table 39. API methods are related to 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 40. 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. |
| 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 41. 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 the activity instance 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 42. API methods for variables and custom properties*

| Method | Description |
|---|---|
| getVariable | Retrieves the specified variable. |
| setVariable | Sets the specified variable. |
| 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.

**About this task**

For more information on the Human Task Manager API, see the Javadoc in the com.ibm.task.api package.

## Starting an invocation task that invokes a synchronous interface

An invocation task is associated with a Service Component Architecture (SCA) component. When the task is started, it invokes the SCA component. Start an invocation task synchronously only if the associated SCA component can be called synchronously.

**About this task**

Such an SCA component can, for example, be implemented as a microflow or as 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.

**Procedure**

1. Optional: List the task templates to find the name of the invocation 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),
  (TimeZone)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 invocation task that invokes an asynchronous interface

An invocation task is associated with a Service Component Architecture (SCA)
component. When the task is started, it invokes the SCA component. Start an
invocation task asynchronously only if the associated SCA component can be called
asynchronously.

**About this task**

Such an SCA component can, for example, be implemented as a long-running
process or a one-way operation.

This scenario creates an instance of a task template and passes some customer
data.

**Procedure**

1. Optional: List the task templates to find the name of the invocation 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),
    (TimeZone)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,
                         (ReplyHandlerWrapper)null);
```

## Creating and starting a task instance

This scenario shows how to create an instance of a task template that defines a collaboration task (also known as a *human task* in the API) and start the task instance.

**Procedure**

1. Optional: List the task templates to find the name of the collaboration 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),
  (TimeZone)null);
```

   The results are sorted by name. The query returns an array containing the first 50 sorted 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 collaboration 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,
                                       (ReplyHandlerWrapper)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 to-do tasks or collaboration tasks

To-do tasks (also known as *participating tasks* in the API) or collaboration tasks (also known as *human tasks* in the API) are assigned to various people in your organization through work items. To-do tasks and their associated work items are created, for example, when a process navigates to a human task activity.

**About this task**

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.

**Procedure**

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",
                (String)null, (Integer)null, (TimeZone)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 collaboration task instances (also known as *human tasks* in the API) or to-do task instances (also known as *participating tasks* in the API).

**Before you begin**

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.

**About this task**

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.

**Procedure**

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",
                                   (String)null,
                                   (Integer)null,
                                   (TimeZone)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 to-do task (also known as a *participating* task in the API) or a collaboration task (also known as a *human task* in the API) 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.

**About this task**

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.

**Procedure**

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",
                                   (String)null, (Integer)null, (TimeZone)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.

**Procedure**

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.

**Procedure**

1. List the task instances that are finished.

   ```
   QueryResultSet result =
       task.query("DISTINCT TASK.TKIID",
                   "TASK.STATE = TASK.STATE.STATE_FINISHED",
                    (String)null, (Integer)null, (TimeZone)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.

**About this task**

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.

**Procedure**

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'",
                    (String)null, (Integer)null, (TimeZone)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.

**About this task**

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 human task activity instance, a process instance, or a task instance. The reasons are derived from the role that the user has for the object. An object can have multiple work items because a user can have different roles in association with the object, and a work item is created for each of these roles. For example, a to-do task instance can have an administrator, reader, editor, and owner work item at the same time.

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'",
                                    (String)null, (Integer)null,
                                    (TimeZone)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'",
                                    (String)null, (Integer)null,
                                    (TimeZone)null);
if ( result.size() > 0 )
{
  result.first();
  // delete the work item
  task.deleteWorkItem((TKIID)(result.getOID(1)),
                      WorkItem.REASON_READER,"Smith");
}
```

Chapter 11. Developing EJB client applications  **453**

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'",
               (String)null, (Integer)null, (TimeZone)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.

**About this task**

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.

You can model ad-hoc To-do or Collaboration tasks by creating instances of the com.ibm.task.api.TaskModel class, and using them to either create a reusable task template, or directly create a run-once task instance. To create an instance of the TaskModel class, a set of factory methods is available in the com.ibm.task.api.ClientTaskFactory factory class. Modeling human tasks at runtime is based on the Eclipse Modeling Framework (EMF).

**Procedure**

1. Create an org.eclipse.emf.ecore.resource.ResourceSet using the createResourceSet factory method.

2. Optional: If you intend to use complex message types, you can either define them using the org.eclipse.xsd.XSDFactory that you can get using the factory method getXSDFactory(), or directly import an existing XML schema using the loadXSDSchema factory method .

   To make the complex types available to the WebSphere Process Server, deploy them as part of an enterprise application.

3. Create or import a Web Services Definition Language (WSDL) definition of the type javax.wsdl.Definition.

   You can create a new WSDL definition using the createWSDLDefinition method. Then you can add it a port type and operation. You can also directly import an existing WSDL definition using the loadWSDLDefinition factory method.

4. Create the task definition using the createTTask factory method.

   If you want to add or manipulate more complex task elements, you can use the com.ibm.wbit.tel.TaskFactory class that you can retrieve using the getTaskFactory factory method .

5. Create the task model using the createTaskModel factory method, and pass it the resource bundle that you created in the step 1 and which aggregates all other artifacts you created in the meantime.

6. Optional: Validate the model using the TaskModel validate method.

**Results**

Use one of the Human Task Manager EJB API create methods that have a **TaskModel** parameter to either create a reusable task template, or a run-once task instance.

> **Related concepts**
> "Task templates" on page 38
> A human task template contains the definition of a deployed task model that was created using WebSphere Integration Developer, or at runtime using the Business Process Choreographer APIs.

## 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.

**About this task**

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

**Procedure**

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" ),
          (Map)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:

```
atask.createTask( taskModel, (String)null, "HTM" );
```

• The following snippet creates a task template:

```
task.createTaskTemplate( taskModel, (String)null );
```

**Results**

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.

**About this task**

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

**Procedure**

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" ),
             (Map)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, "BOapplication", "HTM" );
```

- The following snippet creates a task template:

```
task.createTaskTemplate( taskModel, "BOapplication" );
```

**Results**

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.

**About this task**

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

**Procedure**

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", (String)null, (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. 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, "BOapplication", "HTM" );
```

- The following snippet creates a task template:

```
task.createTaskTemplate( taskModel, "BOapplication" );
```

**Results**

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.

**About this task**

The example runs only inside the context of the calling enterprise application, for which the resources are loaded.

**Procedure**

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", (String)null, (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. 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" );
     ```

**Results**

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.

# 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 43. API methods for task templates

| Method | Description |
|---|---|
| getTaskTemplate | Retrieves the specified task template. |
| createAndCallTask | Creates and runs a task instance from the specified task template and waits synchronously for the result. |
| createAndStartTask | Creates and starts a task instance from the specified task template. |
| createTask | Creates a task instance from the specified task template. |
| createInputMessage | Creates an input message for the specified task template. For example, create a message that can be used to start a task. |
| queryTaskTemplates | Retrieves task templates that are stored in the database. |

## Task instances

The following methods are available to work with task instances.

Table 44. API methods for task instances

| Method | Description |
|---|---|
| getTask | Retrieves a task instance; the task instance can be in any state. |
| callTask | Starts an invocation task synchronously. |
| startTask | Starts a task that has already been created. |
| suspend | Suspends the collaboration or to-do task. |
| resume | Resumes the collaboration or to-do task. |
| terminate | Terminates the specified task instance. If an invocation 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. |

*Table 44. API methods for task instances  (continued)*

| Method | Description |
|---|---|
| deleteWorkItem | Deletes the work item. |

## Escalations

The following methods are available to work with escalations.

*Table 45. 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 46. 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. |
| 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 to-do task, a collaboration task, an invocation task, or an administration task.

You cannot use all of the actions provided by the HumanTaskManager interface for all kinds of tasks. The following table shows the actions that you can carry out on each kind of task.

| | Kind of task | | | |
|---|---|---|---|---|
| Action | To-do task | Collaboration task | Invocation task | Administration task |
| callTask | | | X | |
| cancelClaim | X | X[1] | | |
| claim | X | X[1] | | |
| complete | X | X[1] | | X |
| completeWithFollowOnTask[4] | X | X[1] | | |
| completeWithFollowOnTask[5] | | X[3] | X[3] | |
| createFaultMessage | X | X | X | X |

| Action | Kind of task | | | |
|---|---|---|---|---|
| | To-do task | Collaboration task | Invocation task | Administration task |
| createInputMessage | X | X | X | X |
| createOutputMessage | X | X | X | X |
| createWorkItem | X | X[1] | X | X |
| delete | X[1] | X[1] | X | X[1] |
| deleteWorkItem | X | X[1] | X | X |
| getCustomProperty | X | X[1] | X | X |
| getDocumentation | X | X[1] | X | X |
| getFaultNames | X | X[1] | | |
| getFaultMessage | X | X[1] | X | |
| getInputMessage | X | X[1] | X | |
| getOutputMessage | X | X[1] | X | |
| getUsersInRole | X | X[1] | X | X |
| getTask | X | X[1] | X | X |
| getUISettings | X | X[1] | X | X |
| resume | X | X[1] | | |
| setCustomProperty | X | X[1] | X | X |
| setFaultMessage | X | X[1] | | |
| setOutputMessage | X | X[1] | | |
| startTask | X[1] | X[1] | X | X |
| startTaskAsSubtask[6] | X | X[1] | | |
| startTaskAsSubtask[7] | | X[3] | X[3] | |
| suspend | X | X[1] | | |
| suspendWithCancelClaim | X | X[1] | | |
| terminate | X[1] | X[1] | X[1] | |
| transferWorkItem | X | X[1] | X | X |
| update | X | X[1] | 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

# Developing applications for business processes and human tasks

People are involved in most business process scenarios. For example, a business process requires people interaction when the process is started or administered, or when human task activities are performed. To support these scenarios, you need to use both the Business Flow Manager API and the Human Task Manager API.

**About this task**

To involve people in business process scenarios, you can include the following task kinds in the business process:

- An inline invocation task (also known as an *originating task* in the API).

  You can provide an invocation task for every receive activity, for each onMessage element of a pick activity, and for each onEvent element of an event handler. This task then controls who is authorized to start a process or communicate with a running process instance.

- An administration task.

  You can provide an administration task to specify who is authorized to administer the process or perform administrative operations on the failed activities of the process.

- A to-do task (also known as a *participating task* in the API).

  A to-do task implements a human task activity. This type of activity allows you to involve people in the process.

Human task activities in the business process represent the to-do tasks that people perform in the business process scenario. You can use both the Business Flow Manager API and the Human Task Manager API to realize these scenarios:

- The business process is the container for all of the activities that belong to the process, including the human task activities that are represented by to-do tasks. When a process instance is created, a unique object ID (PIID) is assigned.

- When a human task activity is activated during the execution of the process instance, an activity instance is created, which is identified by its unique object ID (AIID). At the same time, an inline to-do task instance is also created, which is identified by its object ID (TKIID). The relationship of the human task activity to the task instance is achieved by using the object IDs:

  - The to-do task ID of the activity instance is set to the TKIID of the associated to-do task.
  - The containment context ID of the task instance is set to the PIID of the process instance that contains the associated activity instance.
  - The parent context ID of the task instance is set to the AIID of the associated activity instance.

- The life cycles of all inline to-do task instances are managed by the process instance. When the process instance is deleted, then the task instances are also deleted. In other words, all of the tasks that have the containment context ID set to the PIID of the process instance are automatically deleted.

## Determining the process templates or activities that can be started

A business process can be started by invoking the call, initiate, or sendMessage methods of the Business Flow Manager API. If the process has only one starting activity, you can use the method signature that requires a process template name as a parameter. If the process has more than one starting activity, you must explicitly identify the starting activity.

**About this task**

When a business process is modeled, the modeler can decide that only a subset of users can create a process instance from the process template. This is done by associating an inline invocation task to a starting activity of the process and by

specifying authorization restrictions on that task. Only the people that are potential starters or administrators of the task are allowed to create an instance of the task, and thus an instance of the process template.

If an inline invocation task is not associated with the starting activity, or if authorization restrictions are not specified for the task, everybody can create a process instance using the starting activity.

A process can have more than one starting activity, each with different people queries for potential starters or administrators. This means that a user can be authorized to start a process using activity A but not using activity B.

**Procedure**

1. Use the Business Flow Manager API to create a list of the current versions of process templates that are in the started state.

   **Tip:** The queryProcessTemplates method excludes only those process templates that are part of applications that are not yet started. So, if you use this method without filtering the results, the method returns all of the versions of the process templates regardless of which state they are in.

   ```
   // current timestamp in UTC format, converted to yyyy-mm-ddThh:mm:ss
   String now = (new UTCDate()).toXsdString();
   String whereClause = "PROCESS_TEMPLATE.STATE =
                         PROCESS_TEMPLATE.STATE.STATE_STARTED AND
                         PROCESS_TEMPLATE.VALID_FROM =
                         (SELECT MAX(VALID_FROM) FROM PROCESS_TEMPLATE
                          WHERE NAME=PROCESS_TEMPLATE.NAME AND
                          VALID_FROM <= TS('" + now + "'))";

   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
                                     ( whereClause,
                                       "PROCESS_TEMPLATE.NAME",
                                       (Integer)null, (TimeZone)null);
   ```
   The results are sorted by process template name.

2. Create the list of process templates and the list of starting activities for which the user is authorized.

   The list of process templates contains those process templates that have a single starting activity. These activities are either not secured or the logged-on user is allowed to start them. Alternatively, you might want to gather the process templates that can be started by at least one of the starting activities.

   **Tip:** A process administrator can also start a process instance. To get a complete list of templates, you also need to read the administration task template that is associated with the process template, and check whether the logged-on user is an administrator.

   ```
   List authorizedProcessTemplates = new ArrayList();
   List authorizedActivityServiceTemplates = new ArrayList();
   ```

3. Determine the starting activities for each of the process templates.

   ```
   for( int i=0; i<processTemplates.length; i++ )
   {
     ProcessTemplateData template = processTemplates[i];
     ActivityServiceTemplateData[] startActivities =
                   process.getStartActivities(template.getID());
   ```

4. For each starting activity, retrieve the ID of the associated inline invocation task template.

```
for( int j=0; j<startActivities.length; j++ )
{
  ActivityServiceTemplateData activity = startActivities[j];
  TKTID tktid = activity.getTaskTemplateID();
```

a. If an invocation task template does not exist, the process template is not
   secured by this starting activity.

   In this case, everybody can create a process instance using this start activity.

```
boolean isAuthorized = false;
    if ( tktid == null )
  {
    isAuthorized = true;
    authorizedActivityServiceTemplates.add(activity);
  }
```

b. If an invocation task template exists, use the Human Task Manager API to
   check the authorization for the logged-on user.

   In the example, the logged-on user is Smith. The logged-on user must be a
   potential starter of the invocation task or an administrator.

```
if ( tktid != null )
  {
    isAuthorized =
      task.isUserInRole
          (tkid, "Smith", WorkItem.REASON_POTENTIAL_STARTER) ||
        task.isUserInRole(tktid, "Smith", WorkItem.REASON_ADMINISTRATOR);

    if ( isAuthorized )
    {
      authorizedActivityServiceTemplates.add(activity);
    }
  }
```

   If the user has the specified role, or if people assignment criteria for the role
   are not specified, the isUserInRole method returns the value `true`.

5. Check whether the process can be started using only the process template
   name.

```
if ( isAuthorized && startActivities.length == 1 )
  {
    authorizedProcessTemplates.add(template);
  }
```

6. End the loops.

```
  } // end of loop for each activity service template
}   // end of loop for each process template
```

## Processing a single person workflow that includes human tasks

Some workflows are performed by only one person, for example, ordering books
from an online bookstore. This example shows how to implement the sequence of
actions for ordering the book as a series of human task activities (to-do tasks). Both
the Business Flow Manager and the Human Task Manager APIs are used to
process the workflow.

**About this task**

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 human task
activities (to-do tasks). If the purchaser decides to order several books, this is
equivalent to claiming the next human task activity. Information about the
sequence of tasks is maintained by Business Flow Manager, while the tasks
themselves are maintained by Human Task Manager.

Compare this example with the example that uses only the Business Flow Manager API.

**Procedure**

1. Use the Business Flow Manager API to get the process instance that you want to work on.

   In this example, an instance of the CustomerOrder process.

   ```
   ProcessInstanceData processInstance =
      process.getProcessInstance("CustomerOrder");
   String piid = processInstance.getID().toString();
   ```

2. Use the Human Task Manager API to query the ready to-do tasks (kind participating) that are part of the specified process instance.

   Use the containment context ID of the task to specify the containing process instance. For a single person workflow, the query returns the to-do task that is associated with the first human task activity in the sequence of human task activities.

   ```
   //
    // Query the list of to-do tasks that can be claimed by the logged-on user
    // for the specified process instance
    //
   QueryResultSet result =
        task.query("DISTINCT TASK.TKIID",
                   "TASK.CONTAINMENT_CTX_ID = ID('" + piid + "') AND
                    TASK.STATE = TASK.STATE.STATE_READY AND
                    TASK.KIND = TASK.KIND.KIND_PARTICIPATING AND
                    WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                   (String)null, (Integer)null, (TimeZone)null);
   ```

3. Claim the to-do task that is returned.

   ```
   if (result.size() > 0)
   {
    result.first();
    TKIID tkiid = (TKIID) result.getOID(1);
    ClientObjectWrapper input = task.claim(tkiid);
    DataObject activityInput = 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.

4. Determine the human task activity that is associated with the to-do task.

   You can use one of the following methods to correlate activities to their tasks.

   - The task.getActivityID method:
     ```
     AIID aiid = task.getActivityID(tkiid);
     ```
   - The parent context ID that is part of the task object:
     ```
     AIID aiid = null;
     Task taskInstance = task.getTask(tkiid);

     OID oid = taskInstance.getParentContextID();
     if ( oid != null and oid instanceof AIID )
     {
       aiid = (AIID)oid;
     }
     ```

5. When work on the task is finished, use the Business Flow Manager API to complete the task and its associated human task activity, and claim the next human task activity in the process instance.

   To complete the human task 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(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
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 human task activity and its associated to-do task,
// and claim the next human task activity
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

   This action sets an output message that contains the order number and claims the next human task activity in the sequence. 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 there are no more successor activities that can be assigned to this user, Null is returned.

   If the process contains parallel paths that can be followed and these paths contain human task 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.

6. Work on the next human task activity.

```
ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // read the values
  ...
}

aiid = successor.getAIID();
```

7. Continue with step 5 to complete the human task activity and to retrieve the next human task activity.

# Handling exceptions and faults

A BPEL process might encounter a fault at different points in the process.

**About this task**

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

**About this task**

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

**Procedure**

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",
                    (String)null, (Integer)null, (TimeZone)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 aiid = (AIID) result.getOID(1);
  ClientObjectWrapper faultMessage = process.getFaultMessage(aiid);
  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

**About this task**

If an activity causes a fault to occur, the fault type determines the actions that you can take to repair the activity.

**Procedure**

1. List the human task 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",
                   (String)null, (Integer)null, (TimeZone)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 aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);

  ProcessException excp = activity.getUnhandledException();
  if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException fault = (ApplicationFaultException)excp;
   String faultName = fault.getFaultName();
  }
}
```

# Chapter 12. Developing Web service API client applications

You can develop client applications that access business process applications and human task applications through Web services APIs.

**About this task**

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 426 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 446 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.

**Procedure**

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.

**About this task**

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.

**Related tasks**

"Copying files from the client CD" on page 479
The files necessary to access the Web services APIs are available on the WebSphere Process Server client CD.

## 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.

**About this task**

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:

**Procedure**

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.

**Results**

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"/>
```

**Related concepts**

"Adding security (Java Web services)" on page 487
You must secure Web service communications by implementing security mechanisms in your client application.

**Related tasks**

"Adding security (.NET)" on page 496
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.

## 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 you begin**

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.

**About this task**

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.

**Related tasks**

"Generating a proxy client (.NET)" on page 492
.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.

"Specifying the Web service endpoint address" on page 474
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.

**Publishing the business process WSDL:**

Use the administrative console to publish the WSDL file.

**Procedure**

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**.

**Results**

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.

**Procedure**

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**.

**Results**

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.

**About this task**

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.

**Procedure**

1. Launch the WebSphere Integration Developer 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.

"Creating helper classes for BPEL processes (.NET)" on page 493
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.

"Creating helper classes for BPEL processes (Java Web services)" on page 485
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.

# 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.

**Procedure**

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 HTMWS.wsdl files to that of the WebSphere application server that is hosting the Web services APIs.

**Related tasks**

"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.

"Publishing and exporting artifacts from the server environment" on page 474
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.

## 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.

**About this task**

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**.

**Related tasks**

"Copying files from the client CD" on page 479
The files necessary to access the Web services APIs are available on the WebSphere Process Server client CD.

**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.

**Procedure**
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://si.n1.ibm.com:9080/
                  BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
   ```

   **Related concepts**
   "Adding security (Java Web services)" on page 487
   You must secure Web service communications by implementing security mechanisms in your client application.

   **Related tasks**
   "Adding security (.NET)" on page 496
   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.

**Procedure**
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://si.n1.ibm.com:9081/
                  HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

**Related concepts**

"Adding security (Java Web services)" on page 487
You must secure Web service communications by implementing security mechanisms in your client application.

**Related tasks**

"Adding security (.NET)" on page 496
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.

**About this task**

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 you begin**

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.

**Procedure**

1. Add the appropriate WSDL file to your project:
   - For business processes:
     a. Unzip the exported file BPEContainer_*nodename_servername*_WSDLFiles.zip to a temporary directory.
     b. Import the subdirectory META-INF from the unzipped directory BPEContainer_*nodename_servername*.ear/b.jar.
   - For human tasks:
     a. Unzip the exported file TaskContainer_*nodename_servername*_WSDLFiles.zip to a temporary directory.
     b. Import the subdirectory META-INF from the unzipped directory TaskContainer_*nodename_servername*.ear/h.jar.

   A new directory wsdl and subdirectory structure are created in your project.
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.

   **Note:** If you cannot select the **Web services** option in the **Preferences** menu, you must first enable the required capabilities as follows: **Window** → **Preferences** → **Workbench** → **Capabilities**. Click on **Web Service Developer** and click **OK**. Then reopen the Preferences window and change the **Code Generation** option.
3. Select the BFMWS.WSDL or HTMWS.WSDL file located in the newly-created wsdl directory.
4. Right-click and choose **Web services** → **Generate client**.

   Before continuing with the remaining steps, ensure that the server has started.
5. On the Web Services window, click **Next** to accept all defaults.
6. On the Web Service Selection window, click **Next** to accept all defaults.
7. On the Client Environment Configuration window:
   a. Click **Edit** and change the Web service runtime option to IBM WebSphere
   b. Change the J2EE Version option to 1.4.
   c. Click **OK**.
   d. Click **Next**.
8. This step is only necessary if you need to generate a Web Services client that includes both Business Process and Human Task Web Services APIs, as there are duplicate methods in both WSDL files.

a. On the Web Service Proxy window, select Define custom mapping for namespace to package then click **OK**.
b. On the Web Service Client namespace to package mapping window, add the following namespaces and package:

For BFMWS.wsdl:

| Namespace | Package |
|---|---|
| http://www.ibm.com/xmlns/prod/websphere/business-process/types/6.0 | com.ibm.sca.bpe |
| http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0 | com.ibm.sca.bpe |
| http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0/Binding | com.ibm.sca.bpe |
| http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0 | com.ibm.sca.bpe |

For HTMWS.wsdl:

| Namespace | Package |
|---|---|
| http://www.ibm.com/xmlns/prod/websphere/human-task/types/6.0 | com.ibm.sca.task |
| http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0 | com.ibm.sca.task |
| http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0/Binding | com.ibm.sca.task |
| http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0 | com.ibm.sca.task |

If asked to confirm overwriting, click **YesToAll**.
9. Click **Finish**.

**Results**

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 you begin**

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.

**About this task**

**Procedure**
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.

**Related tasks**

"Creating a client application (Java Web services)" on page 487
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.

# 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.

**Before you begin**

To create helper classes, you must have exported the WSDL file of the Web services API from the WebSphere Process Server environment.

**About this task**

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>
   ```

   The WSDL file is created when you develop a process or human task module.

   To create the corresponding SOAP message in your client application 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 in your client application:

   ```
   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

   **Related tasks**

   "Exporting business objects" on page 477
   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.

   ⬆ 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 you begin**

Before starting to create a client application, generate the proxy client and any necessary helper classes.

**About this task**

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.

**Procedure**

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 proxy
      BFMIFProxy proxy = new BFMIFProxy();
 // 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 482
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 485
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.

"Using WSDL2Java to generate a proxy client" on page 484
WSDL2Java is a command-line tool that generates a proxy client. A proxy client make it easier to program client applications.

# 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 64
  Actions that you can take on human tasks depend on your authorization role. This role can be a system-level J2EE role or an instance-based role.

  "Authorization roles for business processes" on page 32
  A role is a set of people who share the same level of authorization. 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.

## Implementing the user name token

The user name token security mechanism provides user name and password credentials.

**About this task**

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:

**Procedure**

1. Create a security token:
   a. Open the **Deployment Editor** of your module
   b. Click the **WS Extension** tab.
   c. Under **Service References**, the following Web Service References may be listed:
      - service/BFMWSService for business processes
      - service/HTMWSService for human tasks

      Which are listed depends on whether BFMWS.wsdl (for business process), HTMWS.wsdl (for human tasks), or both, were added when generating the proxy client.
   d. For both service references:
      1) Select one of the **Service References**.
      2) Expand the **Request Generator Configuration** section.

3) Expand the **Security Token** subsection.

4) Click **Add**. The Security Token window opens.

5) In the **Name** field, type a name for the new security token: **UserNameTokenBFM** or **UserNameTokenHTM** .

6) In the **Token type** drop-down list, select **Username**. (The **Local name** field is automatically populated with a default value.)

7) Leave the **URI** field blank. No URI value is required for a user name token.

8) Click **OK**.

2. Create a token generator:

   a. Open the **Deployment Editor** of your module

   b. Click on the **WS Binding** tab

   c. Under **Service References**, the same Web Service References are listed as in the previous step:
   - service/BFMWSService for business processes
   - service/HTMWSService for human tasks

   d. For both service references:

      1) Select one of the **Service References**.

      2) Expand the **Security Request Generator Binding Configuration** section.

      3) Expand the **Token Generator** subsection.

      4) Click **Add**. The Token Generator window opens.

      5) In the **Name** field, type a name for the new token generator, such as "UserNameTokenGeneratorBFM" or "UserNameTokenGeneratorHTM".

      6) In the **Token generator class** field, ensure that the following token generator class is selected: **com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator**.

      7) In the **Security token** drop-down list, select the appropriate security token that you created earlier.

      8) Select the **Use value type** check box.

      9) In the **Value type** field, select **Username Token**. (The **Local name** field is automatically populated to reflect your choice of **Username Token**.)

      10) 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".

      11) If you choose **NonPromptCallbackHandler**, you must specify a valid user name and password in the corresponding field of the deployment descriptor.

      12) Click **OK**.

**Related tasks**

"Specifying the Web service endpoint address" on page 474
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.

**About this task**

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:

**Procedure**

1. In the Rational Application Developer environment available in WebSphere
   Integration Developer, choose **WS Binding** ⟶ **Security Request Generator
   Binding Configuration** ⟶ **Token Generator**.
2. Create a security token:
   a. Open the **Deployment Editor** of your module
   b. Click the **WS Extension** tab.
   c. Under **Service References**, the following **Web Service References** may be
      listed:
      - service/BFMWSService for business processes
      - service/HTMWSService for human tasks

      Which are listed depends on whether BFMWS.wsdl (for business process),
      HTMWS.wsdl (for human tasks), or both, were added when generating the
      proxy client.
   d. For both service references:
      1) Select one of the **Service References**.
      2) Expand the **Request Generator Configuration** section.
      3) Expand the **Security Token** subsection.
      4) Click **Add**. The Security Token window opens.
      5) In the **Name** field, type a name for the new security token:
         **LTPATokenBFM** or **LTPATokenHTM** .
      6) In the **Token type** drop-down list, select **LTPAToken**. (The **URI** and
         **Local name** fields are automatically populated with default values.)
      7) Click **OK**.
3. Create a token generator:
   a. Open the **Deployment Editor** of your module
   b. Click on the **WS Binding** tab
   c. Under **Service References**, the same Web Service References are listed as in
      the previous step:

- service/BFMWSService for business processes
- service/HTMWSService for human tasks

d. For both service references:

1) Select one of the **Service References**.

2) Expand the **Security Request Generator Binding Configuration** section.

3) Expand the **Token Generator** subsection.

4) Click **Add**. The Token Generator window opens.

5) In the **Name** field, type a name for the new token generator, such as "LTPATokenGeneratorBFM" or "LTPATokenGeneratorHTM".

6) In the **Token generator class** field, ensure that the following token generator class is selected: **com.ibm.wsspi.wssecurity.token.LTPATokenGenerator**.

7) In the **Security token** drop-down list, select the appropriate security token that you created earlier.

8) Select the **Use value type** check box.

9) In the **Value type** field, select **LTPAToken**. (The **URI** and **Local name** fields are automatically populated to reflect your choice of **LTPA Token**.)

10) In the **Call back handler** field, type either "com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler".

11) Click **OK**.

**Results**

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.

**About this task**

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.

## 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.

**Before you begin**

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.

**About this task**

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.

**Procedure**

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:proxyclient.cs c:\ws\bin\HTMWS.wsdl
   ```

2. Compile the proxy client as a Dynamic Link Library (DLL) file.

   **Related tasks**

   "Publishing WSDL files" on page 476
   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.

# 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.

**Before you begin**

To create helper classes, you must have exported the WSDL file of the Web services API from the WebSphere Process Server environment.

**About this task**

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.

**Procedure**

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 tasks**

"Exporting business objects" on page 477
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.

**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 you begin**

Before starting to create a client application, generate the proxy client and any necessary helper classes.

**About this task**

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.

**Procedure**

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 492
.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 493
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.

**About this task**

These security mechanisms can include user name token (user name and
password), or custom binary and XML-based security tokens.

**Procedure**

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.

**About this task**

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.Write( " | tableName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].tableName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | columnName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].columnName.PadLeft(20) );
      }
```

```
      Console.WriteLine();
      Console.Write( " | data type ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         QueryColumnInfoTypeType 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.Write("--> 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

**Related concepts**

"Select clause" on page 394
The select clause in the query function identifies the object properties that are to be returned by a query.

"Where clause" on page 395
The where clause in the query function describes the filter criteria to apply to the query domain.

"Order-by clause" on page 396
The order-by clause in the query function specifies the sort criteria for the query result set.

# 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.

**Related reference**

## 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).

**About this task**

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.

# Chapter 13. Developing JMS client applications

You can develop client applications that access business process applications through the Java Messaging Service (JMS) API.

**About this task**

## Introduction to JMS

WebSphere Process Server Version 6.1 supports asynchronous messaging, based on the Java Messaging Service (JMS) programming interface, as a method of communication.

JMS provides a common way for Java clients (client applications or J2EE applications) to create, send, receive, and read requests as JMS messages.

JMS is an asynchronous message-based interface that:
- Uses either **point-to-point or publish/subscribe messaging**. Message-based frameworks can push information to other applications without their requesting it explicitly. The same information can be delivered to many subscribers in parallel. The Business Process Choreographer's JMS interface supports point-to-point messaging only.
- Offers **rhythm independence**. JMS frameworks function asynchronously, but are also able to simulate a synchronous request/response mode. This allows source and target systems to work simultaneously without having to wait for each other. This ability is extremely useful for the Business Process Choreographer, as it provides the ability to interact asynchronously with long-running business processes.
- Supports **transactions**. Transactions enable client applications to handle groups of messages sent or received as a single atomic unit. JMS transactions run within the server's transaction. For the Business Process Choreographer's JMS interface, you typically send and receive a single message for each transaction.
- **Guarantees information delivery**. JMS frameworks can manage messages in transactional mode and ensure message delivery (though without any guarantee of timeliness of delivery). For the Business Process Choreographer, this reliable message delivery capability is particularly important because it is dealing with business processes.
- Ensures **interoperability between heterogeneous frameworks**. The source and target applications can operate in heterogeneous environments without having to handle problems of communication and execution related to their respective frameworks.
- **Makes exchanges more fluid**. Switching to message mode allows finer-grained information to be exchanged.

## Requirements for business processes

Business processes developed with the WebSphere Integration Developer to run on the Business Process Choreographer must conform to specific rules to be accessible through the JMS API.

The requirements are:

1. The interfaces of business processes 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 WebSphere Integration Developer.

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?

# Accessing the JMS interface

To send and receive messages through the JMS interface, an application must first create a connection to the BPC.cellname.Bus, create a session, then generate message producers and consumers.

**About this task**

The process server accepts Java Message Service (JMS) messages that follow the point-to-point paradigm. An application that sends or receives JMS messages must perform the following actions.

The following example assumes that the JMS client is executed in a managed environment (EJB, application client, or Web client container). If you want to execute the JMS client in a J2SE environment, refer to "IBM Client for JMS on J2SE with IBM WebSphere Application Server" at http://www-1.ibm.com/support/docview.wss?uid=swg24012804.

**Procedure**

1. Create a connection to the BPC.*cellname*.Bus. No preconfigured connection factory exists for a client application's requests: a client application can either use the JMS API's ReplyConnectionFactory or create its own connection factory, in which case it can use Java Naming and Directory Interface (JNDI) lookup to retrieve the connection factory. The JNDI-lookup name must be the same as the name specified when configuring the Business Process Choreographer's external request queue. The following example assumes the client application creates its own connection factory named "jms/clientCF".

```
//Obtain the default initial JNDI context.
Context initialContext = new InitialContext();

// Look up the connection factory.
// Create a connection factory that connects to the BPC bus.
// Call it, for example, "jms/clientCF".
// Also configure an appropriate authentication alias.
ConnectionFactory connectionFactory =
          (ConnectionFactory)initialcontext.lookup("jms/clientCF");

// Create the connection.
Connection connection = connectionFactory.createConnection();
```

2. Create a session so that message producers and consumers can be created.

```
// Create a transaction session using auto-acknowledgement.
Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
```

3. Create a message producer to send messages. The JNDI-lookup name must be the same as the name specified when configuring the Business Process Choreographer's external request queue.

```
// Look up the destination of the Business Process Choreographer input queue to
// send messages to.
Queue sendQueue = (Queue) initialcontext.lookup("jms/BFMJMSAPIQueue");

// Create a message producer.
MessageProducer producer = session.createProducer(sendQueue);
```

4. Create a message consumer to receive replies. The JNDI-lookup name of the reply destination can specify a user-defined destination, but it can also specify the default (Business Process Choreographer-defined) reply destination jms/BFMJMSReplyQueue. In both cases, the reply destination must lie on the BPC.<cellname>.Bus.

```
// Look up the destination of the reply queue.
Queue  replyQueue = (Queue) initialcontext.lookup("jms/BFMJMSReplyQueue");

// Create a message consumer.
MessageConsumer consumer = session.createConsumer(replyQueue);
```

5. Send a message.

```
// Start the connection.
connection.start();

// Create a message - see the task descriptions for examples - and send it.
// This method is defined elsewhere ...
String payload = createXMLDocumentForRequest();
TextMessage requestMessage = session.createTextMessage(payload);

// Set mandatory JMS header.
// targetFunctionName is the operation name of JMS API
// (for example, getProcessTemplate, sendMessage)
requestMessage.setStringProperty("TargetFunctionName", targetFunctionName);

// Set the reply queue; this is mandatory if the replyQueue
// is not the default queue (as it is in this example).
requestMessage.setJMSReplyTo(replyQueue);

// Send the message.
producer.send(requestMessage);

// Get the message ID.
String jmsMessageID = requestMessage.getJMSMessageID();

session.commit();
```

6. Receive the reply.

```
// Receive the reply message and analyse the reply.
TextMessage  replyMessage = (TextMessage) consumer.receive();

// Get the payload.
String payload = replyMessage.getText();

session.commit();
```

7. Close the connection and free the resources.

```
 // Final housekeeping; free the resources.
session.close();
connection.close();
```

**Note:** It is not necessary to close the connection after each transaction. Once a connection has been started, any number of request and response messages can be exchanged before the connection is closed. The example shows a simple case with a single call within a single business method.

# Structure of a Business Process Choreographer JMS message

The header and body of each JMS message must have a predefined structure.

A Java Message Service (JMS) message consists of:
- A message header for message identification and routing information.
- The body (payload) of the message that holds the content.

The Business Process Choreographer supports text message formats only.

**Message header**

JMS allows clients to access a number of message header fields.

The following header fields can be set by a Business Process Choreographer JMS client:
- **JMSReplyTo**

  The destination to send a reply to the request. If this field is not specified in the request message, the reply is sent to the Export interface's default reply destination (an Export is a client interface rendering of a business process component). This destination can be obtained using `initialContext.lookup("jms/BFMJMSReplyQueue");`
- **TargetFunctionName**

  The name of the WSDL operation, for example, "queryProcessTemplates". This field must always be set. Note that the TargetFunctionName specifies the operation of the generic JMS message interface described here. This should not be confused with operations provided by concrete processes or tasks that can be invoked indirectly, for example, using the **call** or **sendMessage** operations.

A Business Process Choreographer client can also access the following header fields:
- **JMSMessageID**

  Uniquely identifies a message. Set by the JMS provider when the message is sent. If the client sets the JMSMessageID before sending the message, it is overwritten by the JMS provider. If the ID of the message is required for authentication purposes, the client can retrieve the JMSMessageID after sending the message.
- **JMSCorrelationID**

  Links messages. Do not set this field. A Business Process Choreographer reply message contains the JMSMessageID of the request message.

Each response message contains the following JMS header fields:
- **IsBusinessException**

  "False" for WSDL output messages, or "true" for WSDL fault messages.

ServiceRuntimeExceptions are not returned to asynchronous client applications. When a severe exception occurs during the processing of a JMS request message, it results in a runtime failure, causing the transaction that is processing this request message to roll back. The JMS request message is then delivered again. If the failure occurs early, during processing of the message as part of the SCA Export (for example, while deserializing the message), retries are attempted up to the maximum number of failed deliveries specified by the SCA Export's receive destination. After the maximum number of failed deliveries is reached, the request message is added to the system exception destination of the Business Process

Choreographer bus. If, however, the failure occurs during actual processing of the request by the Business Flow Manager's SCA component, the failed request message is handled by the WebSphere Process Server's failed event management infrastructure, that is, it may end up in the failed event management database if retries do not resolve the exceptional situation.

**Message body**

The JMS message body is a String containing an XML document representing the document/literal wrapper element of the operation.

A simple example of a valid request message body is:

```
<?xml version="1.0" encoding="UTF-8"?>
<_6:queryProcessTemplates xmlns:_6="http://www.ibm.com/xmlns/prod/
       websphere/business-process/services/6.0">
<whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</_6:queryProcessTemplates>
```

**Related tasks**

"Checking the response message for business exceptions" on page 510
JMS client applications must check the message header of all response messages for business exceptions.

# Authorization for JMS renderings

To authorize use of the JMS interface, security settings must be enabled in WebSphere Application Server.

When the business process container is installed, the role **JMSAPIUser** must be mapped to a user ID. This user ID is used to issue all JMS API requests. For example, if **JMSAPIUser** is mapped to "User A", all JMS API requests appear to the process engine to originate from "User A".

The **JMSAPIUser** role must be assigned the following authorities:

| Request | Required authorization |
|---------|------------------------|
| forceTerminate | Process administrator |
| sendEvent | Potential activity owner or process administrator |

**Note:** For all other requests, no special authorizations are required.

Special authority is granted to a person with the role of business process administrator. A business process administrator is a special role; it is different from the process administrator of a process instance. A business process administrator has all privileges.

You cannot delete the user ID of the process starter from your user registry while the process instance exists. If you delete this user ID, the navigation of this process cannot continue. You receive the following exception in the system log file:

```
no unique ID for: <user ID>
```

# Overview of the JMS API

The JMS message interface (hereafter referred to as the "JMS API") allows you to develop client applications that asynchronously access business processes running in the Business Process Choreographer environment.

The JMS API allows client applications to asynchronously interact with microflows and long-running processes.

The JMS API exposes the same interface as the Web services API, with the following exceptions:

- With the Web services API, the `call` operation can only be used to invoke microflows. Using the JMS API, however, the `call` operation can be used to invoke both microflows and long-running processes.
- The following operations are not exposed through the JMS API:
    - The `callAsync` operation (together with its associated callback operations).
    - The `completeAndClaimSuccessor` and `getParticipatingTask` operations

## Example - executing a long-running process

For a generic client application to work with long-running processes, the sequence of steps is:

1. Set up the JMS environment, as described in "Accessing the JMS interface" on page 504.
2. Obtain a list of installed process definitions:
    - Send `queryProcessTemplates`
    - This returns a list of **ProcessTemplate** objects.
3. Obtain a list of start activities (receive or pick with `createInstance="yes"`):
    - Send `getStartActivities`.
    - This returns a list of **InboundOperationTemplate** objects.
4. Create an input message. This is environment-specific, and may require the use of predeployed, process-specific artifacts.
5. Create a process instance:
    - Issue a `sendMessage`.

    With the JMS API, you may also use the `call` operation for interacting with long-running request-response operations provided by a business process. This operation returns the operation result or fault to the specified reply-to destination, even after a long period of time. Therefore, if you use the `call` operation, you do not need to use the `query` and `getOutputMessage` operations to obtain the process' output or fault message.
6. Optionally, obtain output messages from the process instances by repeating the following steps:
    - Issue `query` to obtain the finished state of the process instance.
    - Issue `getOutputMessage` .
7. Optionally, work with additional operations exposed by the process:
    - `getWaitingActivities` or `getActiveEventHandlers` to obtain a list of **InboundOperationTemplate** objects.
    - Create input messages
    - Send messages with `sendMessage`

8. Optionally, get and set custom properties defined on the process or contained activities with `getCustomProperties` and `setCustomProperties`.
9. Optionally, finish working with a process instance:
   - Send `delete` and `terminate` to finish working with the long-running process.

# Developing JMS applications

JMS client applications must be developed in Java using the Java 2 Enterprise Edition (J2EE) environment.

**About this task**

JMS client applications exchange request and response messages with the JMS API. To create a request message, the client application fills a JMS TextMessage message body with an XML element representing the document/literal wrapper of the corresponding operation.

# Copying artifacts

A number of artifacts can be copied from the WebSphere environment to help in the creation of JMS client applications.

Use of these artifacts is only mandatory if the BOXMLSerializer is used to create the JMS message body.

There are two ways to obtain these artifacts:
- Publish and export them from the WebSphere Process Server environment.

  For WebSphere Process Server 6.1, all client artifacts are to be found in the *install_root*\ProcessChoreographer\client directory. For the JMS API, these artifacts are:

      BFMIF.wsdl
      BFMIF.xsd
      BPCGen.xsd
- Copy files from the WebSphere Process Server client CD.

## Publishing artifacts from the server environment
To help develop client applications that access the JMS API, you can publish a number of artifacts from the WebSphere server environment.

**About this task**

For WebSphere Process Server 6.1, all client artifacts are to be found in the *was_home*\ProcessChoreographer\client directory. For the JMS API, these artifacts are:

      BFMIF.wsdl
      BFMIF.xsd
      BPCGen.xsd

After these artifacts are published, copy them to your client programming environment.

## Copying files from the client CD
The files necessary to access the JMS API are available on the WebSphere Process Server client CD.

**Procedure**

1. Access the client CD and browse to the `ProcessChoreographer\client` directory.
2. Copy the necessary files to your client application development environment

   For WebSphere Process Server 6.1, all client artifacts are to be found in the \ProcessChoreographer\client directory. For the JMS API, these artifacts are:

   > BFMIF.wsdl
   >
   > BFMIF.xsd
   >
   > BPCGen.xsd

## Checking the response message for business exceptions

JMS client applications must check the message header of all response messages for business exceptions.

**About this task**

A JMS client application must first check the **IsBusinessException** property in the response message's header.

For example:

```
// receive response message
Message receivedMessage = ((JmsProxy) getToBeInvokedUponObject().receiveMessage();
String strResponse = ((TextMessage) receivedMessage).getText();

if (receivedMessage.getStringProperty("IsBusinessException") {
  // strResponse is a bussiness fault
  // any api can end w/a processFaultMsg
  // the call api also w/a businessFaultMsg
 }
else {
  // strResponse is the output message
}
```

**Related concepts**

"Structure of a Business Process Choreographer JMS message" on page 506
The header and body of each JMS message must have a predefined structure.

# Chapter 14. Developing Web applications for business processes and human tasks, using JSF components

Business Process Choreographer provides several JavaServer Faces (JSF) components. You can extend and integrate these components to add business-process and human-task functionality to Web applications.

**About this task**

You can use WebSphere Integration Developer to build your Web application.

**Procedure**

1. Create a dynamic project and change the Web Project Features properties 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

   If you are deploying your Web application on a remote server, also add the following files. These files are needed for remotely accessing the Business Process Choreographer APIs.

   - bpe137650.jar
   - task137650.jar

   In WebSphere Process Server, all of these files are in the following directory:

   - On Windows systems: *install_root*\ProcessChoreographer\client
   - On UNIX, Linux, and i5/OS systems: *install_root*/ProcessChoreographer/client

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 library references 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. To ensure correct error handling, you also need to define both an error bean and a navigation target for the error page in the JSF configuration file.

      ```
      <faces-config>
      ...
      <managed-bean>
        <managed-bean-name>BPCError</managed-bean-name>
        <managed-bean-class>com.ibm.bpc.clientcore.util.ErrorBeanImpl
        </managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
      </managed-bean>

      ...
      <navigation-rule>
      ...
      <navigation-case>
      <description>
      The general error page.
      </description>
      <from-outcome>error</from-outcome>
      <to-view-id>/Error.jsp</to-view-id>
      </navigation-case>
      ...
      </navigation-rule>
      </faces-config>
      ```

      In error situations that trigger the error page, the exception is set on the error bean.

   e. Implement the custom code that you need to support the JSF components.

5. Deploy the application.

If you are deploying the application in a network deployment environment, change the target resource Java Naming and Directory Interface (JNDI) names to values where the Business Flow Manager and Human Task Manager APIs can be found in your cell.

- If your business process containers are configured on another server in the same managed cell, the names have the following structure:

  `cell/nodes/nodename/servers/servername/com/ibm/bpe/api/BusinessManagerHome`
  `cell/nodes/nodename/servers/servername/com/ibm/task/api/HumanTaskManagerHome`

- If your business process containers are configured on a cluster in the same cell, the names have the following structure:

  `cell/clusters/clustername/com/ibm/bpe/api/BusinessFlowManagerHome`
  `cell/clusters/clustername/com/ibm/task/api/HumanTaskManagerHome`

Map the EJB references to the JNDI names or manually add the references to the ibm-web-bnd.xmi file.

The following table lists the reference bindings and their default mappings.

*Table 47. 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 |

**Results**

Your deployed Web application contains the functionality provided by the Business Process Choreographer Explorer components.

**What to do next**

If you are using custom JSPs for the process and task messages, you must map the Web modules that are used to deploy the JSPs to the same servers that the custom JSF client is mapped to.

**Related concepts**

"Error handling in JSF components" on page 515
The JavaServer Faces (JSF) components exploit a predefined managed bean, BPCError, for error handling. In error situations that trigger the error page, the exception is set on the error bean.

**Related tasks**

"Accessing the remote interface of the session bean" on page 388
An EJB client application accesses the remote interface of the session bean through the remote home interface of the bean.

# 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.
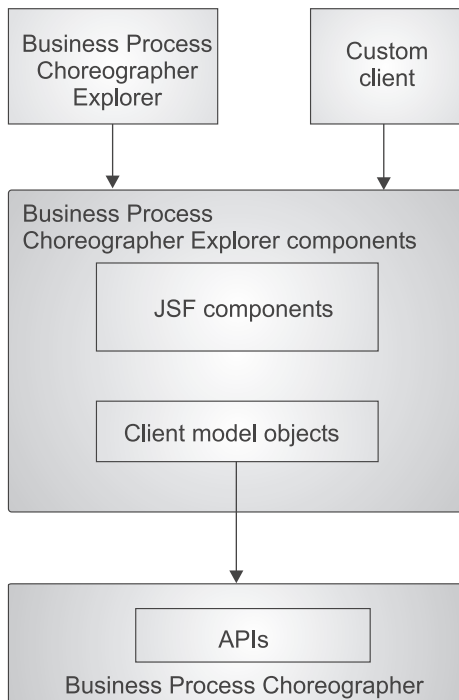


## JSF components

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.

# Error handling in JSF components

The JavaServer Faces (JSF) components exploit a predefined managed bean, BPCError, for error handling. In error situations that trigger the error page, the exception is set on the error bean.

This bean implements the com.ibm.bpc.clientcore.util.ErrorBean interface. 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();
}
```

**Related concepts**

"Error handling in the List component" on page 521
When you use the List component to display lists in your JSF application, you
can take advantage of the error handling functions provided by the
com.ibm.bpe.jsf.handler.BPCListHandler class.

# Default converters and labels for client model objects

The client model objects implement the corresponding interfaces of the Business
Process Choreographer API.

The List component and the Details component operate on any bean. You can
display all of the properties of a bean. However, if you want to set the converters
and labels that are used for the properties of a bean, you must use either the
`column` tag for the List component, or the `property` tag for the Details component.
Instead of setting the converters and labels, you can define default converter and
labels for the properties by defining the following static methods. You can define
the following static methods:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
        getConverter(String property);
```

The following table shows the client model objects that implement the
corresponding Business Flow Manager and Human Task Manager API classes and
provide default labels and converter for their properties. 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 48. How Business Process Choreographer interfaces are mapped to client model objects*

| Business Process Choreographer interface | Client model object class |
|---|---|
| com.ibm.bpe.api.ActivityInstanceData | com.ibm.bpe.clientmodel.bean.ActivityInstanceBean |
| com.ibm.bpe.api.ActivityServiceTemplateData | com.ibm.bpe.clientmodel.bean.ActivityServiceTemplateBean |
| com.ibm.bpe.api.ProcessInstanceData | com.ibm.bpe.clientmodel.bean.ProcessInstanceBean |
| com.ibm.bpe.api.ProcessTemplateData | com.ibm.bpe.clientmodel.bean.ProcessTemplateBean |
| com.ibm.task.api.Escalation | com.ibm.task.clientmodel.bean.EscalationBean |
| com.ibm.task.api.Task | com.ibm.task.clientmodel.bean.TaskInstanceBean |
| com.ibm.task.api.TaskTemplate | com.ibm.task.clientmodel.bean.TaskTemplateBean |

# 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.

**Procedure**

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>TaskPoolQuery</managed-bean-name>
<managed-bean-class>sample.TaskPoolQuery</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>

<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.

You can add any type of query logic to the JSF application as long as the result of the query can be represented as list of strongly-typed beans. For example, the TaskPoolQuery is implemented using a list of com.ibm.task.clientmodel.bean.TaskInstanceBean objects.

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 TaskPoolQuery 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) 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)",
      (String)null,
      (Integer)null,
      (TimeZone)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.

**Results**

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.

**Related concepts**

"User-specific time zone information" on page 520
The JavaServer Faces (JSF) components provide a utility for handling user-specific time zone information in the List component.

**Related reference**

"List component: Tag definitions" on page 521
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.

# How lists are processed

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 to associate the list entries with the details pages for the different kinds of items. 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. You can register implementations of this interface in the configuration file of your JavaServer Faces (JSF) application.

The notification is triggered when a link in the list is clicked. Links are rendered for all of the columns for which the **action** attribute is set. The value of the **action** attribute is either a JSF navigation target, or a JSF action method that returns a JSF navigation target.

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 runtime, 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().

### Default converters and labels

The items returned by a query must be beans and their class must match the class specified as the type in the definition of the BPCListHandler class or com.ibm.bpc.clientcore.Query interface. In addition, the List component checks whether the item class or a superclass implements the following methods:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
        getConverter(String property);
```

If these methods are defined for the beans, the List component uses the label as the default label for the list and the SimpleConverter as the default converter for the property. You can overwrite these settings with the **label** and **converterID** attributes of the bpe:list tag. For more information, see the Javadoc for the SimpleConverter interface and the ColumnTag class.

## User-specific time zone information

The JavaServer Faces (JSF) components provide a utility for handling user-specific time zone information in the List component.

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();
}
```

# Error handling in the List component

When you use the List component to display lists in your JSF application, you can take advantage of the error handling functions provided by the com.ibm.bpe.jsf.handler.BPCListHandler class.

## 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.

  **Related concepts**

  "Error handling in JSF components" on page 515
  The JavaServer Faces (JSF) components exploit a predefined managed bean, BPCError, for error handling. In error situations that trigger the error page, the exception is set on the error bean.

# 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 renders all of the columns for each of the objects.

*Table 49. bpe:list attributes*

| Attribute | Required | Description |
|---|---|---|
| buttonStyleClass | no | The cascading style sheet (CSS) style class for rendering the buttons in the footer area. |
| cellStyleClass | no | The CSS style class for rendering individual table cells. |
| checkbox | no | Determines whether the check box for selecting multiple items is rendered. The attribute has a value of either true or false. If the value is set to true, the check box column is rendered. |
| headerStyleClass | no | The CSS style class for rendering the table header. |
| model | yes | A value binding for a managed bean of the com.ibm.bpe.jsf.handler.BPCListHandler class. |
| 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. |
| rowClasses | no | The CSS style class for rendering the rows in the table. |
| selectAll | no | If this attribute is set to true, all of the items in the list are selected by default. |

*Table 49. bpe:list attributes (continued)*

| Attribute | Required | Description |
|-----------|----------|-------------|
| styleClass | no | The CSS style class for rendering the overall table containing titles, rows, and paging buttons. |

*Table 50. bpe:column attributes*

| Attribute | Required | Description |
|-----------|----------|-------------|
| action | no | If this attribute is specified, a link is rendered in the column. Either a JavaServer Faces action method or the Faces navigation target is triggered when this link is clicked. A JavaServer Faces action method has the following signature: String method(). |
| converterID | no | The Faces converter ID that is used for converting the property value. If this attribute is not set, any Faces converter ID that is provided by the model for this property is used. |
| label | no | A literal or value binding expression that is used as a label for the header of the column or the cell of the table header row. If this attribute is not set, any label that is provided by the model for this property is used. |
| name | yes | The name of the property that is displayed in this column. |

# 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.

**Procedure**

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.

   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. The bean class can be any JavaBeans class. If the bean provides default converter and property labels, the converter and the label are used for the rendering in the same way as for the List component.

   **Results**

   Your JSF application now contains a JavaServer page that displays the details of the specified object, for example, the details of a task instance.

   **Related reference**

   "Details component: Tag definitions"
   The Business Process Choreographer Explorer Details component displays the properties of tasks, work items, activities, process instances, and process templates.

# 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 51. bpe:details attributes*

| Attribute | Required | Description |
|---|---|---|
| columnClasses | no | A list of cascading style sheet style (CSS) style classes, separated by commas, for rendering columns. |
| id | no | The JavaServer Faces ID of the component. |
| model | yes | A value binding for a managed bean of the com.ibm.bpe.jsf.handler.BPCDetailsHandler class. |
| rowClasses | no | A list of CSS style classes, separated by commas, for rendering rows. |
| styleClass | no | The CSS class that is used for rendering the HTML element. |

*Table 52. bpe:property attributes*

| Attribute | Required | Description |
|---|---|---|
| converterID | no | The ID used to register the converter in the JavaServer Faces (JSF) configuration file. |
| label | no | The label for the property. If this attribute is not set, a default label is provided by the client model class. |
| 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. |

## 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.

**About this task**

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.

**Procedure**

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
   implements the Command interface. This command class (MyClaimCommand)
   is referred to by the `bpe:command` tag in the JSP file.

```
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.

d. If the **commandClass** attribute does not have a command class specified, the action is immediately called. For example, for the refresh command in the example, the JSF value expression `#{TaskInstanceList.refreshList}` is called instead of a command.

**Results**

Your JSF application now contains a JavaServer page that implements a customized command bar.

    **Related reference**

    "CommandBar component: Tag definitions" on page 528
    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.

# How commands are processed

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.

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.

# 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 53. bpe:commandbar attributes*

| Attribute | Required | Description |
|---|---|---|
| buttonStyleClass | no | The cascading style sheet (CSS) style class that is used for rendering the buttons in the command bar. |
| id | no | The JavaServer Faces ID of the component. |
| 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. |
| styleClass | no | The CSS style class that is used for rendering the command bar. |

*Table 54. bpe:command attributes*

| Attribute | Required | Description |
|---|---|---|
| action | no | A JavaServer Faces action method or the Faces navigation target that is to be triggered by the command button. The navigation target that is returned by the action overwrites all other navigation rules. The action is called when either an exception is not thrown or an ErrorsInCommandException exception is thrown by the command. |
| commandClass | no | The name of the command class. An instance of the class is created by the CommandBar component and run if the command button is selected. |
| commandID | yes | The ID of the command. |
| context | no | An object that provides context for commands that are specified using the **commandClass** attribute. The context object is retrieved when the command bar is first accessed. |
| immediate | no | Specifies when the command is triggered. If the value of this attribute is `true`, the command is triggered before the input of the page is processed. The default is `false`. |
| label | yes | The label of the button that is rendered in the command bar. |

*Table 54. bpe:command attributes  (continued)*

| Attribute | Required | Description |
|-----------|----------|-------------|
| rendered | no | Determines whether a button is rendered. The value of the attribute can be either a Boolean value or a value expression. |
| styleClass | no | The CSS style class that is used for rendering the button. This style overrides the button style defined for the command bar. |

# 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.

**About this task**

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.

**Procedure**

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.bpe.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
itemListener list in 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.

**Results**

Your JSF application now contains a JavaServer page that can render data objects and primitive types.

> **Related reference**
> "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.

# 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 55. bpe:form attributes*

| Attribute | Required | Description |
|---|---|---|
| id | no | The JavaServer Faces ID of the component. |
| model | yes | A value binding expression that refers to either a `commonj.sdo.DataObject` object or a `com.ibm.bpc.clientcore.MessageWrapper` object. |
| readOnly | no | If this attribute is set to `true`, a read-only form is rendered. By default, this attribute is set to `false`. |
| simplification | no | If this attribute is set to `true`, properties that contain simple types and have a cardinality of zero or one are shown. By default, this attribute is set to `true`. |
| 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. |
| styleClass4invalidInput | no | The CSS style class name for rendering input that is not valid. |
| styleClass4output | no | The CSS style class name for rendering the output elements. |

*Table 55. bpe:form attributes  (continued)*

| Attribute | Required | Description |
|-----------|----------|-------------|
| styleClass4table | no | The class name of the CSS table style for rendering the tables rendered by the message component. |
| styleClass4validInput | no | The CSS style class name for rendering input that is valid. |

# Chapter 15. Developing JSP pages for task and process messages

The Business Process Choreographer Explorer interface provides default input and output forms for displaying and entering business data. You can use JSP pages to provide customized input and output forms.

**About this task**

To include user-defined JavaServer Pages (JSP) pages in the Web client, you must specify them when you model a human task in WebSphere Integration Developer. For example, you can provide JSP pages 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 pages 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 all of the labels and input fields of a message.

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. The information is provided as a map that maps the XML Path Expression (XPath) of the properties that are not valid to the validation exceptions that occurred.

To add customized forms to Business Process Choreographer Explorer, complete the following steps using WebSphere Integration Developer.

**Procedure**

1. Create the customized forms.

   The user-defined JSP pages for the input and output forms used in the Web interface need access to the message data. Use Java snippets in a JSP or the JSP execution language to access the message data. Data in the forms is available through the request context.

2. Assign the JSP pages to a task.

   Open the human task in the human task editor. In the client settings, specify the location of the user-defined JSP pages 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 pages 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. If the JSPs are deployed separately, make the JSPs available on the server where the Business Process Choreographer Explorer or the custom client is deployed.

If you are using custom JSPs for the process and task messages, you must map the Web modules that are used to deploy the JSPs to the same servers that the custom JSF client is mapped to.

**Results**

The customized forms are rendered in Business Process Choreographer Explorer at runtime.

# User-defined JSP fragments

The user-defined JavaServer Pages (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>
```

# Chapter 16. 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.

**About this task**

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 people 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.

**About this task**

Complete the following steps to create an API event handler.

**Procedure**

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.`*`plug-in_name`*`APIEventHandlerPlugin`, where *plug-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.`**`Customer`**`APIEventHandlerPlugin`.

   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`.

**Results**

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 both API event handlers and notification event handlers. If you want to use both an API event handler and a notification event handler, the plug-in implementations must have the same name, for example, `Customer` as the event handler name for the SPI implementation.

You can implement both plug-ins using a single class, or two separate classes. In both cases, you need to create two files in the META-INF/services/ directory of your JAR file, for example, com.ibm.task.spi.CustomerNotificationEventHandlerPlugin and com.ibm.task.spi.CustomerAPIEventHandlerPlugin.

Package the plug-in implementation and the helper classes in a single JAR file.

**What to do next**

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.

   **Related concepts**

   "Scenarios for invoking tasks" on page 56
   The various ways in which tasks can be invoked is described here.

## 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.

**About this task**

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.

**Procedure**

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.`*`plug-in_name`*`NotificationEventHandlerPlugin`, where *plug-in_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.`**`HelpDeskRequest`**`NotificationEventHandlerPlugin`.

   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`.

**Results**

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 both API event handlers and notification event handlers. If you want to use both an API event handler and a notification event handler, the plug-in implementations must have the same name, for example, `Customer` as the event handler name for the SPI implementation.

You can implement both plug-ins using a single class, or two separate classes. In both cases, you need to create two files in the META-INF/services/ directory of your JAR file, for example, com.ibm.task.spi.CustomerNotificationEventHandlerPlugin and com.ibm.task.spi.CustomerAPIEventHandlerPlugin.

Package the plug-in implementation and the helper classes in a single JAR file.

**What to do next**

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.

>  **Related concepts**
>
>  "Escalations" on page 48
>  An escalation is an alert that is raised automatically when a human task is not actioned in the specified amount 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.

# Creating plug-ins to post-process people 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 people queries returned by people 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.

**About this task**

You can have only one post-processing plug-in; this means that the plug-in must handle the people query 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 people 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 people 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 people query results, you use the StaffQueryResultPostProcessorPlugin 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 people query results.

**Procedure**

1. Write a class that implements the StaffQueryResultPostProcessorPlugin 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_name*StaffQueryResultPostProcessorPlugin, 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.StaffQueryResultPostProcessorPlugin interface, the name of
      the configuration file is
      com.ibm.task.spi.**MyHandler**StaffQueryResultPostProcessorPlugin.

   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 post processes people query results 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 people 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 Manager (**Application servers** → *server_name* → **Human task
      container** → **Custom properties**).

b. Add a custom property with the name **Staff.PostProcessorPlugin**, and a value of the name that you gave to your plug-in, `MyHandler` in this example.

**Related concepts**

"Sharing people assignments" on page 87
For a specific task role, the same people assignment criteria are used in all instances of a task template. This is because all of the task instances are instantiated from the same task template. To avoid rerunning people queries, the result of a query is shared across task instances of a task template.

# Installing plug-ins

To use a plug-in, you must install the plug-in so that it can be accessed by the task container.

**About this task**

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.

**What to do next**

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.

**About this task**

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.

# Chapter 17. Installing business process and human task applications

You can distribute Service Component Architecture (SCA) modules that contain business processes or human tasks, or both, to deployment targets. A deployment target can be a server or a cluster.

**Before you begin**

Verify that Business Flow Manager, Human Task Manager, or both are installed and configured for each application server or cluster on which you want to install your application.

**About this task**

You can install business process and task applications from the administrative console, from the command line, or by running an administrative script, for example.

**Results**

After a business process or human task application is installed, all of the business process templates and human task templates are put into the start state. You can create process instances and task instances from these templates.

**What to do next**

Before you can create process instances or task instances, you must start the application.

**Related concepts**

"Deployment of business processes and human tasks" on page 548
When WebSphere Integration Developer or service deployment generates the deployment code for your process or task, each process component or task component is mapped to one session enterprise bean. All deployment code is packaged in the enterprise application (EAR) file. Additionally, for each process, a Java class which represents Java code in this process is generated and embedded in the EAR file during installation of the enterprise application. Each new version of a model that is to be deployed must be packaged in a new enterprise application.

"How business process and human task applications are installed in a network deployment environment"
When process templates or human task templates are installed in a network deployment environment, the following actions are performed automatically by the application installation.

## How business process and human task applications are installed in a network deployment environment

When process templates or human task templates are installed in a network deployment environment, the following actions are performed automatically by the application installation.

The application is installed asynchronously in stages. Each stage must complete successfully before the following stage can begin.

1. The application installation starts on the deployment manager.

   During this stage, the business process templates and human task templates are configured in the WebSphere configuration repository. The application is also validated. If errors occur, they are reported in the System.out file, in the System.err file, or as FFDC entries on the deployment manager.

2. The application installation continues on the node agent.

   During this stage, the installation of the application on one application server instance is triggered. This application server instance is either part of, or is, the deployment target. If the deployment target is a cluster with multiple cluster members, the server instance is chosen arbitrarily from the cluster members of this cluster. If errors occur during this stage, they are reported in the SystemOut.log file, in the SystemErr.log file, or as FFDC entries on the node agent.

3. The application runs on the server instance.

   During this stage, the process templates and human templates are deployed to the Business Process Choreographer database on the deployment target. If errors occur, they are reported in the System.out file, in the SystemErr.log file, or as FFDC entries on this server instance.

   **Related tasks**

   Chapter 17, "Installing business process and human task applications," on page 547
   You can distribute Service Component Architecture (SCA) modules that contain business processes or human tasks, or both, to deployment targets. A deployment target can be a server or a cluster.

# Deployment of business processes and human tasks

When WebSphere Integration Developer or service deployment generates the deployment code for your process or task, each process component or task component is mapped to one session enterprise bean. All deployment code is packaged in the enterprise application (EAR) file. Additionally, for each process, a Java class which represents Java code in this process is generated and embedded in the EAR file during installation of the enterprise application. Each new version of a model that is to be deployed must be packaged in a new enterprise application.

When you install an enterprise application that contains business processes or human tasks, then these are stored as business process templates or human task templates, as appropriate, in the Business Process Choreographer database. 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.

You can deploy many different versions of a process template or task template, each in a different enterprise application. When you install a new enterprise application, the version of the template that is installed is determined as follows:

- If the name of the template and the target namespace do not already exist, a new template is installed
- If the template name and target namespace are the same as those of an existing template, but the valid-from date is different, a new version of an existing template is installed

**Note:** The template name is derived from the name of the component and not from the business process or human task.

If you do not specify a valid-from date, the date is determined as follows:
- If you use WebSphere Integration Developer, the valid-from date is the date on which the human task or the business process was modeled.
- If you use service deployment, the valid-from date is the date on which the serviceDeploy command was run. Only collaboration tasks get the date on which the application was installed as the valid-from date.

  **Related tasks**

  Chapter 17, "Installing business process and human task applications," on page 547

  You can distribute Service Component Architecture (SCA) modules that contain business processes or human tasks, or both, to deployment targets. A deployment target can be a server or a cluster.

## Installing business process and human task 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.

**About this task**

Perform the following steps to install business process applications interactively.

**Procedure**

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.

**About this task**

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.

**Procedure**

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.
```

# Uninstalling business process and human task applications, using the administrative console

You can use the administrative console to uninstall applications that contain business processes or human tasks.

**Before you begin**

To uninstall an application that contains business processes or human tasks, the following prerequisites must be met:
- If the application is installed on a stand-alone server, the server must be running and have access to the Business Process Choreographer database.
- If the application is installed on a cluster, the deployment manager and at least one cluster member must be running. The cluster member have access to the Business Process Choreographer database.
- If the application is installed on managed server, the deployment manager and this server must be running. The server must have access to the Business Process Choreographer database.
- All of the business process templates and human task templates that belong to the application must be in the stopped state.
- There are no instances of business process or human task templates present in any state.
- 

For stand-alone server environments that are used as development and unit test environments, the server can be configured to run in development mode. This configuration does not require that the templates be stopped and no instances be present. However, this configuration is not valid for production environments.

**About this task**

To uninstall an enterprise application that contains business processes or human tasks, perform the following actions:

**Procedure**

1. Stop all process and task templates in the application.

    This action prevents the creation of process and task instances.

    a. Click **Applications** › **SCA modules** in the administrative console navigation pane.

    b. Select the module that contains the templates that you want to stop.

    c. Under Additional Properties, click **Business Processes** or **Human Tasks**, or both, as appropriate.

    d. Select all process and task templates by clicking the appropriate check box.

    e. 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 environment, the deployment manager, all 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 the application has no business process instances or human task instances.

   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.

**Results**

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.

**Before you begin**

To uninstall an application that contains business processes or human tasks, the following prerequisites must be met:

- If the application is installed on a stand-alone server, the server must be running and have access to the Business Process Choreographer database.
- If the application is installed on a cluster, the deployment manager and at least one cluster member must be running. The cluster member have access to the Business Process Choreographer database.
- If the application is installed on managed server, the deployment manager and this server must be running. The server must have access to the Business Process Choreographer database.
- All of the business process templates and human task templates that belong to the application must be in the stopped state.
- There are no instances of business process or human task templates present in any state.
- 

For stand-alone server environments that are used as development and unit test environments, the server can be configured to run in development mode. This configuration does not require that the templates be stopped and no instances be present. However, this configuration is not valid for production environments.

In addition, 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. To ensure that the administrative client automatically connects to the server process, do not use the `-conntype NONE` option as a command option.

**About this task**

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 the -force option deletes all process instance and task instance data, you should use this option with care.**

**Procedure**
1. Change to the Business Process Choreographer samples directory.

    On Windows platforms, enter:

    `cd install_root\ProcessChoreographer\admin`

    On Linux, UNIX, and i5/OS platforms, enter:

    `cd install_root/ProcessChoreographer/admin`
2. Stop the templates and uninstall the corresponding application.

    On Windows platforms, enter:

    ```
    install_root\bin\wsadmin -f bpcTemplates.jacl
                             [-user user_name]
                             [-password user password]
                             -uninstall application_name
                             [-force]
    ```

    On Linux, UNIX, and i5/OS platforms, enter:

    ```
    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.

**Results**

The application is uninstalled.

# Part 5. Monitoring business processes and tasks

# Chapter 18. Monitoring business processes and human tasks

**Before you begin**

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.

**About this task**

WebSphere Process Server includes the Common Event Infrastructure that provides standard formats and mechanisms for managing event data.

Business Process Choreographer emits 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.

# Chapter 19. Monitoring 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.

For Business Process Choreographer version 6.1 two event formats can occur:

**WebSphere Business Monitor 6.0.2 format**
WebSphere Business Monitor 6.0.2 format events occur when there are processes modeled within WebSphere Integration Developer 6.0.2, or if the WebSphere Business Monitor 6.0.2 format mode is enabled within WebSphere Integration Developer 6.1. If not specified otherwise, the object-specific content for these events is written as *extendedDataElement* XML elements of the type string.

**WebSphere Business Monitor 6.1 format**
WebSphere Business Monitor 6.1 format events occur when there are processes modeled within WebSphere Integration Developer 6.1, and the WebSphere Business Monitor 6.1 format mode is enabled. The object-specific content for these events is written as XML elements in the `xs:any` slot in the eventPointData folder of the Common Base Event, and the payload message is written to the *applicationData* section. The structure of the XML is defined in the XML Schema Definition (XSD) file BFMEvents.xsd. The file can be found in the `install_root\ ProcessChoreographer\client` directory.

### 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 | A string value that represents the state of the process. It has the format: *state number-state description*. This attribute can have one of the following values:<br><br>1 - STATE_READY<br>2 - STATE_RUNNING<br>3 - STATE_FINISHED<br>4 - STATE_COMPENSATING<br>5 - STATE_FAILED<br>6 - STATE_TERMINATED<br>7 - STATE_COMPENSATED<br>8 - STATE_TERMINATING<br>9 - STATE_FAILING<br>10 - STATE_INDOUBT<br>11 - STATE_SUSPENDED<br>12 - STATE_COMPENSATION_FAILED |
| principal | The principal on whose behalf the current execution step is executed. This is usually the starter of the process. |
| 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. If there is no explicit monitoring definition available, for example, if **Enable default events** is selected, the default payload type is 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 | A string value that identifies the activity kind. This value has the format: *kind number-kind description*. This attribute can have one of the following values:<br><br>3 - KIND_EMPTY<br>21 - KIND_INVOKE<br>23 - KIND_RECEIVE<br>24 - KIND_REPLY<br>25 - KIND_THROW<br>26 - KIND_TERMINATE<br>27 - KIND_WAIT<br>29 - KIND_COMPENSATE<br>30 - KIND_SEQUENCE<br>32 - KIND_SWITCH<br>34 - KIND_WHILE<br>36 - KIND_PICK<br>38 - KIND_FLOW<br>40 - KIND_SCOPE<br>42 - KIND_SCRIPT<br>43 - KIND_STAFF<br>44 - KIND_ASSIGN<br>45 - KIND_CUSTOM<br>46 - KIND_RETHROW<br>47 - KIND_FOR_EACH_SERIAL<br>48 - KIND_FOR_EACH_PARALLEL<br>1000 - SQLSnippet<br>1001 - RetrieveSet<br>1002 - InvokeInformationService<br>1003 - AtomicSQLSnippetSequence |
| state | A string value that represents the state of the activity. It has the format: *state number-state description*. Note that the state codes for activities are different from those used for processes. This attribute can have one of the following values:<br><br>1 - STATE_INACTIVE<br>2 - STATE_READY<br>3 - STATE_RUNNING<br>4 - STATE_SKIPPED<br>5 - STATE_FINISHED<br>6 - STATE_FAILED<br>7 - STATE_TERMINATED<br>8 - STATE_CLAIMED<br>9 - STATE_TERMINATING<br>10 - STATE_FAILING<br>11 - STATE_WAITING<br>12 - STATE_EXPIRED<br>13 - STATE_STOPPED |
| 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. If there is no explicit monitoring definition available, for example, if **Enable default events** is selected, the default payload type is full. |

## 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 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. If there is no explicit monitoring definition available, for example, if **Enable default events** is selected, the default payload type is 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. If the variable *variableName* has not been initialized, there is no *variableData* element.<br><br>This attribute is only used for WebSphere Business Monitor 6.0.2 format events. For WebSphere Business Monitor 6.1 format events, the content of the variable is written to the *applicationData* section, which contains one content element with the name set to the name of the variable. |
| 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. If the variable *variableName* has not been initialized, there is no *VariableData_BO* element.<br><br>This attribute is only used for WebSphere Business Monitor 6.0.2 format events. For WebSphere Business Monitor 6.1 format events, the content of the variable is written to the *applicationData* section, which contains one content element with the name set to the name of the variable. |

| Attribute | Description |
|---|---|
| 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. If there is no explicit monitoring definition available, for example, if **Enable default events** is selected, the default payload type is full. |

# Extension names for business process events

The extension name indicates the payload of the event. A list of all the extension names for business process events and their corresponding payload can be found here.

The 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 element that provides additional data about the event. The names of event elements are in uppercase, for example, BPC.BFM.BASE, and the names of XML elements are in mixed case, for example, *BPCEventCode*. Except where indicated, all data elements are of the type string.

The following extension names are available for business process events:

## BPC.BFM.BASE

BPC.BFM.BASE inherits the XML elements from WBIMonitoringEvent.

*Table 56. XML elements for BPC.BFM.BASE*

| XML element | Description |
|---|---|
| *BPCEventCode* | The Business Process Choreographer event code that identifies the event nature. |
| *processTemplateName* | The name of the process template. This name can differ from the display name. |
| *processTemplateValidFrom* | The valid from attribute of the process template. |
| *eventLocalCounter* | The local counter is used to discover the order of two events that occur in the same transaction. For a microflow instance, this counter reconstructs an order of all the emitted events. For long-running processes, the local counter indicates an order in the current navigation transaction. |

## BPC.BFM.PROCESS.BASE

BPC.BFM.PROCESS.STATUS inherits the XML elements from "BPC.BFM.BASE."

*Table 57. XML elements for BPC.BFM.PROCESS.BASE*

| XML element | Description |
|---|---|
| *processInstanceExecutionState* | The current execution state of the process in the following format: *<state code>-<state name>*. This attribute can have one of the following values:<br><br>1 - STATE_READY<br>2 - STATE_RUNNING<br>3 - STATE_FINISHED<br>4 - STATE_COMPENSATING<br>5 - STATE_FAILED<br>6 - STATE_TERMINATED<br>7 - STATE_COMPENSATED<br>8 - STATE_TERMINATING<br>9 - STATE_FAILING<br>10 - STATE_INDOUBT<br>11 - STATE_SUSPENDED<br>12 - STATE_COMPENSATION_FAILED |

## BPC.BFM.PROCESS.STATUS

BPC.BFM.PROCESS.STATUS inherits the XML elements from "BPC.BFM.PROCESS.BASE."

*Table 58. XML elements for BPC.BFM.PROCESS.STATUS*

| XML element | Description |
|---|---|
| *processTemplateId* | The ID of the process template. |
| *processInstanceDescription* | The description of the process instance. |

*Table 58. XML elements for BPC.BFM.PROCESS.STATUS (continued)*

| XML element | Description |
|---|---|
| *principal* | The name of the user who is associated with this event. |

## BPC.BFM.PROCESS.START

BPC.BFM.PROCESS.START inherits the XML elements from "BPC.BFM.PROCESS.STATUS" on page 564.

*Table 59. XML elements for BPC.BFM.PROCESS.START*

| XML element | Description |
|---|---|
| *username* | For BPC.BFM.PROCESS.START, this is the name of the user who requested the start or restart of the process. |

## BPC.BFM.PROCESS.FAILURE

BPC.BFM.PROCESS.FAILURE inherits the XML elements from "BPC.BFM.PROCESS.STATUS" on page 564.

*Table 60. XML elements for BPC.BFM.PROCESS.FAILURE*

| XML element | Description |
|---|---|
| *processFailedException* | The exception message that lead to the failure of the process. |

## BPC.BFM.PROCESS.CORREL

BPC.BFM.PROCESS.CORREL inherits the XML elements from "BPC.BFM.PROCESS.STATUS" on page 564.

*Table 61. XML elements for BPC.BFM.PROCESS.CORREL*

| XML element | Description |
|---|---|
| *correlationSet* | The correlation set instance, in the following format:<br><br>```<?xml version="1.0"?>```<br>```<correlationSet name="correlation set```<br>```            name">```<br>```  <property name="property name"```<br>```            value="property value"/>*```<br>```</correlationSet>``` |

## BPC.BFM.PROCESS.WISTATUS

BPC.BFM.PROCESS.WISTATUS inherits the XML elements from "BPC.BFM.PROCESS.STATUS" on page 564.

*Table 62. XML elements for BPC.BFM.PROCESS.WISTATUS*

| XML element | Description |
|---|---|
| *username* | For BPC.BFM.PROCESS.WISTATUS this is a list of users with work items that were created or deleted. |

## BPC.BFM.PROCESS.WITRANSFER

BPC.BFM.PROCESS.WITRANSFER inherits the XML elements from
"BPC.BFM.PROCESS.STATUS" on page 564.

*Table 63. XML elements for BPC.BFM.PROCESS.WITRANSFER*

| XML element | Description |
|---|---|
| *current* | The user name of the current owner of the work item. This is the user whose work item has been transferred to someone else. |
| *target* | The user name of the new owner of the work item. |

## BPC.BFM.PROCESS.ESCALATED

BPC.BFM.PROCESS.ESCALATED inherits the XML elements from
"BPC.BFM.PROCESS.STATUS" on page 564.

*Table 64. XML elements for BPC.BFM.PROCESS.ESCALATED*

| XML element | Description |
|---|---|
| *escalationName* | The name of the escalation. |
| *operation* | For BPC.BFM.PROCESS.ESCALATED, the operation that is associated with the event handler for which the inline invocation task is escalated. |
| *portTypeName* | The port type name of the operation that is associated with the event handler for which the inline invocation task is escalated. |
| *portTypeNamespace* | The port type namespace of the operation that is associated with the event handler for which the inline invocation task is escalated. |

## BPC.BFM.PROCESS.EVENT

BPC.BFM.PROCESS.EVENT inherits the XML elements from
"BPC.BFM.PROCESS.STATUS" on page 564.

*Table 65. XML elements for BPC.BFM.PROCESS.EVENT*

| XML element | Description |
|---|---|
| *message* or *message_BO-* | The input message 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.<br><br>This attribute is only used for WebSphere Business Monitor 6.0.2 format events. For WebSphere Business Monitor 6.1 format events, the content of the message is written to the *applicationData* section, which contains one content element with the name set to the name of the message. |
| *operation* | Name of the operation for the received event. |
| *portTypeName* | The port type name of the operation that is associated with the event handler. |
| *portTypeNamespace* | The port type namespace of the operation that is associated with the event handler. |

## BPC.BFM.PROCESS.PARTNER

BPC.BFM.PROCESS.PARTNER inherits the XML elements from
"BPC.BFM.PROCESS.STATUS" on page 564.

*Table 66. XML elements for BPC.BFM.PROCESS.PARTNER*

| XML element | Description |
|---|---|
| *partnerLinkName* | The name of the partner link. |

## BPC.BFM.ACTIVITY.BASE

BPC.BFM.ACTIVITY.BASE inherits the XML elements from "BPC.BFM.BASE" on
page 564.

*Table 67. XML elements for BPC.BFM.ACTIVITY.BASE*

| XML element | Description |
|---|---|
| *activityKind* | The activity kind, for example, sequence or invoke. The format is: *<kind code>-<kind name>*. This attribute can have one of the following values:<br><br>3 - KIND_EMPTY<br>21 - KIND_INVOKE<br>23 - KIND_RECEIVE<br>24 - KIND_REPLY<br>25 - KIND_THROW<br>26 - KIND_TERMINATE<br>27 - KIND_WAIT<br>29 - KIND_COMPENSATE<br>30 - KIND_SEQUENCE<br>32 - KIND_SWITCH<br>34 - KIND_WHILE<br>36 - KIND_PICK<br>38 - KIND_FLOW<br>40 - KIND_SCOPE<br>42 - KIND_SCRIPT<br>43 - KIND_STAFF<br>44 - KIND_ASSIGN<br>45 - KIND_CUSTOM<br>46 - KIND_RETHROW<br>47 - KIND_FOR_EACH_SERIAL<br>48 - KIND_FOR_EACH_PARALLEL<br>1000 - SQLSnippet<br>1001 - RetrieveSet<br>1002 - InvokeInformationService<br>1003 - AtomicSQLSnippetSequence |
| *state* | The current state of the activity instance in the format: *<state code>-<state name>*. This attribute can have one of the following values:<br><br>1 - STATE_INACTIVE<br>2 - STATE_READY<br>3 - STATE_RUNNING<br>4 - STATE_SKIPPED<br>5 - STATE_FINISHED<br>6 - STATE_FAILED<br>7 - STATE_TERMINATED<br>8 - STATE_CLAIMED<br>9 - STATE_TERMINATING<br>10 - STATE_FAILING<br>11 - STATE_WAITING<br>12 - STATE_EXPIRED<br>13 - STATE_STOPPED |
| *bpelId* | The wpc:id attribute of the activity in the BPEL file. It is unique for activities in a process model. |

## BPC.BFM.ACTIVITY.STATUS

BPC.BFM.ACTIVITY.STATUS inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 567.

*Table 68. XML elements for BPC.BFM.ACTIVITY.STATUS*

| XML element | Description |
|---|---|
| *activityTemplateName* | The name of the activity template. this can differ from the display name. |
| *activityTemplateId* | The internal ID of the activity template. |
| *activityInstanceDescription* | The description of the activity instance. |
| *principal* | The name of the user who claimed the activity. |

## BPC.BFM.ACTIVITY.FAILURE

BPC.BFM.ACTIVITY.FAILURE inherits the XML elements from
"BPC.BFM.ACTIVITY.STATUS."

*Table 69. XML elements for BPC.BFM.ACTIVITY.FAILURE*

| XML element | Description |
|---|---|
| *activityFailedException* | The exception that caused the activity to fail. |

## BPC.BFM.ACTIVITY.MESSAGE

BPC.BFM.ACTIVITY.MESSAGE inherits the XML elements from
"BPC.BFM.ACTIVITY.STATUS."

*Table 70. XML elements for BPC.BFM.ACTIVITY.MESSAGE*

| XML element | Description |
|---|---|
| *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.<br><br>This attribute is only used for WebSphere Business Monitor 6.0.2 format events. For WebSphere Business Monitor 6.1 format events, the content of the message is written to the *applicationData* section, which contains one content element with the name set to the name of the message. |

## BPC.BFM.ACTIVITY.CLAIM

BPC.BFM.ACTIVITY.CLAIM inherits the XML elements from
"BPC.BFM.ACTIVITY.STATUS."

*Table 71. XML elements for BPC.BFM.ACTIVITY.CLAIM*

| XML element | Description |
|---|---|
| *username* | For BPC.BFM.ACTIVITY.CLAIM this is the user for whom the task has been claimed. |

## BPC.BFM.ACTIVITY.WISTATUS

BPC.BFM.ACTIVITY.WISTATUS inherits the XML elements from "BPC.BFM.ACTIVITY.STATUS" on page 569.

*Table 72. XML elements for BPC.BFM.ACTIVITY.WISTATUS*

| XML element | Description |
|---|---|
| *username* | For BPC.BFM.ACTIVITY.WISTATUS this is a list users who are associated with the work item. |

## BPC.BFM.ACTIVITY.WITRANSFER

BPC.BFM.ACTIVITY.WITRANSFER inherits the XML elements from "BPC.BFM.ACTIVITY.STATUS" on page 569.

*Table 73. XML elements for BPC.BFM.ACTIVITY.WITRANSFER*

| XML element | Description |
|---|---|
| *current* | The user name of the current owner of the work item. This is the user whose work item has been transferred to someone else. |
| *target* | The user name of the new owner of the work item. |

## BPC.BFM.ACTIVITY.FOREACH

BPC.BFM.ACTIVITY.FOREACH inherits the XML elements from "BPC.BFM.ACTIVITY.STATUS" on page 569.

*Table 74. XML elements for BPC.BFM.ACTIVITY.FOREACH*

| XML element | Description |
|---|---|
| *parallelBranchesStarted* | The number of branches started. |

## BPC.BFM.ACTIVITY.ESCALATED

BPC.BFM.ACTIVITY.ESCALATED inherits the XML elements from "BPC.BFM.ACTIVITY.STATUS" on page 569.

*Table 75. XML elements for BPC.BFM.ACTIVITY.ESCALATED*

| XML element | Description |
|---|---|
| *escalationName* | The name of the escalation. |
| *operation* | For BPC.BFM.ACTIVITY.ESCALATED, the operation that is associated with the event handler for which the inline invocation task is escalated. |

## BPC.BFM.ACTIVITY.EVENT

BPC.BFM.ACTIVITY.EVENT inherits the XML elements from
"BPC.BFM.ACTIVITY.MESSAGE" on page 569.

*Table 76. XML elements for BPC.BFM.ACTIVITY.EVENT*

| XML element | Description |
|---|---|
| *operation* | The name of the operation for the received event. |

## BPC.BFM.LINK.STATUS

BPC.BFM.LINK.STATUS inherits the XML elements from "BPC.BFM.BASE" on
page 564.

*Table 77. XML elements for BPC.BFM.LINK.STATUS*

| XML element | Description |
|---|---|
| *elementName* | The name of the link. |
| *description* | The description of the link. |
| *flowBpelId* | The ID of the flow activity where the link is defined. |

## BPC.BFM.VARIABLE.STATUS

BPC.BFM.VARIABLE.STATUS inherits the XML elements from "BPC.BFM.BASE"
on page 564.

*Table 78. XML elements for BPC.BFM.VARIABLE.STATUS*

| XML element | Description |
|---|---|
| *variableName* | The name of the variable. |
| *variableData* or *variableData_BO* | If the variable *variableName* is not initialized, there is no *variableData* or *VariableData_BO* element. The variable's data is represented either as a String or business object (BO). The format depends on whether the **Monitor Compatible Events** option was selected on the **Event Monitor** tab in WebSphere Integration Developer.<br><br>This attribute is only used for WebSphere Business Monitor 6.0.2 format events. For WebSphere Business Monitor 6.1 format events, the content of the variable is written to the *applicationData* section, which contains one content element with the name set to the name of the variable. |
| *bpelId* | The Business Process Choreographer ID for the variable. |
| *principal* | The name of the user who updated the variable. |

# 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.

An event is emitted when the state of a process or activity changes. The following types of events can be caused by business process:
- "Process events"
- "Activity events" on page 574
- "Activity scope events" on page 577
- "Link events" on page 578
- "Variable events" on page 578

## XML Schema Definition (XSD) files

The event structure is described in the XML Schema Definition (XSD) file BFMEvents.xsd. The file can be found in the *install_root*\ProcessChoreographer\client directory.

## Key to table columns

The columns in the following tables contain:

**Code**    Contains the number of the event. For WebSphere Business Monitor 6.0.2 format events, the value is written to the Common Base Event as an extended data element with the name *BPCEventCode*. For WebSphere Business Monitor 6.1 format events, the value is written to the xs:any slot of the Common Base Event.

**Extension name**
> The *extensionName* contains a string value which defines the event specific information that is contained in the Common Base Event. This is also the name of the XML element that provides additional data about the event. For further information on extension names, see "Extension names for business process events" on page 563.

**Situation**
> Refers to the situation name of the business process event. For details of situations, see "Situations in business process events" on page 578.

**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 | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 21000 | BPC.BFM.PROCESS.START | Start | ENTRY | Process started |

| Code | Extension name | Situation | Event nature | Description |
|---|---|---|---|---|
| 21001 | BPC.BFM.PROCESS.STATUS | Report | SUSPENDED | Process suspended. To suspend process instances use the Business Process Choreographer Explorer. |
| 21002 | BPC.BFM.PROCESS.STATUS | Report | RESUMED | Process resumed. Only suspended processes can be resumed. To resume process instances use the Business Process Choreographer Explorer. |
| 21004 | BPC.BFM.PROCESS.STATUS | Stop | EXIT | Process completed |
| 21005 | BPC.BFM.PROCESS.STATUS | Stop | TERMINATED | Process terminated. To terminate process instances use the Business Process Choreographer Explorer. |
| 21019 | BPC.BFM.PROCESS.START | Report | RESTARTED | Process restarted |
| 21020 | BPC.BFM.PROCESS.STATUS | Destroy | DELETED | Process deleted |
| 42001 | BPC.BFM.PROCESS. FAILURE | Fail | FAILED | Process failed |
| 42003 | BPC.BFM.PROCESS.STATUS | Report | COMPENSATING | Process compensating. To compensate process instances use the Business Process Choreographer Explorer. |
| 42004 | BPC.BFM.PROCESS.STATUS | Stop | COMPENSATED | Process compensated |
| 42009 | BPC.BFM.PROCESS.STATUS | Report | TERMINATING | Process terminating |
| 42010 | BPC.BFM.PROCESS.STATUS | Report | FAILING | Process failing |
| 42027 | BPC.BFM.PROCESS.CORREL | Report | CORRELATION | Correlation set initialized. Emitted when a new correlation set for the process instance is initialized. This is for example the case, when a receive activity with an initiating correlation set receives a message. |

| Code | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 42041 | BPC.BFM.PROCESS. WISTATUS | Report | WI_DELETED | Process work item deleted |
| 42042 | BPC.BFM.PROCESS. WISTATUS | Report | WI_CREATED | Process work item created |
| 42046 | BPC.BFM.PROCESS.STATUS | Fail | COMPFAILED | Process compensation failed |
| 42047 | BPC.BFM.PROCESS.EVENT | Report | EV_RECEIVED | Process event received. To define the event use the process interface. The event is genereated when an event handler that is associated with a process is activated. |
| 42049 | BPC.BFM.PROCESS.ESCALATED | Report | EV_ESCALATED | Process event escalated. This event is genereated when an inline invocation task is escalated, that is defined on the process level and associated with an onEvent event handler. |
| 42056 | BPC.BFM.PROCESS. WITRANSFER | Report | WI_TRANSFERRED | Process work item transferred |
| 42058 | BPC.BFM.PROCESS.PARTNER | Report | PA_CHANGE | Process partner changed. This event is genereated when a new endpoint reference is assigned to a partner link. |

For process events, the following event correlation sphere identifiers have the following content:

- 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 | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 21006 | BPC.BFM.ACTIVITY.MESSAGE | Start | CREATED | Activity ready. This event is generated when a human task activity is started. |
| 21007 | For invoke activities: BPC.BFM.ACTIVITY. MESSAGE. For all other activity types: BPC.BFM.ACTIVITY.STATUS | Start | ENTRY | Activity started. For Invoke activities, business object payload is available. |
| 21011 | For invoke, human task, receive, and reply activities: BPC.BFM.ACTIVITY. MESSAGE. For pick activities: BPC.BFM.ACTIVITY.EVENT. For all other activity types: BPC.BFM.ACTIVITY.STATUS | Stop | EXIT | Activity completed. For invoke, human task, receive, and reply activities, business object payload is available. |
| 21021 | BPC.BFM.ACTIVITY.STATUS | Report | DEASSIGNED | Claim canceled. This event is generated when the claim for a human task activity is canceled.. |
| 21022 | BPC.BFM.ACTIVITY.CLAIM | Report | ASSIGNED | Activity claimed. This event is generated when a human task activity is claimed. |
| 21027 | BPC.BFM.ACTIVITY.STATUS | Stop | TERMINATED | Activity terminated. Long-running activities can be terminated as an effect of fault handling on the scope or process the activity is assigned to. |
| 21080 | BPC.BFM.ACTIVITY.FAILURE | Failed | FAILED | Activity failed |
| 21081 | BPC.BFM.ACTIVITY.STATUS | Report | EXPIRED | Activity expired. You can define expiration on invoke and inline human task activities. |

| Code | Extension name | Situation | Event nature | Description |
|---|---|---|---|---|
| 42005 | BPC.BFM.ACTIVITY.STATUS | Report | SKIPPED | Activity skipped. This event can only apply to activities that have join behavior defined. If the join behavior evaluates to false, then the activity is skipped and the skipped event is emitted. |
| 42012 | BPC.BFM.ACTIVITY.MESSAGE | Report | OUTPUTSET | Activity output message set. Business object payload is available. |
| 42013 | BPC.BFM.ACTIVITY.MESSAGE | Report | FAULTSET | Activity fault message set. Business object payload is available. |
| 42015 | BPC.BFM.ACTIVITY.STATUS | Stop | STOPPED | Activity stopped |
| 42031 | BPC.BFM.ACTIVITY.STATUS | Report | FRETRIED | Activity forcibly retried. To force activities to retry use the Business Process Choreographer Explorer. |
| 42032 | BPC.BFM.ACTIVITY.STATUS | Stop | FCOMPLETED | Activity forcibly completed. To force activities to complete use the Business Process Choreographer Explorer. |
| 42036 | BPC.BFM.ACTIVITY.MESSAGE | Report | EXIT | Activity has message received |
| 42037 | BPC.BFM.ACTIVITY.STATUS | Report | CONDTRUE | Loop condition true |
| 42038 | BPC.BFM.ACTIVITY.STATUS | Report | CONDFALSE | Loop condition false |
| 42039 | BPC.BFM.ACTIVITY. WISTATUS | Report | WI_DELETED | Work item deleted. This event can only apply to pick, inline human tasks, and receive events. |
| 42040 | BPC.BFM.ACTIVITY. WISTATUS | Report | WI_CREATED | Work items created. This event can only apply to pick, inline human tasks, and receive events. |

| Code | Extension name | Situation | Event nature | Description |
|---|---|---|---|---|
| 42050 | BPC.BFM.ACTIVITY.ESCALATED | Report | ESCALATED | Activity escalated. This event can only apply to pick, inline human tasks, and receive events. |
| 42054 | BPC.BFM.ACTIVITY. WISTATUS | Report | WI_REFRESHED | Activity work items refreshed. This event can only apply to pick, inline human tasks, and receive events. |
| 42055 | BPC.BFM.ACTIVITY. WITRANSFER | Report | WI_TRANSFERRED | Work item transferred. This event can only apply to pick, inline human tasks, and receive events. |
| 42057 | BPC.BFM.ACTIVITY. FOREACH | Report | BRANCHES_STARTED | For each - activity branches started |

For activity events, the following event correlation sphere identifiers have the following content:

- 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 | Extension name | Situation | Event nature | Description |
|---|---|---|---|---|
| 42020 | BPC.BFM.ACTIVITY.STATUS | Start | ENTRY | Scope started |
| 42021 | BPC.BFM.ACTIVITY.STATUS | Report | SKIPPED | Scope skipped |
| 42022 | BPC.BFM.ACTIVITY.FAILURE | Fail | FAILED | Scope failed |
| 42023 | BPC.BFM.ACTIVITY.STATUS | Report | FAILING | Scope terminating |
| 42024 | BPC.BFM.ACTIVITY.STATUS | Stop | TERMINATED | Scope terminated |
| 42026 | BPC.BFM.ACTIVITY.STATUS | Stop | EXIT | Scope completed |
| 42043 | BPC.BFM.ACTIVITY.STATUS | Report | COMPENSATING | Scope compensating |
| 42044 | BPC.BFM.ACTIVITY.STATUS | Stop | COMPENSATED | Scope compensated |
| 42045 | BPC.BFM.ACTIVITY.STATUS | Fail | COMPFAILED | Scope compensation failed |
| 42048 | BPC.BFM.ACTIVITY.EVENT | Report | EV_RECEIVED | Activity event received |
| 42051 | BPC.BFM.ACTIVITY.ESCALATED | Report | EV_ESCALATED | Scope event 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 have the following content:

- 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 | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 21034 | BPC.BFM.LINK.STATUS | Report | CONDTRUE | Link evaluated true |
| 42000 | BPC.BFM.LINK.STATUS | Report | CONDFALSE | Link evaluated false |

For link events, the following event correlation sphere identifiers have the following content:

- 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 | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 21090 | BPC.BFM.VARIABLE.STATUS | Report | CHANGED | Variable update. Business object payload is available. |

For the variable event, the following event correlation sphere identifiers have the following content:

- The ECSCurrentID provides the ID of the containing process.
- The ECSParentID is the ECSCurrentID before the process instance start event of the current process.

## 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` |

# Chapter 20. Monitoring 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.

For Business Process Choreographer version 6.1 two event formats can occur:

**WebSphere Business Monitor 6.0.2 format**
WebSphere Business Monitor 6.0.2 format events occur when there are tasks modeled within WebSphere Integration Developer 6.0.2, or if the WebSphere Business Monitor 6.0.2 format mode is enabled within WebSphere Integration Developer 6.1. If not specified otherwise, the object-specific content for these events is written as *extendedDataElement* XML elements of the type string.

**WebSphere Business Monitor 6.1 format**
WebSphere Business Monitor 6.1 format events occur when there are tasks modeled within WebSphere Integration Developer 6.1, and the WebSphere Business Monitor 6.1 format mode is enabled. The object-specific content for these events is written as XML elements in the `xs:any` slot in the eventPointData folder of the Common Base Event. The structure of the XML is defined in the XML Schema Definition (XSD) file HTMEvents.xsd. The file can be found in the *install_root*\ProcessChoreographer\client directory.

### Tasks

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 | The description of the task instance in the default locale. |
| 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. If there is no explicit monitoring definition available, the default payload type is 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. If there is no explicit monitoring definition available, the default payload type is full. |

# Extension names for human task events

The extension name indicates the payload of the human task event. A list of all the extension names for human task events and their corresponding payload can be found here.

The 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 element that provides additional data about the event. The names of event elements are in uppercase, for example BPC.HTM.BASE, and the names of XML elements are in mixed case, for example, *HTMEventCode*. Except where indicated, all data elements are of the type string.

The following extension names are available for human task events:
- "BPC.HTM.BASE" on page 583
- "BPC.HTM.TASK.BASE" on page 583
- "BPC.HTM.TASK.STATUS" on page 583
- "BPC.HTM.TASK.FOLLOW" on page 583
- "BPC.HTM.TASK.MESSAGE" on page 583
- "BPC.HTM.TASK.INTERACT" on page 584
- "BPC.HTM.TASK.FAILURE" on page 584
- "BPC.HTM.TASK.WISTATUS" on page 584
- "BPC.HTM.TASK.WITRANSFER" on page 584
- "BPC.HTM.ESCALATION.STATUS" on page 585
- "BPC.HTM.ESCALATION.WISTATUS" on page 585
- "BPC.HTM.ESCALATION.WITRANSFER" on page 585

## BPC.HTM.BASE

BPC.HTM.BASE inherits the XML elements from WBIMonitoringEvent.

*Table 79. XML elements for BPC.HTM.BASE*

| XML element | Description |
|---|---|
| *HTMEventCode* | The Business Process Choreographer event code that identifies the number of the event type. Possible event codes are listed in the following tables. |

## BPC.HTM.TASK.BASE

BPC.HTM.TASK.BASE inherits the XML elements from "BPC.HTM.BASE."

*Table 80. XML elements for BPC.HTM.TASK.BASE*

| XML element | Description |
|---|---|
| *taskTemplateId* | The ID of the template. |
| *taskTemplateName* | The name of the task template. This can differ from the display name. |
| *taskTemplateValidFrom* | The date and time from when the task template is valid. |

## BPC.HTM.TASK.STATUS

BPC.HTM.TASK.STATUS inherits the XML elements from "BPC.HTM.TASK.BASE."

*Table 81. XML elements for BPC.HTM.TASK.STATUS*

| XML element | Description |
|---|---|
| *taskInstanceDescription* | The description of the task. |

## BPC.HTM.TASK.FOLLOW

BPC.HTM.TASK.FOLLOW inherits the XML elements from "BPC.HTM.TASK.BASE."

*Table 82. XML elements for BPC.HTM.TASK.FOLLOW*

| XML element | Description |
|---|---|
| *followTaskId* | The ID of the task that was started as a follow-on task. |

## BPC.HTM.TASK.MESSAGE

BPC.HTM.TASK.MESSAGE inherits the XML elements from "BPC.HTM.TASK.STATUS."

*Table 83. XML elements for BPC.HTM.TASK.FOLLOW*

| XML element | Description |
|---|---|
| *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.<br><br>This attribute is only used for WebSphere Business Monitor 6.0.2 format events. For WebSphere Business Monitor 6.1 format events, the content of the message is written to the *applicationData* section, which contains one content element with the name set to the name of the message. |

## BPC.HTM.TASK.INTERACT

BPC.HTM.TASK.INTERACT inherits the XML elements from "BPC.HTM.TASK.STATUS" on page 583.

*Table 84. XML elements for BPC.HTM.TASK.INTERACT*

| XML element | Description |
|---|---|
| *username* | For BPC.HTM.TASK.INTERACT, this is the name of the user that is associated with the task. |

## BPC.HTM.TASK.FAILURE

BPC.HTM.TASK.FAILURE inherits the XML elements from "BPC.HTM.TASK.STATUS" on page 583.

*Table 85. XML elements for BPC.HTM.TASK.FAILURE*

| XML element | Description |
|---|---|
| *taskFailedException* | A string containing the *faultNameSpace* and *faultName* separated by a semicolon (;). |

## BPC.HTM.TASK.WISTATUS

BPC.HTM.TASK.WISTATUS inherits the XML elements from "BPC.HTM.TASK.STATUS" on page 583.

*Table 86. XML elements for BPC.HTM.TASK.WISTATUS*

| XML element | Description |
|---|---|
| *username* | For BPC.BPC.TASK.WISTATUS, this is a list of users who have work items that were created or deleted. |

## BPC.HTM.TASK.WITRANSFER

BPC.HTM.TASK.WITRANSFER inherits the XML elements from "BPC.HTM.TASK.STATUS" on page 583.

*Table 87. XML elements for BPC.HTM.TASK.WITRANSFER*

| XML element | Description |
|---|---|
| *current* | For BPC.HTM.TASK.WITRANSFER, this is the name of the current user. This is the user whose work item was transferred to someone else. |
| *target* | For BPC.HTM.TASK.WITRANSFER, this is the user name of the work item receiver. |

## BPC.HTM.ESCALATION.STATUS

BPC.HTM.ESCALATION.STATUS inherits the XML elements from "BPC.HTM.TASK.BASE" on page 583.

*Table 88. XML elements for BPC.HTM.ESCALATION.STATUS*

| XML element | Description |
|---|---|
| *escalationName* | The name of the escalation. |
| *escalationInstanceDescription* | The description of the escalation. |

## BPC.HTM.ESCALATION.WISTATUS

BPC.HTM.ESCALATION.WISTATUS inherits the XML elements from "BPC.HTM.ESCALATION.STATUS."

*Table 89. XML elements for BPC.HTM.ESCALATION.WISTATUS*

| XML element | Description |
|---|---|
| *username* | For BPC.HTM.ESCALATION.WISTATUS, this is a list of users who have work items that are escalated. |

## BPC.HTM.ESCALATION.WITRANSFER

BPC.HTM.ESCALATION.WITRANSFER inherits the XML elements from "BPC.HTM.ESCALATION.STATUS."

*Table 90. XML elements for BPC.HTM.ESCALATION.WITRANSFER*

| XML element | Description |
|---|---|
| *current* | For BPC.HTM.ESCALATION.WITRANSFER, this is the name of the current user. This is the user whose work item was transferred to someone else. |
| *target* | For BPC.HTM.ESCALATION.WITRANSFER, this is the user name of the work item receiver. |

# 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.

An event is emitted when the state of a task changes. The following types of events can be caused by human tasks:

- "Task events"
- "Escalation events" on page 587

**Note:** Events are only emitted for ad-hoc tasks if the business relevance flag is set to true in the task model.

Events for inline tasks are emitted as activity events. For a list of these events, see "Business process events" on page 572.

## XML Schema Definition (XSD) files

The event structure is described in the XML Schema Definition (XSD) file HTMEvents.xsd. The file can be found in the *install_root*\ProcessChoreographer\ client directory.

## Key to table columns

The columns in the following tables contain:

**Code** Contains the number of the event. For WebSphere Business Monitor 6.0.2 format events, the value is written to the Common Base Event as an extended data element with the name *HTMEventCode*. For WebSphere Business Monitor 6.1 format events, the value is written to the xs:any slot of the Common Base Event.

**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 588.

**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 | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 51001 | BPC.HTM.TASK. INTERACT | Report | CREATED | Task created |
| 51002 | BPC.HTM.TASK.STATUS | Destroy | DELETED | Task deleted |
| 51003 | BPC.HTM.TASK.STATUS | Start | ENTRY | Task started |

| Code | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 51004 | BPC.HTM.TASK.STATUS | Stop | EXIT | Task completed |
| 51005 | BPC.HTM.TASK.STATUS | Report | DEASSIGNED | Claim canceled |
| 51006 | BPC.HTM.TASK. INTERACT | Report | ASSIGNED | Task claimed |
| 51007 | BPC.HTM.TASK.STATUS | Stop | TERMINATED | Task terminated |
| 51008 | BPC.HTM.TASK. FAILURE | Fail | FAILED | Task failed |
| 51009 | BPC.HTM.TASK.STATUS | Report | EXPIRED | Task expired |
| 51010 | BPC.HTM.TASK.STATUS | Report | WAITFORSUBTASK | Waiting for subtasks |
| 51011 | BPC.HTM.TASK.STATUS | Stop | SUBTASKCOMPLETED | Subtasks completed |
| 51012 | BPC.HTM.TASK.STATUS | Report | RESTARTED | Task restarted |
| 51013 | BPC.HTM.TASK.STATUS | Report | SUSPENDED | Task suspended |
| 51014 | BPC.HTM.TASK.STATUS | Report | RESUMED | Task resumed |
| 51015 | BPC.HTM.TASK. FOLLOW | Report | COMPLETEDFOLLOW | Task completed and follow-on task started |
| 51101 | BPC.HTM.TASK.STATUS | Report | UPDATED | Task properties updated |
| 51103 | BPC.HTM.TASK. MESSAGE | Report | OUTPUTSET | Output message updated. Business object payload is available. |
| 51104 | BPC.HTM.TASK. MESSAGE | Report | FAULTSET | Fault message updated. Business object payload is available. |
| 51201 | BPC.HTM.TASK. WISTATUS | Destroy | WI_DELETED | Work item deleted |
| 51202 | BPC.HTM.TASK. WISTATUS | Report | WI_CREATED | Work items created |
| 51204 | BPC.HTM.TASK. WITRANSFER | Report | WI_TRANSFERRED | Work item transferred |
| 51205 | BPC.HTM.TASK. WISTATUS | Report | WI_REFRESHED | Work items refreshed |

For task events, the following identifiers of event correlation spheres have the following content:

- 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 | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 53001 | BPC.HTM.ESCALATION. STATUS | Report | ENTRY | Escalation fired |
| 53201 | BPC.HTM.ESCALATION. WISTATUS | Destroy | WI_DELETED | Work item deleted |
| 53202 | BPC.HTM.ESCALATION. WISTATUS | Report | WI_CREATED | Work item created |
| 53204 | BPC.HTM.ESCALATION. WITRANSFER | Report | WI_TRANS-FERRED | Escalation transferred |

| Code | Extension name | Situation | Event nature | Description |
|------|----------------|-----------|--------------|-------------|
| 53205 | BPC.HTM.ESCALATION. WISTATUS | Report | WI_REFRESH-ED | Work item refreshed |

For task events, the following identifiers of event correlation spheres have the following content:

- The ESCcurrentID provides the ID of the escalation.
- The ECSParentID provides the ID of the associated task instance.

## 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 |

| Situation name | Content of the Common Base Event | |
|---|---|---|
| Report | categoryName is set to ReportSituation. | |
| | situationType | |
| | Type | ReportSituation |
| | reasoningScope | EXTERNAL |
| | reportCategory | STATUS |

# Part 6. Tuning

# Chapter 21. Tuning business processes

Use this task to improve the performance of business processes.

**Before you begin**

After successfully running business processes, you can perform this task to improve performance.

**Procedure**

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 application.

   Applications can contain multiple processes. Because microflows perform better than long-running processes. If persistence is not necessary, and the functionality can be handled single threaded in one transaction, prefer modeling microflows. Also consider separating branches of a long-running process into microflows. Furthermore synchronous service invocations are usually faster than asynchronous service invocations. So for performance reasons, prefer synchronous service invocations although this is not the default behavior in long-running processes.

   In long-running processes you can change transaction boundaries. In most cases, performance can be improved by reducing the number of transaction boundaries. However, the optimal number of transaction boundaries can only be found out by performance testing. Also consider using parallel execution paths in your processes instead of serializing activities, and minimizing the size and complexity of data of that flows through your process as serializing and deserializing of data is also expensive. Also minimize the number of events that are emitted for logging.

4. 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 594. 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 606. 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.

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.

**Results**

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.

**About this task**

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.

## Balancing the hardware resources

You can improve the performance of long-running business processes by balancing the hardware resources.

**About this task**

Before starting to tune the system, verify that the computer used is well balanced, that is, that the available resources (CPU, memory and I/O) are in the correct proportions. A computer with one (or many) very fast CPUs but little memory or poor I/O performance will be hard to tune. For interruptible processes, good I/O performance provided by multiple, fast disk drives is as important as adequate processing power and sufficient memory.

For production systems, it is advisable to separate databases from application servers by using separate machines to run the DBMS and the application server. For high load or high availability configurations, consider using a WebSphere cluster on several machines for running the business processes, and a separate machine for the database.

**Procedure**

1. On the database machine, make sure that you allocate enough disks.

2. Allocate enough memory.

   The amount of memory to allocate depends on the platform:
   - For a 32-bit Windows system with 4 GB of physical memory and a local database management system, use the following memory allocation:
     - 512 MB for Windows systems
     - 768 MB for WebSphere Application Server
     - 1.5 GB for the database if you are using DB2. If you are using Oracle, no more than 1 GB for the System Global Area (SGA) and 500 MB for the Program Global Area (PGA).
   - For a 64-bit AIX system with 8 GB of physical memory and a local database management system, use the following memory allocation:
     - 512 MB for AIX systems
     - 1024 MB for WebSphere Application Server
     - 5 GB for the database. Allocate 4 GB for the process database and 1 GB for the messaging database or databases.

       **Tip:** To help ensure optimum performance, do not allocate all memory to the database, because the file caching, for example, also consumes memory. Avoid situations in which data must be swapped to disk because insufficient memory is available.
   - For an i5/OS system, use the Work with System Status (WRKSYSSTS) command to help you to prevent the system from paging memory. If a large number of page faults occur, take one, or more, of the following actions:
     a. Increase the memory available to the WebSphere Process Server or the Enterprise Service Bus server subsystem memory pool
     b. Move the WebSphere Process Server or the Enterprise Service Bus server to another memory pool
     c. Remove jobs from the WebSphere Process Server or the Enterprise Service Bus server subsystem memory pool
   - Fine-tune the heap size of the application server.

     **Note:** You cannot tune the heap size if your application server runs on an i5/OS system.
3. Observe the network utilization. The performance of applications also depends on the speed that messages can pass between the servers and the database server. Where possible, reduce latency in the network.
4. Move workload to other servers.

   Consider which applications or subsystems can be moved to other servers.

**Results**

Your computer hardware is now well balanced.

   **Related tasks**

   "Planning the BPEDB database" on page 108
   Plan the database for Business Process Choreographer.

   "Tuning the application server" on page 600
   Use this task to tune the application server.

# Specifying initial DB2 database settings

Use this task to specify initial DB2 database settings. Note that this information is provided only as an example.

**About this task**

**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 Network Deployment information center.

To achieve good database operation, specify the initial database settings. In addition, use two separate logical disks with different stripe sizes. Also use one database for each instance so that if the messaging engines use a database as the data store they can use the Business Process Choreographer database for messaging, or use a separate database machine for the messaging database.

**Note:** This information does not apply to DB2 UDB for iSeries because these settings are built-in to this database type.

**Procedure**

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.

   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.

On Windows, you can specify a destination drive. The command creates a database on the destination drive in a directory named the same as the default DB2 instance on the server. So, for example, if the database is to be created on drive D:, and the local default instance is DB2, the database data goes to D:\DB2. Thus, to create a DB2 database in the D: directory for Business Process Choreographer, enter the following command:

```
CREATE DATABASE BPEDB ON D: USING CODESET UTF-8 TERRITORY en-us;
```

On UNIX and Linux, enter the following command:

```
CREATE DATABASE BPEDB ON /wasdbfs USING CODESET UTF-8 TERRITORY en-us;
```

where /wasdbfs specifies a directory.

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 a single high-performance disk drive on a Windows system.

```
-- Scriptfile to create tablespaces for DB2 UDB
-- Replace occurence or @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( 'D:/BPE/AUDITLOG' );

CREATE TABLESPACE COMP MANAGED BY SYSTEM USING( 'D:/BPE/COMP' );
```

```
CREATE TABLESPACE INSTANCE MANAGED BY SYSTEM USING( 'D:/BPE/INSTANCE', 'D:/BPE/INSTANCE');

CREATE TABLESPACE WORKITEM MANAGED BY SYSTEM USING( 'D:/BPE/WORKITEM' );

CREATE TABLESPACE SCHEDTS MANAGED BY SYSTEM USING( 'D:/BPE/SCHEDTS' );
```

   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 600.

- 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.

**Results**

Your long-running processes are running as fast as possible under the current environment and loading conditions.

# Specifying initial Oracle database settings

The performance and scalability of an Oracle database is enhanced primarily by optimizing the layout of database files, allocating sufficient memory to the buffer cache to enable efficient caching, and tuning database parameters.

**Procedure**

1. Allocate enough space to the buffer caches.

   Use in-memory caching to produce low-latency response times for database accesses. This means that the buffer caches must be sufficiently large. Set the buffer cache size to at least 700 MB, then monitor cache usage and increase the cache size if necessary.

2. Size log files to reduce log file switches

   The transaction log of an Oracle instance resides in several files that are used in a round-robin fashion. The active log files are switched when one becomes full, allowing the last active log to be archived. Because switching log files is an expensive operation, size the log files so that these switches occur infrequently; 750 MB is a good starting value. Then, monitor the transaction rate and the average log size, and adjust this value as needed.

3. Tune the following database parameters.

   **UNDO_TABLESPACE**
   Make sure that the undo tablespace is not used to more than 70 percent (%) of its size limit.

   **OPEN_CURSORS**
   The default value for this parameter is 50. However, this is often insufficient. The highest value that you can use for this parameter depends on your operating system. Values up to 1000 are supported on most operating systems.

   **MAX_SHARED_SERVERS**
   Specifies the maximum number of shared server processes that can run simultaneously. Use this parameter to reserve process slots for other processes, such as dedicated servers. If a value for the MAX_SHARED_SERVERS parameter is specified, then it should be greater than, or equal to, the value of the SHARED_SERVERS parameter, and less than the value of the PROCESSES parameter. For example, if there are 150 concurrent users, a good starting value for this parameter is MAX_SHARED_SERVERS=70

# Planning messaging engine settings

Use this task to plan your initial settings for the messaging engines.

**About this task**

To achieve the best performance for long-running processes, tune the messaging system for maximum performance of persistent messages. For the data store back-end types, file store is preferred because it performs well. Use a database data store if your environment runs in a cluster and you cannot use a file store.

If you use the service integration capabilities of WebSphere Application Server, follow the instructions given in Setting tuning properties for service integration in the WebSphere Application Server Network Deployment information center, to set up and tune the data stores for the messaging engines.

**Results**

Your messaging engines are operating optimally.

> **Related tasks**
>
> "Planning the messaging engine database" on page 117
> You can improve performance by using a separate database for the messaging engine for the Business Process Choreographer bus.

# Tuning the application server

Use this task to tune the application server.

**Before you begin**

Before you start this task, you must have specified the initial settings for the database, as described in "Specifying initial DB2 database settings" on page 596.

**About this task**

To ensure that the business process container can perform optimally, you need to adjust the server settings.

**Procedure**

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: BPEDDataSourceDb2 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

      To view the activation specifications for the process server, in the administrative console, click **Resources** ⟶ **JMS Providers** ⟶ **Default messaging** ⟶ **Activation specifications**.
   c. For the Business Process Choreographer database, calculate the number of parallel JDBC connections required, $p = 1.1 * t$

      The value of $p$ must not be greater than the number of connections allowed by the database.

d.  For the messaging database, calculate the number of parallel JDBC connections required, $m = t + x$, where $x$ is the number of additional JMS sessions to allow for overload situations where additional messages are generated and must be served

e.  Set the SQL Statement cache size to 50

2.  Tune the JDBC provider settings for the Business Process Choreographer database (BPEDB).

a.  Set **Max Connections** to the value $p$. The value of $p$ must not be greater than the number of connections allowed by the database.

b.  Set the SQL **Statement cache size** to 300.

3.  Repeat step 2 to tune the JDBC provider settings for the SCA messaging engine data store.

4.  Tune the JDBC provider settings for the messaging database.

Set **Max Connections** to the value $m$.

5.  Tune the heap size.

Here are some guidelines for the size of the server heap on 32-bit systems. These guidelines do not apply to servers that run on i5/OS systems.

*   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.

For 64-bit systems, 1 to 2 GB is a reasonable size for the heap.

6.  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.

**Results**

The application server is tuned for improved performance.

**Related tasks**

"Balancing the hardware resources" on page 594
You can improve the performance of long-running business processes by balancing the hardware resources.

# Fine-tuning the database

Use this task to fine-tune the database.

**Before you begin**

The business process container and business processes must be running.

**About this task**

**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. However, this information does not apply to DB2 UDB for iSeries.

**Procedure**

1.  Assign sizes to buffer pools according to their usage and hit ratio

The buffer pool hit ratio indicates the percentage of database requests that can be satisfied from data that is already in the pool. It should be close to 100 percent, but any value over 90 percent is acceptable. Increase the **SIZE** parameter for a buffer pool until you get a satisfactory hit rate. Monitor the total memory allocation. If you make the buffer pool too large, the system starts to swap. In this case, decrease the size of the buffer pool, or make additional memory available.

If you are using DB2 Version 8, you can calculate the buffer pool hit ratio. You can obtain the values that are needed for the calculation from buffer pool snapshots. Use the following command to get the snapshots:

```
DB2 get snapshot for all bufferpools
```

For more information on calculating the hit ratio, refer to the DB2 V8 information center.

If you are using DB2 Version 9, use the BP_HITRATIO administrative view to retrieve the hit ratio information. For more information on this view, refer to the DB2 V9 information center.

The DB2 Configuration Advisor suggests values for buffer pool sizes, the Business Process Choreographer database, by default, uses only the IBMDEFAULTBP default buffer pool. You can set the size of this buffer pool using the following command:

```
DB2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE 120000
```

This command displays buffer pools with their size in pages, and the size of each page:

```
DB2 select BPNAME, NPAGES, PAGESIZE from syscat.bufferpools
```

2. Tune the lock list space to help ensure optimum performance.

All locks require storage, and this storage is limited. Transactions that requesting locks beyond this limit must be aborted, and they therefore degrade the performance.

a. 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 sqldEscalateLocks 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 lock space has been exceeded.

b. Increase the values of the MAXLOCKS and LOCKLIST parameters.

These parameters control the behavior of the database in lock escalations. A lock escalation converts several individual row-level locks in the same table to a single-table-level lock.

If a transaction uses more than the value of the MAXLOCKS parameter of the lock list, the database manager converts these locks to a single table lock in order not to exceed the lock space limitation. However, the lock escalation increases the probability of deadlocks considerably. Therefore, increase the value of the MAXLOCKS parameter to 60 percent.

Increase the value of the LOCKLIST parameter 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
```

3. 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.

   - Tune the following DB2 parameters:

   **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.

   **DLCHKTIME**

   This parameter specifies the deadlock detection time frame. The
   default is 10 seconds.

   **LOCKTIMEOUT**

   This parameter specifies the time an application waits for a lock. The
   default is -1, which means that the application waits until the lock is
   granted, or a deadlock occurs. The value of this parameter should be
   always greater than the value of the DLCHKTIME parameter so that
   a deadlock is reported as a deadlock and not as a lock timeout. A
   good initial value for this parameter is 30 seconds. You might want
   to set the value higher if load tests show transaction times to be
   longer than 30 seconds.

   **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.

4. 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
   might suggest other values.

   **NUM_IOCLEANERS**

   For query-only applications set the value to 0, for regular processing
   use values between 1 and the number of disk drives in the system (see
   also the NUM_IOSERVERS parameter). For large buffer pools a higher
   number is usually beneficial.

   **NUM_IOSERVERS**

   Must match the number of physical disks that the database resides on.
   You should have at least as many IO servers as you have disks. IO
   servers do not use many system resources, so it is better to set a value
   that is too high rather than too low.

5. Update DB2 statistics for your database.

After initially putting load on your system, or whenever the data volume in the database changes significantly, consider updating the DB2 system catalog tables that contain the statistics. Use the RUNSTATS command to update the statistics.

The RUNSTATS command is best run using a script. The following example shows such a script. It assumes that you are logged on as the user bpeuser with the password password, and you are connected to the Business Process Choreographer database, BPEDB. The DB2 commands generate a Windows command file that updates the statistics for all of the tables in the relevant table spaces in the BPEDB database. The TEMPLATE table space tables are omitted because the information is not accessed or updated frequently.

```
db2 -x "select 'db2 runstats on table '
       concat rtrim(tabschema)
       concat '.'
       concat tabname
       concat ' with distribution and detailed indexes all '
from syscat.tables
where
       type='T'  AND
        TBSPACEID IN (select TBSPACEID from sysibm.systablespaces
              where TBSPACE IN ('INSTANCE', 'WORKITEM', 'STAFFQUERY',
                  'AUDITLOG', 'SCHEDTS'))" > runStatsScript.cmd

echo db2 connect reset >> runStatsScript.cmd
```

The resulting command file updates the statistics for the specified tables. It contains entries similar to the following:

```
db2 runstats on table BPEUSER.ACTIVITY_INSTANCE_B_T with distribution and
 detailed indexes all
db2 runstats on table BPEUSER.AUDIT_LOG_T with distribution and
 detailed indexes all
...
db2 connect reset
```

You might want to extend the command file to run the REORG command before calling the RUNSTATS command. Refer to the DB2 documentation for information on how to reorganize your database tables using the REORG command.

6. Avoid deadlocks.

   Deadlocks happen when at least two transactions block each other's resource access. Deadlocks can result from poor database configuration. They can also result from the way in which the Business Process Choreographer APIs are used. To avoid deadlocks, each API call or query of the objects in the database should run in its own transaction.

   Business Flow Manager can recover from a database deadlock. However, there might be a major performance impact due to the time between when the deadlock is detected and the rolled-back transactions are retried. Therefore deadlocks should be avoided for performance reasons.

   To check for deadlocks, examine the db2diag.log file and use the DB2 monitors.

   a. Increase the log level for the db2diag.log file to get more information about bottlenecks within the database.

      Increase the value of the DIAGLEVEL parameter from 3 (default) to 4 to include errors, warnings, and informational messages. You can change the value using the following command:

      ```
      db2 update dbm cfg using DIAGLEVEL 4
      ```

   b. Create a DB2 event monitor.

      Event monitors provide more information about certain events, for example, deadlocks.

1) Create an event monitor using the following command:

```
db2 create event monitor monitor_name for statements, connections,
    transactions, deadlocks with details write to file file_name
```

2) Start the event monitor using the following command:

```
db2 set event monitor monitor_name state=1
```

3) Collect information using the following command:

```
db2evmon -db database_name -evm monitor_name output_file_name
```

c. Use the database snapshot monitors to gather statistics.

The snapshot monitors use the database monitor switches. For the database instance, the monitor switches have the following default settings:

Buffer pool (DFT_MON_BUFPOOL) = ON

Lock (DFT_MON_LOCK) = ON

Sort (DFT_MON_SORT) = OFF

Statement (DFT_MONSTMT) = OFF

Table (DFT_MON_TABLE) = OFF

Timestamp (DFT_MON_TABLE) = ON

Unit of work (DFT_MON_Table) = OFF

To see your current settings for the database instance, use the following command, and search for all of the parameters starting with DFT_MON_:

```
db2 get dbm cfg
```

These settings are different from those of the database. For the database, the monitor recording switches have the following default settings:

Switch list for db partition number 0

Buffer Pool Activity Information (BUFFERPOOL) = ON

Lock Information (LOCK) = ON

Sorting Information (SORT) = OFF SQL

Statement Information (STATEMENT) = OFF

Table Activity Information (TABLE) = OFF

Take Timestamp Information (TIMESTAMP) = ON

Unit of Work Information (UOW) = OFF

To see your current setting for the database, use the following command:

```
db2 get monitor switches
```

- To update the settings for one of the database monitors, for example, the lock monitor, use the following command:

```
db2 update monitor switches using lock on
```

This setting is valid only for the current database session.

- To update the settings for one of monitor switches for the database instance, for example, the lock monitor, use the following command:

```
db2 update dbm cfg using DFT_MON_LOCK OFF
```

To activate the setting, restart the database instance.

- Before you enable a snapshot monitor, reset the counters using the following command:

```
db2 reset monitor all
```

- To get a current snapshot after you restart a database instance, use the following command:

```
db2 get snapshot for all on database_name output_file_name
```

**Results**

Your long-running processes are running as fast as possible under the current
environment and loading conditions.

# Fine-tuning the messaging provider

Use this task to improve the performance of your messaging provider.

**About this task**

**Procedure**

If you use the service integration capabilities of WebSphere Application Server,
refer to tuning and problem solving for messaging engine data stores in the
WebSphere Application Server information center.

**Results**

The performance of your messaging provider is improved.

# Tuning microflows

Use this task to improve the performance of microflows.

**About this task**

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.

**Procedure**
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 600.
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. 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.
4. 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 **Servers** → **Application Servers**
   → *server* → **Add properties** → **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 people queries, that is, ones that use replacement 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 takes 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. If you define a volatile flag for the instance tables in the Business Process Choreographer database, this helps the DB2 optimizer to decide on an appropriate data access plan. This flag specifies that the index is always used instead of a table scan, even with empty or almost empty tables.

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.

## Optimize task and process queries

The query and queryAll API calls for retrieving task and process lists can result in complex SQL queries that include combinations of multiple database tables. An optimized representation of the data helps to address performance requirements, particularly for human workflow applications where multiple users access task lists concurrently.

**About this task**

If Business Process Choreographer is tuned for queries, response times usually are in the region of subseconds on an adequately sized system, even under high load. You can apply standard database calculations to calculate the response time of queries.

High-volume human workflow scenarios are best tuned with query tables. Query tables provide a precalculated set of data that is relevant for specific queries. For example, query properties must be joined by the database with tasks or process instances when the query runs. If query tables are used, these SQL joins do not need to be calculated anymore at query execution time.

The implementation and maintenance effort for query tables is higher than for standard database tuning techniques. Carefully consider standard database optimization techniques, such as indexes, log file distribution, and memory, before you use query tables.

Two approaches to query tables are supported: materialized views and custom tables. Decide whether to use materialized views or custom tables based on maintenance costs, development costs, and your requirements on the currency of the data that is returned by task and process list queries:

- Use materialized views to take advantage of the asynchronous update mechanism, which provides optimal query and process navigation performance.
  - Updates occur only when the materialized view is used
  - Setup, use, and maintenance is relatively simple
  - Can be implemented without changes to the application source code
- Use custom tables to include data from other applications in standard queries using the query or the queryAll interface. Additionally, custom tables can be used to provide an optimized representation of the data that is needed for task and process queries
  - Database triggers or other techniques can be used to synchronously update a custom table which is optimized for task and process list queries
  - Queries must be changed to query the data provided in the custom table

  **Related information**

  ⮕ Business Process Choreographer query() and queryAll methods: best practices

  ⮕ Tuning human workflows

  ⮕ DB2 information center: materialized query tables

# Chapter 22. Tuning Business Process Choreographer Explorer

The following suggestions provide various ways to improve the performance of the Business Process Choreographer Explorer.

**Procedure**

1. Consider increasing the maximum heap size of the server.

   Web clients naturally increase the load on your system. The more clients that are connected to your server, the more objects that have to be kept in memory. Therefore consider increasing the maximum heap size of your server. This improves the response time of your application, and increases the maximum number of users that can work in parallel with the application.

2. Tune the Web container thread pool.

   The size of the thread pool and the thread inactivity timeout can affect the performance of the Web container. You can change these settings using the administrative console (**Servers** ⇨ **Application Servers** ⇨ *server_name* ⇨ **Thread Pools** ⇨ **WebContainer**).

   a. Adjust the maximum and minimum pool size.

      All HTTP requests for Web client applications are processed using threads from the Web container thread pool. You can adjust the minimum and maximum pool size to influence the performance of your Web client.

      The number of maximum threads in the pool does not represent the number of requests your application server can process concurrently. If all of the threads in the pool are in use, additional requests are queued until they can be assigned to a thread. If a client request waits for a thread to be assigned, the response time increases for the client. However, if the maximum number is set too high, the system might get overloaded resulting in an even worse response time for the clients. It might also cause other applications to slow down dramatically.

      To determine whether changing the container size might result in a performance gain, you can use Tivoli® Performance Viewer to monitor the load on the threads (PercentMaxed counter) and the number of active threads (ActiveThreads counter) for the Web container module. If the value of the PercentMaxed counter is consistently in double digits, then the Web container might be a bottleneck. In this case, increase the number of threads. If the number of active threads is lower than the number of threads in the pool, decreasing the thread pool size might result in a performance gain.

   b. Adjust the thread inactivity timeout.

      The thread inactivity timeout defines after how many milliseconds of inactivity that should elapse before a thread is reclaimed. Changing this value might also impact your response times. A value of 0 indicates no wait time.

3. Decrease the threshold for large lists.

   If you are working with large task or process lists, you might want to decrease the search limit for lists to avoid gathering data that is not accessed by users.

   **Related tasks**

"Using the administrative console to configure the Business Process Choreographer Explorer" on page 208
You can use the administrative console to configure Business Process Choreographer Explorer.

**Related reference**

"Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer" on page 208
This script file configures Business Process Choreographer Explorer and all the necessary resources on a server or cluster.

# Chapter 23. Tuning Business Process Choreographer Observer

The time required to generate a report can vary, and depends on many factors. The following suggestions provide various ways to improve the performance of Business Process Choreographer Observer.

## Choose the appropriate database management system

The Business Process Choreographer Observer can use a DB2, Oracle, or Derby database. A Derby database is ideal for development, demonstrations, and prototyping where performance is not a primary consideration, and there is not a lot of data. For a production system, however, use either a DB2 or an Oracle database because they are much faster for large quantities of data.

## Update your database statistics

For DB2 and Oracle databases, updating the database statistics when you have a populated production database can improve the performance dramatically.

- To update the statistics for a DB2 database, enter the following commands:

```
RUNSTATS ON TABLE schema_prefix.EVENT_ACT_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.EVENT_PRC_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_ACT_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_PRC_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.OPEN_EVENTS_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.QUERY_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.SLICES_T FOR INDEXES ALL;
```

- To update the statistics for an Oracle database, enter the following commands:

```
ANALYZE TABLE schema_prefix.EVENT_ACT_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.EVENT_PRC_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.INST_ACT_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.INST_PRC_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.OPEN_EVENTS_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.QUERY_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.SLICES_T COMPUTE STATISTICS;
```

Where *schema_prefix* is the name of the database schema that was used when the Business Process Choreographer Observer database was created. For more information about updating the database statistics, refer to the documentation for your database.

## Emit relevant events only

In WebSphere Integration Developer, you can define the logging of activities or processes at a very detailed level. Activity audit events are only recognized by Business Process Choreographer Observer if events are also generated for the process that contains the activity. Activity events that cannot be associated with a process are ignored by the event collector application, and they are not stored in the database. To reduce the number of emitted events, perform the following steps:

1. Select the process templates that you want to audit, and disable the emission of events for processes that you are not interested in.
2. Select the activities of this process template that you want to audit. Check whether you can omit some of the events without impacting your report results.

For Business Process Choreographer Observer to provide an accurate picture of an activity or a process, you should either audit all or none of the event types.

### Choose a user-defined functions (UDFs) implementation

To create the reports, Business Process Choreographer Observer requires some specific UDFs to be installed in the database. The UDFs are provided as an SQL-based implementation and a Java-based implementation. The SQL implementation performs faster than the Java implementation, but has some disadvantages. If you are using the Java implementation, consider switching to the SQL implementation.

### Use a separate database

If Business Process Choreographer Observer uses the same database as Business Process Choreographer, they will have a negative impact on each other's performance. You will get better performance if Business Process Choreographer Observer has a separate database. Also consider hosting the Observer database on a separate database server.

### Use a separate machine

If Business Process Choreographer Observer is installed on a machine that hosts other applications, such as the BPEContainer or TaskContainer applications, consider running Business Process Choreographer Observer on a separate machine that has enough resources to meet your performance expectations.

### Increase timeout values

It can take a long time to generate a report. If it takes too long, a transaction timeout or a connection timeout of the JDBC driver might occur. If this happens, increase the timeout values as follows:

1. In the administrative console, navigate to **Servers** → **Application servers** → *server_name* → **Transaction Service**.
2. If the **Total transaction lifetime timeout** value is less than the **Maximum transaction timeout** value, make it the same.
3. If you are still experiencing performance problems, set the **Total transaction lifetime timeout** value to 0 and increase the **Maximum transaction timeout** value.
4. If you are still experiencing performance problems, set both the **Total transaction lifetime timeout** and the **Maximum transaction timeout** values to 0, and increase the value of the connection timeout for the JDBC driver. To do this, navigate to the connection pool properties for your data source. Click under **JDBC** → **JDBC providers** > *JDBC provider* → **Data sources** → *data_source_name* → **Connection pool properties**, and increase the **Connection timeout** value.

In a server cluster, you must adjust the transaction timeout values for all of the cluster members.

### Delete unnecessary data

The report performance depends on the amount of instance and event data in the observer database. Performance is reduced if reports query large amounts of data. Your report performance can improve if you reduce the number of process and

activity instances that are in the Business Process Choreographer Observer database. Regularly deleting unnecessary or old information can help to improve performance.

**Related tasks**

"Selecting between Java and SQL user-defined functions" on page 240
You can either use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the database for the Business Process Choreographer Observer.

"Deleting data from the observer database" on page 300
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. Deleting unnecessary data can improve the performance generating reports.

# Part 7. Troubleshooting

# Chapter 24. Troubleshooting the Business Process Choreographer configuration

Use this topic to solve problems relating to the configuration of Business Process Choreographer and its Business Flow Manager, or Human Task Manager components.

**About this task**

The purpose of this section is to aid you in understanding why the configuration of Business Flow Manager or Human Task Manager 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 configuration.

> **Related information**
>
> ▶ Troubleshooting Guide for WebSphere Process Server

## Troubleshooting the Business Process Choreographer database and data source

Use this task to solve problems with the Business Process Choreographer database and data source.

**About this task**

Both Business Flow Manager and Human Task Manager 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  sqldEscalateLocks Probe:4   Database:BPEDB

ADM5503E  The escalation of "10" locks on table "GRAALFS .ACTIVITY_INSTANCE_T"
          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 the class path definition for your JDBC provider is correct.

  - Check that the component-managed authentication alias is set to BPCDB_*nodeName.serverName_Auth_Alias* if Business Process Choreographer is configured on a server, and BPCDB_*clusterName_Auth_Alias* if Business Process Choreographer is configured on a cluster.

• If you are using Derby:

  - 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 ij command line processor, make sure that you have set up the Java environment, and that your classpath environment variable includes the following JAR files:

    - derby.jar

    - derbytools.jar

  - If you are using the embedded Derby driver, and you cannot connect to your Derby database using the Derby tools (like ij), 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 Derby may have already booted the database
  c:\WebSphere\AppServer\profiles\profile_name\databases\BPEDB.
```

ensure that only one application accesses the Derby database at a time.

• If you have written a client that uses Business Process Choreographer APIs without first authenticating the user, you should modify the client to perform a login before using the APIs. After migration, the J2EE roles BPEAPIUser and TaskAPIUser are set to the value Everyone, which maintains backward compatibility by maintaining the 6.0.x behavior of not requiring a login when application security is enabled. After you have fixed your client, you must change these roles to the value AllAuthenticated to prevent unauthenticated users accessing the APIs. For new installations these roles default to the value AllAuthenticated.

• If you get a database error when installing an enterprise application that contains a business process or human task, make sure that the database system used by the business process container is running and accessible. When an enterprise application is installed, any process templates and task templates are written into the Business Process Choreographer database.

- If you have problems using national characters. Make sure that your database was created with support for Unicode character sets.
- If tables and views cannot be found in the database and the create schema option is not enabled, check the following:
  - If a database schema qualifier is configured, check the following:
    - The schema qualifier must match the schema in the database. It must be the same schema as used in the scripts.
    - The user must be granted the privileges to work with the database tables and views.
  - If no schema qualifier is configured, ensure that:
    - The authentication alias of the user must be the same user ID as the one that is used to run the scripts, or must match the schema qualifier that is used in the scripts.
    - The user must be granted the privileges to work with the database tables and views.
- If the create schema option is enabled, and the database table and views cannot be found, the database tables and objects will be created automatically using the following terms:
  - If a schema qualifier is configured, the tables and views will be created using the schema qualifier.
  - If no schema qualifier is configured, the tables and views will be created using the user ID.

# 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 of the profile tool. You can find more detailed traces in the createBPCObjects.traceout file in the same directory. On Windows systems, these files are in the *install_root*/logs/manageprofiles/*profileName*/logs directory, and on Linux, UNIX, and i5/OS systems, they are in the *install_root*\logs\manageprofiles\*profileName*\logs.

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 profile logs directory. This directory is in the *profile_root* directory.

## Administrative scripts

All of the Business Process Choreographer scripts that are run using wsadmin are logged in the wsadmin.traceout file in the logs directory of the profile tool. 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, bpeupgrade.jacl, clientconfig.jacl, and bpeunconfig.jacl write their log files in the logs directory with the names bpeconfig.log, bpeupgrade.log, clientconfig.log, and bpeunconfig.log.

The following configuration scripts write their log files in the logs directory to the files setupOberver.log and setupEventCollector.log, respectively.

- setUpEventCollector.bat and setupObserver.bat (Windows systems)
- setUpEventCollector.sh and setupObserver.sh (Linux and UNIX systems)
- setUpEventCollector and setupObserver (i5/OS systems)

Also check the wsadmin.traceout file.

### Administrative utility scripts

The administrative scripts in the admin subdirectory of the ProcessChoreographer directory do not write their own log files. Check the wsadmin.traceout file and the application server log files.

# 6.0.x Business Process Choreographer API client fails in a 6.1 environment

You did not migrate your 6.0.x Business Process Choreographer API client when you upgraded to WebSphere Process Server Version 6.1. When you try to run your client in the 6.1 environment, the client fails.

### Symptom

Exceptions similar to the following are written to the SystemOut.log file:

```
[9/6/07 21:05:27:093 PDT] 00000045 ExceptionUtil E CNTR0020E: EJB threw an unexpected
(non-declared) exception during invocation of method "processMessage" on
bean "BeanId(validateDataApp#validateDataEJB.jar#component.validateItem, null)".
Exception data: javax.ejb.AccessLocalException: ;
nested exception is: com.ibm.websphere.csi.CSIAccessException:
SECJ0053E: Authorization failed for /UNAUTHENTICATED while invoking
(Home)com/ibm/bpe/api/BusinessFlowManagerHome create:4
securityName: /UNAUTHENTICATED;accessID: UNAUTHENTICATED is not granted any of the required
roles: BPEAPIUser
com.ibm.websphere.csi.CSIAccessException: SECJ0053E: Authorization failed for
/UNAUTHENTICATED while invoking (Home)com/ibm/bpe/api/BusinessFlowManagerHome
create:4 securityName: /UNAUTHENTICATED;accessID: UNAUTHENTICATED is not granted any of the required
roles: BPEAPIUser
at com.ibm.ws.security.core.SecurityCollaborator.performAuthorization(SecurityCollaborator.java:484)
at com.ibm.ws.security.core.EJSSecurityCollaborator.preInvoke(EJSSecurityCollaborator.java:218)
at com.ibm.ejs.container.EJSContainer.preInvokeForStatelessSessionCreate(EJSContainer.java:3646)
at com.ibm.ejs.container.EJSContainer.preInvoke(EJSContainer.java:2868)
at com.ibm.bpe.api.EJSLocalStatelessGenericBusinessFlowManagerEJBHome_a412961d.create(Unknown Source)
```

### Reason

If your Business Process Choreographer API client does not include user authentication, it relies on a security hole. In WebSphere Process Server Version 6.1 this security hole has been fixed, which causes the client to fail.

### Resolution

Modify your API client to force user to log on to the client before they can use the APIs.

As a temporary workaround, you can change the mappings for the BPEAPIUser and the TaskAPIUser roles. To change the mapping:

1. In the administrative console, click **Applications** → **Enterprise Applications** → **BPEContainer_suffix**, and under **Detail Properties** click **Security role to user/group mapping**

2. Change the BPEAPIUser role from `All authenticated` to `Everyone`, and click **OK**.

3. Repeat step 2 for the TaskContainer_suffix and the TaskAPIUser role.

4. After you have modify your client, you must change these roles back to `All authenticated` to prevent unauthenticated users accessing the APIs.

# 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:

```
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 people directory providers, are traced by `com.ibm.ws.staffsupport`.

## What to send support

After enabling tracing, recreate your problem scenario then provide the following files:

- The WebSphere Application Server FFDC log, located in the ffdc folder
- The following log files:
  - SystemOut.log
  - SystemErr.log
  - trace.log

On Linux, UNIX, and i5/OS systems, these files are located in the *profile_root*/logs/*server_name* directory. On Windows platforms they are located in the *profile_root*\logs\*server_name* directory.

If your problem scenario causes a lot of logging, backup files for the logs might be created with names, such as SystemOut_07.10.01_11.00.51.log. You can use the administrative console to change the number of backup files that are created and the size of the log files. It might be good to increase both of these values to ensure that you capture all of the data.

**Related information**

Troubleshooting Guide for WebSphere Process Server

# Chapter 25. Troubleshooting business processes and human tasks

Use this topic to solve problems relating to business processes and human tasks.

**About this task**

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.

**About this task**

In Business Process Choreographer Explorer, you can search for error message codes on the IBM technical support pages.

**Procedure**

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**.

**What to do next**

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 that need these classes may still be executing.

## 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.

# Unexpected exception during invocation of the processMessage method (message: CNTR0020E)

The business process container has stopped and the client could not connect to the server.

## Resolution

Verify that the business process container is running.

# 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 no

## 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.

   "Managing the life cycle of a business process" on page 429
   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.

   "Repairing activities" on page 441
   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.

# 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. Perform Querying and replaying failed messages, using the administrative console.
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

Because the subprocess invocation leads to a remote EJB method call, 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.JMSException: 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.

# The failed state of the process instance does not allow the requested sendMessage action to be performed (Message: CWWBE0126E)

An attempt to send an event to a process instance results in an 'EngineProcessWrongStateException' message.

### Reason

A common reason for this error is that a message is sent to a receive or pick activity to create a new process instance, but a new process instance cannot be instantiated. This situation occurs if the values that are passed with the message for the correlation set that is used by this activity match an existing process instance, which is already in the failed state.

### Resolution

To correct this problem you must either delete the existing process instance, or pass a correlation set value that does not match 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 Derby or Oracle 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.

**About this task**

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.

**Procedure**

1. Start the WebSphere log analyzer.

   Run one of the following scripts:
   - On Windows systems, *install_root*/bin/waslogbr.bat
   - On Linux, UNIX, and i5/OS systems, *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
      - *profile_root*\profiles\*profile_name*\logs\activity.log file on Windows systems
      - *profile_root*/profiles/*profile_name*/logs/activity.log file on Linux, UNIX, and i5/OS systems

   b. Click **Open**.

# Troubleshooting the administration of business processes and human tasks

This article describes how to solve some common problems with business processes and human tasks.

**About this task**

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 safely. 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.

- A long-running business process that is started by an invocation task fails to start. A JSP snippet makes the invocation task available to users. In the following example, the synchronous calling pattern createAndCallTask is used. In this case, the long-running business process fails to start:

```
HumanTaskManager htm = ...
TaskTemplate taskTemplate = htm.getTaskTemplate( "start the process" );
Task task = htm.createAndCallTask( taskTemplate.getTKTID() );
```

```
while (task.getState() != TASK.TASK_STATE_FINISHED)
{
    Sleep(100);
}
```

A long-running process consists of several transactions and its invocation style is asynchronous. Therefore it must be started using the asynchronous calling pattern, createAndStartTask.

```
HumanTaskManager htm = ...
TaskTemplate taskTemplate = htm.getTaskTemplate( "start the process" );
Task task = htm.createAndStartTask( taskTemplate.getTKTID() );
while (task.getState() != TASK.TASK_STATE_FINISHED)
{
    Sleep(100);
}
```

In addition, transaction attribute in the JSP deployment descriptor must be set to NotSupported. This ensures that the code snippet is executed without a transaction, and the createAndStartTask method opens a new transaction to start the process instance. This transaction is committed when the createAndStartTask method returns, and the message is visible.

It is a good practice to prepare the "while" loop for states other than finished. For example, if during the execution of the process an activity fails, the end state might be TASK.TASK_STATE_FAILED.

# Troubleshooting escalation e-mails

Use this information to solve problems relating to escalation e-mails.

**About this task**

Escalations are triggered when human tasks do not progress as expected. The escalation creates work items. It can also send e-mails to the users that are affected by the escalation. If you are having problems with escalation e-mails, use the information here to help you to solve the problems.

- Check the SystemOut.log file for error messages relating to people assignments or e-mail addresses.
- If the SystemOut.log file does not contain any relevant messages, enable the debug mode for the mail session server.

  In the administrative console, click **Resources** → **Mail** → **Mail sessions** → *HTMMailSession_server*, and select the **Enable debug mode** check box. When an escalation e-mail is sent, debug information is written to the SystemOut.log file.
- If you are using virtual member manager as the people directory provider and you are having problems with e-mail addresses, enable the Staff.Diagnosis custom property.

  1. In the administrative console, click **Applications servers** → *server_name*.
  2. Under **Business Integration**, expand **Business Process Choreographer**, and click **Human Task Manager Configuration**.
  3. In the **Configuration** tab, under **Additional Properties**, click **Custom Properties** → **Staff.Diagnosis**, and type on in the **Value** field.

  When an escalation e-mail is sent, additional information about the people assignment is written to the SystemOut.log file.
- Check if the Human Task Manager hold queue contains messages.

  1. In the administrative console, click **Applications servers** → *server_name*.

2. Under **Business Integration**, expand **Business Process Choreographer**, and click **Human Task Manager Configuration**.

3. In the **Runtime** tab, click **Replay Hold Queue**. The messages in the hold queue are shown in the **Hold queue messages** field.

If the hold queue contains messages, check the First Failure Data Capture (FFDC) directory of your server for more information about the error.

- Check the values of the custom properties for the number of times an e-mail is resent and the timeout for sending an e-mail.

  1. In the administrative console, click **Applications servers** → *server_name*.

  2. Under **Business Integration**, expand **Business Process Choreographer**, and click **Human Task Manager Configuration**.

  3. In the **Configuration** tab, under **Additional Properties**, click **Custom Properties**.

  4. Check the values of the **EscalationEmail.RetryTimeout** and the **EscalationEmail.MaxRetries** fields.

     **EscalationEmail.RetryTimeout**
     > Specifies how long Human Task Manager waits until it resends an e-mail notification that failed. The default value for this field is 3600 s. (one hour) If the retry fails, then the retry timeout is doubled dynamically for every time the retry fails. By default, if the first retry fails, another retry is made after two hours.

     **EscalationEmail.MaxRetries**
     > Specifies the number of times Human Task Manager tries to resend an e-mail notification that failed. The default value for this field is 4 retries. If the value of this field is set to 0, a failed e-mail notification is not resent. If all of the retries fail, then a message is put into the hold queue. You can see the messages in the hold queue in the administrative console in the **Runtime** tab for Human Task Manager. If you replay the messages, this is equivalent to sending the e-mail for the first time.

# Troubleshooting people assignment

Use the following information to help solve problems relating to the assignment of people to authorization roles.

**About this task**

This information covers the following problems:
- Errors during the deployment of the people directory provider
- Entries in the people directory are not reflected in work item assignments
- Changes to the people directory are not immediately reflected in work-item assignments
- Unexpected people assignments for tasks or process instances
- Stopped human tasks
- Error and warning messages relating to people assignment
- Issues with group work items and the "Group" people assignment criteria
- Cleanup of stored people assignment results

You can also search for additional information in the Technical support search page.

**Errors during the deployment of the people directory provider**

> If you are using the Lightweight Directory Access Protocol (LDAP) people directory provider, deployment might fail due to incorrect values of the provider 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 people directory are not reflected in work item assignments**

> The maximum number of user IDs retrieved by a people 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 people directory provider is, for example, `LDAPTransformation.xsl`. This file is in the *install-root*/ProcessChoreographer/Staff directory on Linux, UNIX, and i5/OS platforms, and in the *install-root*\ProcessChoreographer\Staff directory on Windows platforms. The default Threshold value is 20. To change this value:
>
> 1. Create a new people directory 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.

**Changes to the people directory are not immediately reflected in work-item assignments**

> Business Process Choreographer caches the results of people assignments evaluated against a people directory, such as an LDAP server, in the runtime database. When changes occur in the people directory, these are not immediately reflected in the database cache.
>
> The *Administration guide* describes three ways to refresh this cache:
>
> - **Refreshing people query results, using the administrative console**. Use this method if you have major changes and need to refresh the results for almost all people queries.
> - **Refreshing people 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 people query results.
> - **Refreshing people query results, using the refresh daemon**. Use this method to set up a regular and automatic refresh of all expired people 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. The group membership list of the process starter ID that is used for process navigation, is never refreshed.

**Unexpected people assignments for tasks or process instances**

> Default people assignments are performed if you do not define people assignment criteria for certain roles for your tasks, or if people assignment

fails or returns no result. These defaults might result in unexpected user authorization; for example, a process starter might receive 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 91. 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 people assignment fails or does not return proper results ... |
|---|---|---|
| Process administrator | Process starter becomes process administrator | The following exception occurs and the process is not started:<br><br>EngineAdministratorCannotBe ResolvedException |
| Process reader | No reader | No reader |

*Table 92. 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 people assignment 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 |
| 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 93. 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 people assignment fails or does not return correct results ... |
|---|---|---|
| Task administrator | Originator becomes administrator | 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 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 using the Business Flow Manager API, members of the BPESystemAdministrator role have administrator authorization, and members of the BPESystemMonitor role have reader authorization.

**Note:** When a method is invoked using the Human Task Manager API, members of the TaskSystemAdministrator role have administrator authorization, and members of the TaskSystemMonitor role have reader authorization.

**Stopped human tasks**

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...`

These problems indicate 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.

**Error and warning messages relating to people assignment**

Some common errors can occur when accessing a people directory during people assignment. 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 people directory 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 people assignment 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 people assignment criteria (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 people assignment criteria (verb) parameters and the LDAP directory content for mismatches in the task model. Also check the XSLT file of your people assignment configuration for errors.

**Issues with group work items and the "Group" people assignment criteria**

If you are using the Group people assignment criteria, the following 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 people assignment criteria. Note that the group membership list of the process starter is never refreshed.

**Cleanup of stored people assignment results**

People assignment results are stored in the database. All stored people assignment results are subject to people assignment refreshes. If the task template that contains the task instance that leads to the computation of a people assignment result is deleted, the stored people assignment result is deleted as well. However, the stored people assignment results are not deleted if only the task instances that are using the stored people assignment results are deleted.

To avoid large numbers of stored and unnecessary people assignment results in the database, take the following steps in the context of a task template:

1. Assess whether your people assignment criteria definitions lead to shared or unshared people assignment results.

2. If unshared assignment results occur, consider putting a cleanup procedure in place for people assignment results. Base the cleanup interval on the expected number of task instances, and the unshared people assignment results per cleanup interval. For more information on how to apply a script-based cleanup procedure, refer to Removing unused people query results, using administrative commands.

# Troubleshooting Business Process Choreographer Explorer

Use this to solve problems relating to the Business Process Choreographer Explorer.

**About this task**

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 an error message instead of the login page, 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 633.

- If you can log onto Business Process Choreographer Explorer, but some items are not displayed, or if certain actions are not enabled, this indicates a problem with your authorization.

  Possible solutions to this problem include:
  - Use the administrative console to ensure that WebSphere administrative security is enabled.
  - 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 are not visible 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 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 626
  This describes the solutions to common problems with business process execution.

# Troubleshooting Business Process Choreographer Observer

Refer to the information in this topic if you are experiencing difficulty with Business Process Choreographer Observer.

## Symptoms

No events are displayed on the Welcome page of Business Process Choreographer Observer.

## Reasons and resolutions

The Business Process Choreographer Observer database does not contain any events, or the events are not transformed yet.

| Reason | Resolution |
|--------|-----------|
| CEI logging is not enabled. | Make sure that CEI logging is enabled for the business process container. Refer to Enabling logging for the Business Process Choreographer Observer to enable CEI logging. |
| The Common Event Infrastructure event server or the Business Process Choreographer event collector are not running. | Use the administrative console to check that the Common Event Infrastructure event server and the Business Process Choreographer event collector are running. |

| Reason | Resolution |
|---|---|
| Event monitoring for your business processes is disabled. | Make sure that event monitoring is enabled in the definitions of your process model in WebSphere Integration Developer. Refer to the WebSphere Integration Developer information center for recommendations on how to enable event monitoring for business processes. |
| The event transformer is not triggered. | Restart the Business Process Choreographer event collector to trigger the event transformer. |
| Inappropriate configuration settings of Business Process Choreographer event collector prevent data from being visible in the observer. | Call the setupEventCollector configuration script to change the Business Process Choreographer event collector configuration settings for BPCEventTransformerEventCount, BPCEventTransformerMaxWaitTime, and BPCEventTransformerToleranceTime. Refer to Changing configuration parameters for changing the Business Process Choreographer event collector configuration settings. |

## If the problem continues

- Check the system log file SystemOut.log of the server for error messages.
- Check the deployment and configuration of Business Process Choreographer event collector and Business Process Choreographer Observer. To check the configuration settings, call the setupEventCollector and the setupObserver configuration script. For further information on how to change the Business Process Choreographer event collector configuration settings, refer to "Changing configuration parameters for the Business Process Choreographer Observer" on page 256.
- Enable the tracing facility for the observer components in the administrative console: **Logging and Tracing** → **server1** → **Diagnostic Trace Service** → **Change Log Detail Levels**. Set detail level all for com.ibm.bpe.observer.*, and restart the BPCECollector and the BPCObserver applications.

# Using process-related and task-related audit trail information

Explains the event types and database structures for business processes and human tasks.

**Before you begin**

Logging must be enabled for the business process container, the task container, or both.

**About this task**

If logging is enabled, for every navigation step of a business process or a human task, 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 94. 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 |
| PROCESS_EVENT_ESCALATED | 42049 |
| PROCESS_WORKITEM_TRANSFERRED | 42056 |
| PROCESS_PARTNER_CHANGED | 42058 |

*Table 95. 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 |

*Table 95. Activity events  (continued)*

| Audit event | Event code |
|---|---|
| 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 96. Events related to variables*

| Audit event | Event code |
|---|---|
| VARIABLE_UPDATED | 21090 |

*Table 97. Control link events*

| Audit event | Event code |
|---|---|
| LINK_EVALUATED_TO_TRUE | 21034 |
| LINK_EVALUATED_TO_FALSE | 42000 |

*Table 98. Process template events*

| Audit event | Event code |
|---|---|
| PROCESS_INSTALLED | 42006 |
| PROCESS_UNINSTALLED | 42007 |

*Table 99. Scope instance events*

| Audit event | Event code |
|---|---|
| SCOPE_STARTED | 42020 |

*Table 99. Scope instance events (continued)*

| Audit event | Event code |
|---|---|
| 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 100. 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 |
| 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 |

*Table 100. Task instance events (continued)*

| Audit event | Event code |
|---|---|
| TASK_WORKITEM_DELETED | 51201 |
| TASK_WORKITEM_CREATED | 51202 |
| TASK_WORKITEM_TRANSFERRED | 51204 |
| TASK_WORKITEM_REFRESHED | 51205 |

*Table 101. Task template events*

| Audit event | Event code |
|---|---|
| TASK_TEMPLATE_INSTALLED | 52001 |
| TASK_TEMPLATE_UNINSTALLED | 52002 |

*Table 102. 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 643.

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 103. 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 103. Structure of the AUDIT_LOG_B audit trail view (continued)*

| Name | PTE | PIE | AIE | VAR | CLE | SIE | Description |
|---|---|---|---|---|---|---|---|
| ACTIVITY_KIND | | | x | | | | Kind of the activity on which the event occurred. Possible values are:<br><br>KIND_EMPTY 3<br>KIND_INVOKE 21<br>KIND_RECEIVE 23<br>KIND_REPLY 24<br>KIND_THROW 25<br>KIND_TERMINATE 26<br>KIND_WAIT 27<br>KIND_COMPENSATE 29<br>KIND_SEQUENCE 30<br>KIND_SWITCH 32<br>KIND_WHILE 34<br>KIND_PICK 36<br>KIND_FLOW 38<br>KIND_SCRIPT 42<br>KIND_STAFF 43<br>KIND_ASSIGN 44<br>KIND_CUSTOM 45<br>KIND_RETHROW 46<br>KIND_FOR_EACH_SERIAL 47<br>KIND_FOR_EACH_PARALLEL 49<br><br>These are the constants defined for ActivityInstanceData.KIND_* |
| ACTIVITY_STATE | | | x | | | | State of the activity that is related to the event. Possible values are:<br><br>STATE_INACTIVE 1<br>STATE_READY 2<br>STATE_RUNNING 3<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<br><br>These are the constants defined for ActivityInstanceData.STATE_* |
| 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 103. 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:<br><br>PROCESS_FAILED<br>ACTIVITY_FAILED<br>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:<br>• Process instance work item deleted<br>• Activity instance work item deleted<br>• Process instance work item created<br>• Activity instance work item created |
| ADDITIONAL_ INFO | | x | x | | | x | The contents of this field depends on the type of the event:<br><br>**ACTIVITY_WORKITEM_ TRANSFERRED, PROCESS_WORK_ITEM_ TRANSFERRED**<br>The name of the user that received the work item.<br><br>**ACTIVITY_WORKITEM_ CREATED, ACTIVITY_WORKITEM_ REFRESHED, ACTIVITY_ESCALATED**<br>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).<br><br>**PROCESS_EVENT_RECEIVED, SCOPE_EVENT_RECEIVED**<br>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 theAUDIT_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 104. 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 645. |
| 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:<br><br>TASK_FAILED<br>TASK_FAULT_MESSAGE_UPDATED |

*Table 104. 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:<br><br>TASK_FAILED<br>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 |
| | | | x | ESCALATION_WORKITEM_TRANSFERRED |
| 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 |
| | | | x | ESCALATION_WORKITEM_DELETED |
| PARENT_CONTEXT_ID | x | | | 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. |
| 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:<br><br>KIND_HUMAN  101<br>KIND_ORIGINATING  103<br>KIND_PARTICIPATING  105<br>KIND_ADMINISTRATIVE  106 |

*Table 104. Structure of the TASK_AUDIT_LOG audit trail view (continued)*

| Name | TIE | TTE | EIE | Description |
|------|-----|-----|-----|-------------|
| TASK_STATE | x | | | The state of the task or task template. Possible values for task templates are:<br><br>STATE_STARTED 1<br>STATE_STOPPED 2<br><br>Possible values for task instances are:<br><br>'1' :'STATE_INACTIVE'<br>'2' :'STATE_READY'<br>'3' :'STATE_RUNNING'<br>'5' :'STATE_FINISHED'<br>'6' :'STATE_FAILED'<br>'7' :'STATE_TERMINATED'<br>'8' :'STATE_CLAIMED'<br>'12' :'STATE_EXPIRED'<br>'101':'FORWARDED' |
| 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 | The reason for the assignment of the work item. Possible values are:<br><br>POTENTIAL_OWNER 1<br>EDITOR  2<br>READER 3<br>OWNER  4<br>POTENTIAL_STARTER 5<br>STARTER  6<br>ADMINISTRATOR 7<br>POTENTIAL_SENDER 8<br>ORIGINATOR 9<br>ESCALATION_RECEIVER 10<br>POTENTIAL_INSTANCE_CREATOR 11<br><br>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. |

# Part 8. Appendixes

# 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.

this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

## Programming interface information

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

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

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

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

## Trademarks and service marks

IBM, the IBM logo, AIX, DB2, i5/OS, Informix, iSeries, MQSeries, WebSphere, and z/OS are registered trademarks and Cloudscape, DB2 Connect, and DB2 Universal Database are trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

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

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

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

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

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



IBM WebSphere Process Server for Multiplatforms, Version 6.1.0

**IBM** ®