IBM WebSphere Process Server for Multiplatforms

IBM

Business Process Choreographer

Version 7.0.0

30 April 2010

This edition applies to version 7, release 0, 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

Contents **v**

Contents **vii**

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

Business processes overview

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 versioning.
- Extensions for manually setting or controlling the transactional boundaries in a business process.
- Timeouts for activities.

Process templates

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

Process properties are specified when the process is defined. In the runtime environment, properties for process templates are stored in the runtime database. They can be accessed using the Business Process Choreographer database views, such as the PROCESS_TEMPLATE view, or using query tables.

In addition, 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
> When a process template is in the stopped state, no new instances of this template can be created and started. Existing instances of the template continue to run until they complete.

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 23
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.

"Transactional behavior of business processes" on page 25
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.

Process instances

A process instance is the instantiation 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 a fault occurs that is not handled by the process.

Many process instance properties are inherited from the corresponding process template. Others, such as the state of the process instance, are assigned and modified during the lifetime of the process instance. All of these properties are stored in the runtime database. They can be accessed using the Business Process Choreographer database views, such as the PROCESS_INSTANCE view or QUERY_PROPERTY view, or using query tables.

Process versioning overview

Business processes evolve over time. They need to reflect changing environments and business needs. These changes might be business-driven changes, such as changes in regulations, or optimizations of business processes. Business-process applications can include long-running instances. These instances can run for weeks, months, or even years. This characteristic imposes specific requirements on the introduction of new versions of business processes.

To create a new version of your process, you create a module in WebSphere Integration Developer. This module contains the new version of the process, based on a copy of the original process. Make the changes that you need to the new version of the process, give the new version a valid-from date, and then deploy this module to your runtime environment. If you want to enable the migration of existing process instances to the new version, you must also define a process migration specification in WebSphere Integration Developer, and deploy it together with the new version of the process.

Depending on the nature of the changes to the process, existing process instances might need to complete using the version they were created and started with, whilst new instances should be created from the new version of the business process. In some situations, for example, when you want to correct an error in the process, you might also want to migrate the running instances of a process to the newer version so that they complete using this version.

Related concepts

"Invocation scenarios for business processes" on page 22
A business process is a Service Component Architecture (SCA) component implementation. 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.

Process versioning: Invoking different versions of a business process

You can include versioning information, such as a valid-from date, when you define the business process in WebSphere Integration Developer. At runtime, you can dynamically invoke the version of the process that is currently valid, or invoke a specific version of the process.

The version of a process is determined by its name and 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 when a process instance is invoked 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, as long as these instances have not been migrated to a newer version. 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 can still be used for existing long-running process instances
- Process templates that become valid in the future according to their valid-from date

An example of late-binding is when a new process is invoked in Business Process Choreographer Explorer. The instance that is created is typically based on the currently valid version of the process, which has a valid-from date that is not in the future.

To apply late-binding when invoking a subprocess, 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.

Process versioning: Migrating running process instances to a new version of the business process

When you introduce a new version of a process, you might want this version to apply to both new process instances and to instances that have already started. This can be important in environments where changes to processes are needed frequently, but where an individual process instance can be relatively long-lived. In these cases you need to migrate the process instances that are running to the newer version.

The following figure shows the steps for defining a new version of the process in WebSphere Integration Developer through to migrating running instances in WebSphere Process Server.

Note: For a new version of a process to become a migration target, you must deploy a migration specification together with the new version of the process.
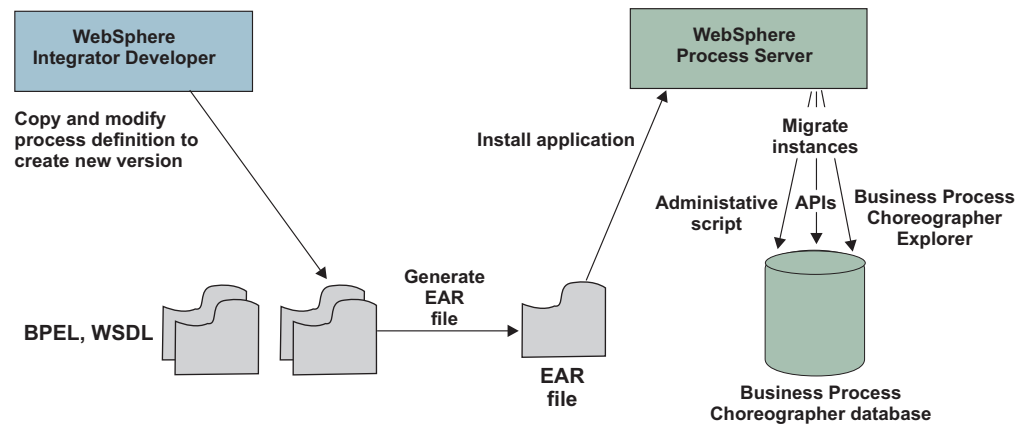
To migrate running process instances to a new version of the process, you can use either an administrative script to migrate process instances in bulk, or Business Process Choreographer Explorer to migrate specific instances. Migration of a process instance means that the process, the variables, and the activities that are at the current position of the process navigation now refer to the new version of the business process and the follow-on navigation depends on the logic of the new version of the business process. The activities that have already been navigated when the process instance is migrated are not migrated. During the migration of a process instance, all the instances of inline human tasks, which belong to this process instance and are not yet in an end state, are also migrated.

If the changes to the process that are contained in the new version of process do not affect the process logic, such as the display name or description of an activity, a running process instance can be migrated any time during the process navigation. However, if the changes affect the process logic, such as new activities, variables, or conditional expressions are changed, you can migrate a running process instance to a newer version only if all of the changes that affect the logic of the process are after the current position in the process navigation. For detailed information about the changes that can be made to a new version of a process, see this Technote.

Any change to an inline human task is considered to affect the logic of the process. You can migrate a running process instance to a newer version only if all of the changed inline human tasks are after the current position in the process navigation.

If events are defined for the business process, you can track process migration using CBEs. An event can be generated when the migration is started and again when the migration is complete. A history of process migrations is also kept in the Business Process Choreographer database.

Example

To work through an example of process instance migration, go to the Business Process Management Samples and Tutorials Web site.

Correlation sets

Correlation is used to track the multiple, long-running exchanges of messages that typically take place between a BPEL process and its partner services. Correlation sets help to route messages to the appropriate process instance based on the contents of the message body thus enabling the process instance to hold a conversation with the partner service.

A correlation set is made up of one or more properties that are defined in a WSDL file. A property alias is a rule that tells Business Process Choreographer how to map data from a message into a correlation property. You can use correlation sets in invoke, receive, pick, and reply activities to indicate which correlation sets occur in the messages that are sent and received. The values of each correlation set uniquely identify the process instance. This is true even if the process instance has already reached an end state, such as the finished state.

A correlation set is required if a process consists of more than one receive or pick activity. The receive or pick activity that initiates a new process instance does not necessarily need a correlation set. The remaining receive or pick activities, however, need a correlation set to uniquely identify the process instance to route the message to.

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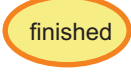
When a process is initiated, the navigation of a business process instance is started, and it begins to interact with its environment. This means that certain interactions are only possible in certain process states, and these interactions, in turn, influence the state of the process instance.

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. Microflows and long-running processes have different state transition diagrams.

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.
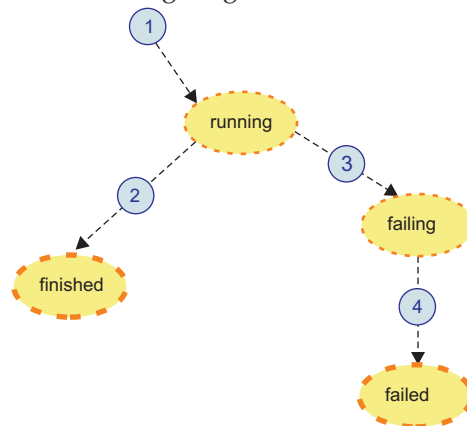
After normal initiation of the process instance, the first process state a process instance reaches is the running state (1). When a process instance runs normally through to completion, the process state changes from running to finished (2). If a fault reaches the process boundary, the process is put 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.
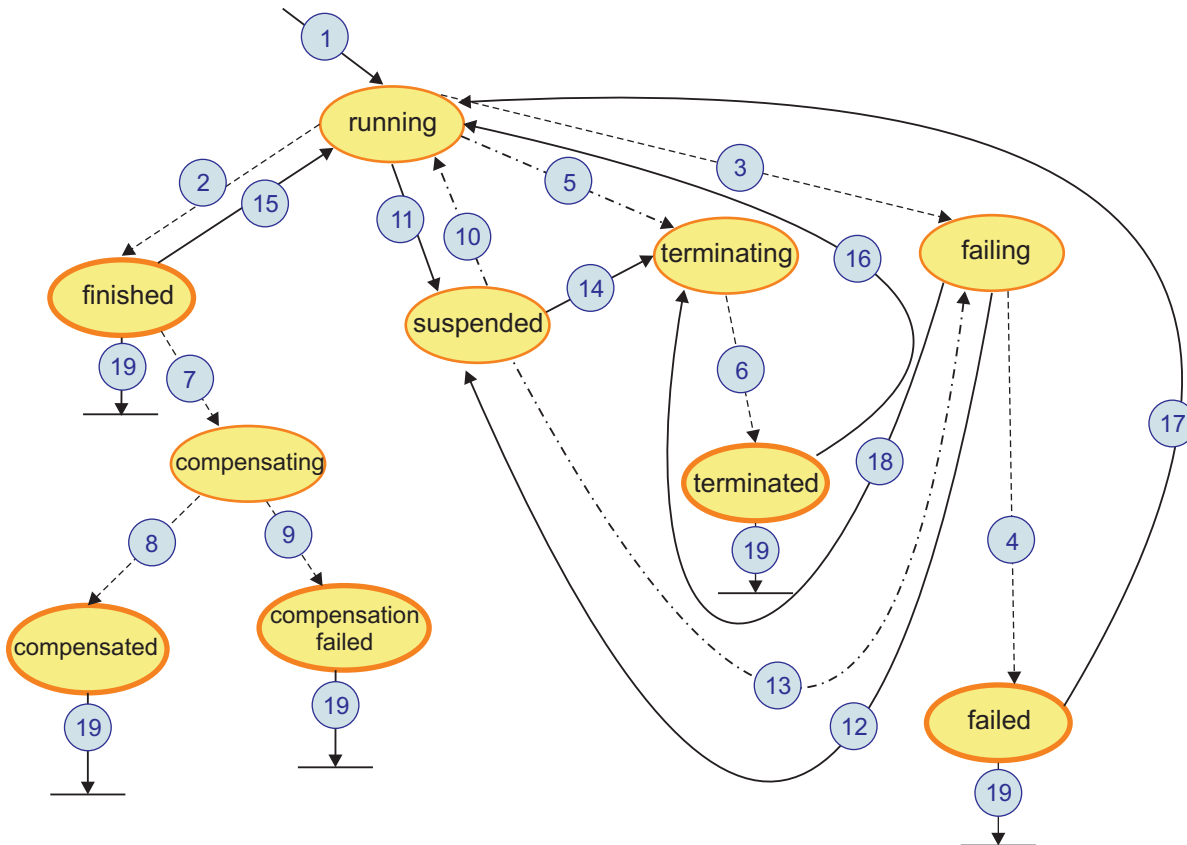
The running, finished, failing, and failed states, and the state transitions between them are the same as for microflows.

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 (19). 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.

Related concepts

"Transactional behavior of business processes" on page 25
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.

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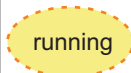
States and state transitions are important in the life cycle of basic activities. Basic activities are grouped into the following activity types. The state transitions diagrams vary according to the activity type:

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

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.

Symbol	Explanation
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, terminate, and Java snippet activities. It introduces the states: inactive, skipped, finished, failed, stopped, and terminated. These states are common to all of the basic activity types.
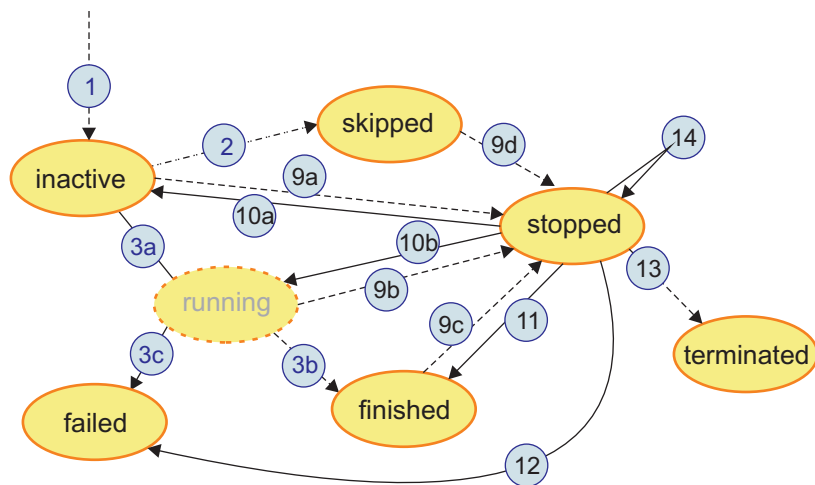
After an activity is created, it is in the inactive state (1). Activities that are enclosed in a flow can have multiple incoming links and a join condition. Before such an activity can start, all of the incoming links must be navigated. The **suppressJoinFailure** attribute of the activity and the outcome 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 subsequent behavior of the activity depends on whether it has an exit condition that is evaluated on entry of the activity.

– If the exit condition evaluates to `true`, the state of the activity changes to skipped (2), and the transition conditions of the links leaving the activity are evaluated.

– If the exit condition evaluates to `false` or if an exit condition is not specified, the activity is activated, and it state changes to running (3a). The activity implementation is run and when it successfully completes, the subsequent behavior of the activity depends on whether it has an exit condition that is evaluated on entry of the activity.

  - If such an exit condition is specified and it evaluates to `true` or if it is not specified, the state of the activity changes to finished (3b), and the transition conditions of the links leaving the activity are evaluated.

  - If the exit condition evaluates to `false` the activity state changes to stopped (9b).

  If the **Continue On Error** setting is set to `yes` and 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.

An activity instance can be skipped in any state, including the inactive state. If the activity is in the inactive state, the state changes from inactive to skipped (2) when the activity is reached by the navigation, regardless of the outcome of the join condition. The transition conditions of the links that leave the activity are also evaluated. If the activity is automatically skipped, the conditions are not evaluated.

Fault handling behavior when the Continue On Error setting for the process is set to `no`

  If the **Continue On Error** setting is set to `no`, a fault that is not caught by a fault link or an immediately enclosing fault handler causes the activity to be put into the stopped state (9a - 9d). The stopped state can be reached in the following situations:

  • The activation of the activity fails, for example, if an exception occurs during the evaluation of the join condition.

    The state of the activity changes from inactive to stopped (9a). The activity can be repaired by an administrator with the help of a forceRetry or forceJoinCondition API request. For a forceRetry API request, the state of the activity changes to inactive (10a) and the activation of the activity is tried again. If the retry is successful, the state changes to running (3a), and finally to finished (3b). If the retry is not successful, the activity is put into the stopped state again (14). For a forceJoinCondition API request, the state of the activity changes to inactive (10a) and then, depending on the condition value passed as an API parameter, the state changes to running (3a) or skipped (2).

    You can change the continue-on-error behavior with the API repair request. If this is done and the activation fails again, the activity ends in the inactive state (10a), and the fault is propagated to the fault handlers of the enclosing scope.

  • The implementation of the activity fails, for example, because the XPath expression in an assign statement causes an exception.

    The state of the activity changes from running to stopped (9b). Because the state change occurs in a single transaction, the running state is not visible.

The activity can be repaired by an administrator with the help of a forceRetry API request. The activity is put back into the running state (10b). The activity can also be repaired with a forceComplete API request. In this case, the activity is put into the finished state (11), and the navigation of the process continues.

If the activity is repaired, the stopped state can be reached again (14) if the implementation of the activity fails again during the repair step. If the continue-on-error behavior is changed with the API repair request and the implementation fails again, the activity ends in the failed state, and the fault is propagated to the fault handlers of the enclosing scope.

- The evaluation of the transition conditions on a link leaving the activity fails.

  The state of the activity was finished or skipped before the error occurred (9c or 9d). The activity can be repaired by an administrator with the help of a forceComplete API request. If the evaluation is then successful, the state is finished again (11). If the evaluation is not successful, the state of the activity is either stopped (14) or failed (12).

  Alternatively, the activity can be repaired with the help of a forceNavigate API request. In this case, the administrator can determine which outgoing links of the activity should be followed. The state of the activity changes back to finished (11), the transition conditions are not evaluated, but the transition condition of the specified links are considered to be evaluated to true. This means that if the activity is in a parallel flow, all other links are navigated as dead paths.

If an activity is in the stopped state and the enclosing scope is terminated, for example, because of an uncaught fault in a parallel branch, the activity is terminated. Its state changes to the terminated state (13).

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.
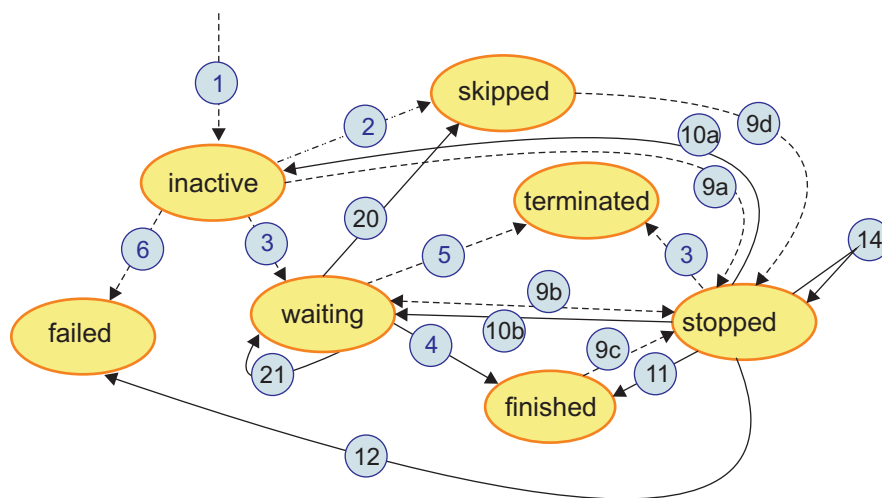
The starting phase of receive and wait activities, and the state transitions to and from the stopped state are the same as for short-lived activities. However, after receive and wait activities are activated, the state changes to waiting instead of running (3). The receive or wait activity is now ready 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, or it can be forced using a force-complete API request. However, if the receive or wait activity has an exit condition with the condition evaluation attribute set to `on exit` and the exit condition evaluates to `false`, the activity state changes to stopped (9b) and not to finished. If an expiration is defined for the activity, an administrator of the activity or an enclosing scope or process can reschedule (21) when it expires without changing the state of the activity.

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. If the **Continue On Error** setting is set to `yes` or the fault is handled by a fault link or fault handler on the enclosing scope, the 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).

A wait or receive activity can be skipped while it is in the waiting state. The state of the activity changes immediately to the skipped state (20). In this case, the transition conditions of the links that leave the activity are evaluated.

State transition diagram for pick (receive choice) activities

The states and state transitions for pick activities (also known as receive choice activities) are shown in the following state diagram.
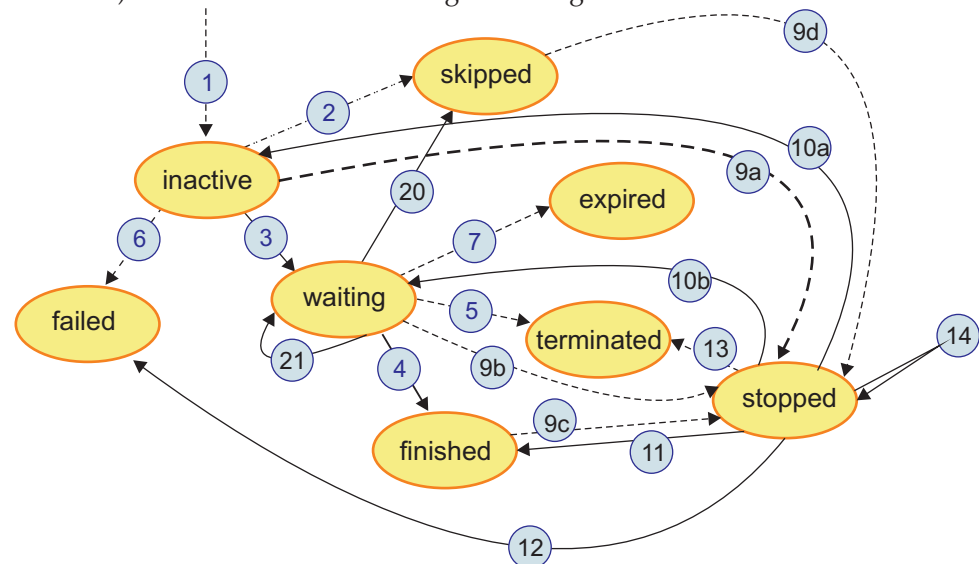
For pick activities, the states and state transitions (1) through (6), and the transitions to and from the stopped and skipped states are the same as for receive activities.

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 invoke activities

For invoke activities, the state diagrams depend on whether the corresponding service is invoked synchronously or asynchronously. 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.
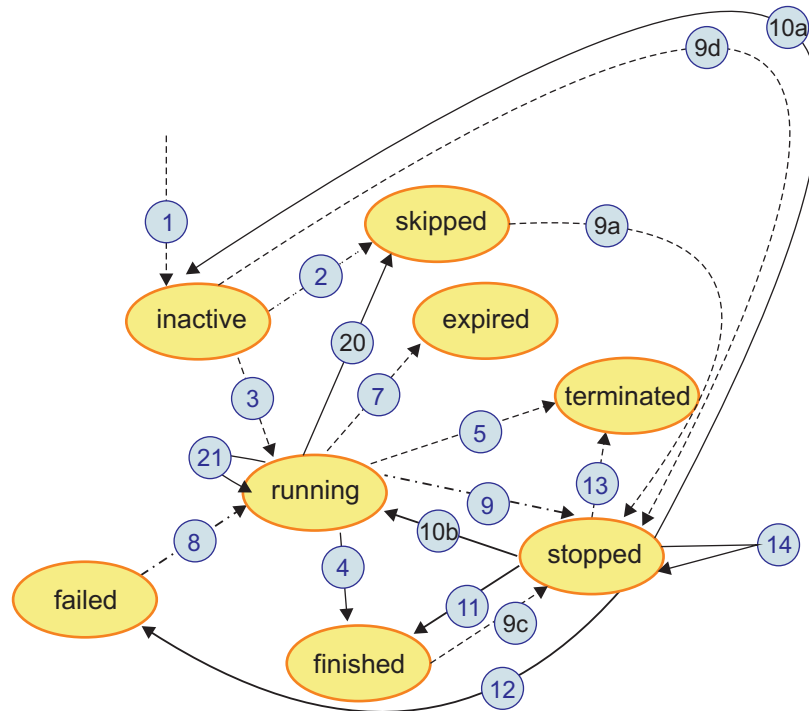
The activation of an invoke activity is the same as the activation of all of the other activity types (1), (2).

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 or 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.

As with all other 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 human task activities with single or sequential ownership

The following diagram shows the states and the state transitions that can occur during the life cycle of a human task activity for tasks with single or sequential ownership.
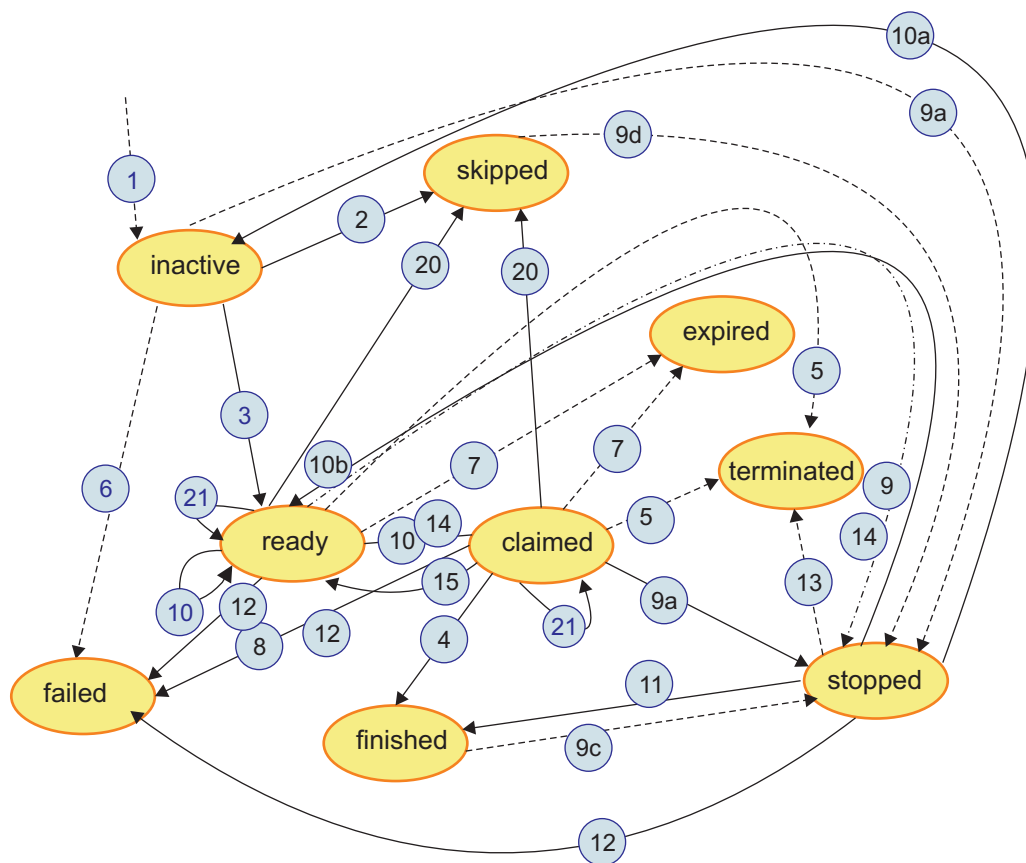
The runtime behavior of a human task activity is similar to that of an invoke activity. The running state of an invoke activity corresponds to the ready and claimed states of a human task activity. The ready state indicates that the activity is available to be worked on by a person. When someone claims 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).

A human task activity can be skipped while it is in the ready or claimed state. In both cases, the state changes to skipped and the inline human task is terminated. In the following navigation step, the transition conditions of the links leaving the activity are evaluated.

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

State transition diagram for human task activities with parallel ownership

The state diagram for tasks with parallel ownership is similar to the state diagram for asynchronous invoke activities. The human task activity is put into the running state when the activity is activated. When all of the required potential owners have completed their work, the activity is finished. The ready and claimed states are not used for activities associated with a task with parallel ownership.

Related concepts

"Fault handling and compensation handling in business processes" on page 33
A fault is any exceptional condition that can change the normal processing of a business process. A fault can be returned from a service invocation, raised explicitly raised by the process, or it can be system fault raised by the runtime environment. A well-designed process should consider faults, and handle them whenever possible. Compensation is one way of handling faults.

"Continue-on-error behavior of activities and business processes" on page 37
When you define a business process, you can specify what should happen if an unexpected fault is raised and a fault handler is not defined for that fault. You can use the **Continue On Error** setting when you define your process to specify that it is to stop where the fault occurs.

Life cycle management of subprocesses

A process that is created and 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.

If the parent is terminated, it must wait until all of the child processes have been terminated. If the invoke activity that called the child process expires, is skipped, force retried or force completed, the child process is terminated first. While the child process is terminating, the invoke activity stays in the running state until the child process reaches the terminated state.

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 within a module, or across module boundaries using SCA Import or Export. 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.

Related concepts

"Life cycle of stand-alone human tasks that are invoked by a business process"
The life cycle of an inline task is always managed by its associated business process. The life cycle of a stand-alone to-do task can be managed by the calling business process depending on the definition of the task.

Life cycle of stand-alone human tasks that are invoked by a business process

The life cycle of an inline task is always managed by its associated business process. The life cycle of a stand-alone to-do task can be managed by the calling business process depending on the definition of the task.

For reuse, it often makes sense to implement a step of the business logic as a separate stand-alone task and to invoke this task from different locations in the main process. When these applications are deployed, the stand-alone task must be deployed to the same Business Process Choreographer database.

A stand-alone to-do task can have a peer-to-peer relationship or a parent-child relationship with the calling process. This relationship determines how the life cycle of the invoked task is managed.

The process-task relationship is determined by the autonomy attribute of the task. This attribute can have one of the following values:

Peer If the task has a peer-to-peer relationship with the business process, the life cycle of the task is independent of the business process.

Child If the task has a parent-child relationship with the business process, some life cycle operations on the process instance are also applied to the task instance. These operations are delete and terminate.

In addition, the following life cycle operations on the calling invoke activity are also applied to the task instance:

- Restarting an invoke activity causes the current task instance to be deleted, and a new task instance to be created and started.
- Forcing completion of the invoke activity causes the task instance to be terminated.
- Skipping the invoke activity in the running state causes the task instance to be terminated.
- Deleting or terminating the invoke activity causes the task instance to be deleted.

If the autonomy attribute of the task is set to child, you can still suspend and resume the task instance independently of the business process.

A parent-child relationship can be established only between processes and tasks 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 process and the task.

Related concepts

"Life cycle management of subprocesses" on page 18
A process that is created and 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.

Dynamic modification of process instances at runtime

Typically, a business process is navigated as defined in the process model. However, sometimes you might need to override the navigation of a process instance at runtime, for example, so that you can repair a process instance, or perform only those activities that are appropriate for the current context.

You can dynamically change the process navigation by jumping forward and back in a process instance, and skipping activities within a process instance. In these situations, you might also need to modify process data that is contained in process variables so that the navigation of the process instance can continue.

Business Process Choreographer Explorer, and the Business Process Choreographer APIs support the dynamic modification of process instances at runtime. In addition, Business Space powered by WebSphere supports redoing parts of a process instance and skipping activities.

Jump forward and back in a process instance

You can use jumps within a process instance to dynamically modify a process instance at runtime. You can jump from one activity (*source activity*) to another

activity (*target activity*). The source activity must be a basic activity in one of the active states; running, waiting, ready, claimed, or stopped. The target activity can be a basic activity or a structured activity.

The following jump actions are available:

Complete and jump
> Complete a human task activity in the claimed state and jump to the target activity

Force-complete and jump
> Force the completion of the activity and continue the navigation of the process from the target activity

Skip and jump
> Skip the source activity and continue the navigation at the target activity

Jump   Jumps from the source activity to the target activity.

For the Jump API only: The source activity must be a basic activity in the skipped, finished, failed, or expired execution state. It must be the last activity that is navigated on the path that contains the activity. The associated process instance must be in the suspended execution state. When the process instance is resumed, navigation is continued at the specified target activity.

The source activity is completed, force-completed, or skipped as part of the jump action. After the jump, the navigation of the process continues at the target activity. You can jump forward in the process, that is, the target activity occurs later in the process instance. You can also jump back to a prior activity in the process.

Jumps are supported between activities in a sequence activity. Jumps are also supported on paths without forks and joins in a generalized flow activity and a parallel-activities activity (also known as a flow activity). For all of these jump actions, the source and the target activity must be on the same nesting level within the containing activity.

Exit conditions on the source activity and for the entry of the target activity are ignored by a jump action.

To perform a jump action, you must be the scope administrator of the enclosing scope, the process administrator, or the system administrator.

Skip an activity

You can also dynamically modify a process instance by skipping activities. You can skip a basic activity that is in one of the active states or a basic activity that might become active later in the process.

If an active activity is skipped, the implementation of the activity is terminated, and the navigation of the process continues after the activity. For example, if the activity has outgoing links, process navigation continues with the evaluation of the transition conditions of the links.

If an activity is skipped that occurs later in the process flow, the activity is marked for skipping. When the navigation reaches the activity, the activity is skipped and the navigation continues after the activity. You can cancel the skip request up until the navigation reaches the activity.

To skip an activity, you must be the scope administrator of an enclosing scope, the process administrator, or the system administrator. In addition, if you are the activity administrator, you can skip an activity that is currently active.

Modify variables

When you change the flow of a process instance at runtime, you often also need to update variables to ensure that the process can flow properly after the jumped or skipped activity. For example, in a repair scenario you can provide valid data before the jump action so that subsequent activities can execute successfully based on that data.

The following actions are supported:
- Get the names of all the variables for a given activity
- Get the actual or initial value of a global or local variable
- Set the value of a global or local variable

To view the value of a variable, you must have at least reader rights for the process or the enclosing scope. To update a variable, you must be the scope administrator, the process administrator, or the system administrator.

Invocation scenarios for business processes

A business process is a Service Component Architecture (SCA) component implementation. 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. These client applications can create and start business process instances, and query and work with existing process instances. The Business Flow Manager API is provided as an EJB, a Web service, a JMS message interface, and a REST interface that you can use to design EJB, Web service, JMS, and REST 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. 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
A business process can call another business process. This can be done using SCA wiring within the same module or across modules. SCA wiring statically associates a caller with another service, also known as *early binding*. When invoking a service offered by another process, late binding

can be used to select the version of the process that is currently valid. This is done using a specification at the partner link of the calling process.

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
Service Component Architecture (SCA) references and SCA interfaces are associated with a WSDL port type containing one or more operations. Operations can be one-way or request-response operations.
- In a one-way operation, the completion of the service is not made known to the invoking client. The service execution ends with the successful invocation of the associated service.
- In a request-response operation, the completion of the service is made known to the invoking client. The service execution ends when the result of the service 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 that is invoked in one of 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, the service, or both are implemented as business processes.
- An endpoint reference (EPR) can be assigned to a BPEL partner link. The EPR determines the endpoint address of the service to be invoked using the partner link. Thus, any service can be invoked dynamically that complies with what SCA allows for dynamic service invocations, for example, Web service binding, MQ JMS binding, MQ 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 4
Business processes can be either long-running or microflows.

"Dynamic binding between business processes and services"
This scenario assumes that a business process is used as a client, and the process model allows BPEL partner links to be assigned when the process runs. Dynamic service bindings allow business processes to invoke services, the addresses of which are determined at runtime. This is especially useful if the service endpoint is unknown at when the process is modeled.

Dynamic binding between business processes and services

This scenario assumes that a business process is used as a client, and the process model allows BPEL partner links to be assigned when the process runs. Dynamic service bindings allow business processes to invoke services, the addresses of which are determined at runtime. This is especially useful if the service endpoint is unknown at when the process is modeled.

The services with which a business process interacts are modeled as partner links in the process model. Before operations on a partner service can be invoked using a partner link, the binding and communication data for the partner service must be available. The relevant information about a partner service is usually set as part of business process deployment.

The Service Component Architecture (SCA) reference that corresponds to the BPEL partner link can be left unwired. In this case, the binding that is used for the invocation defaults to an SCA or a Web service binding, depending on the endpoint address URL. Alternatively, the SCA reference can be prewired to an SCA Import. In this case, the binding and any quality of service specifications are obtained from the SCA Import and only the service endpoint address is overwritten by the endpoint reference.

Include in your process model either an assign activity or a Java snippet activity to which you assign the endpoint reference value of the partner link. If the partner link is not wired, the service is invoked in one of the following ways:

- For a microflow, the service is invoked synchronously
- For a long-running process, the service is invoked asynchronously

Related concepts

"Factors affecting business process interactions" on page 23
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.

Data exchange between business processes and services

A business process can consume service component architecture (SCA) services, or it can be consumed by other SCA services. The way in which 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, 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.

Business processes can be either long-running or microflows.

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. Microflows and long-running processes have different state transition diagrams.

A business process is a Service Component Architecture (SCA) component implementation. 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.
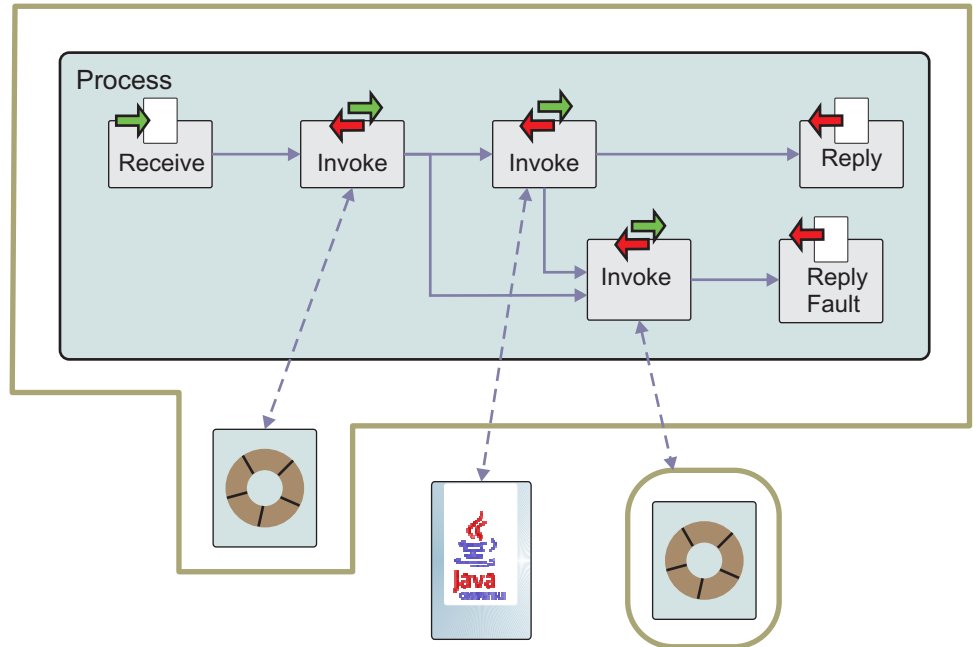
Invoked services and microflow transactions

A microflow runs in one transaction. However, the services that the microflow invokes 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 Service Component Architecture (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	Asynchronous invocation	Synchronous invocation
Synchronous	Synchronous invocation	Synchronous invocation
Asynchronous	Asynchronous invocation	Synchronous invocation

Note: The invocation from a microflow of a request-response operation with a preferred interaction style of "asynchronous" is an example of an antipattern for service invocation. When the invoked service is a long-running process, the microflow transaction can time out before the long-running process completes, and a runtime error occurs.

- The 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 **asynchronousInvocation** qualifier of the microflow must be set to `commit`.

Related concepts
"Compensation handling in business processes" on page 40
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.

Transactional behavior of long-running processes

A long-running process spans multiple transactions. Each transaction is triggered either by a Java Messaging Service (JMS) message or by a work-manager-based implementation.

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.
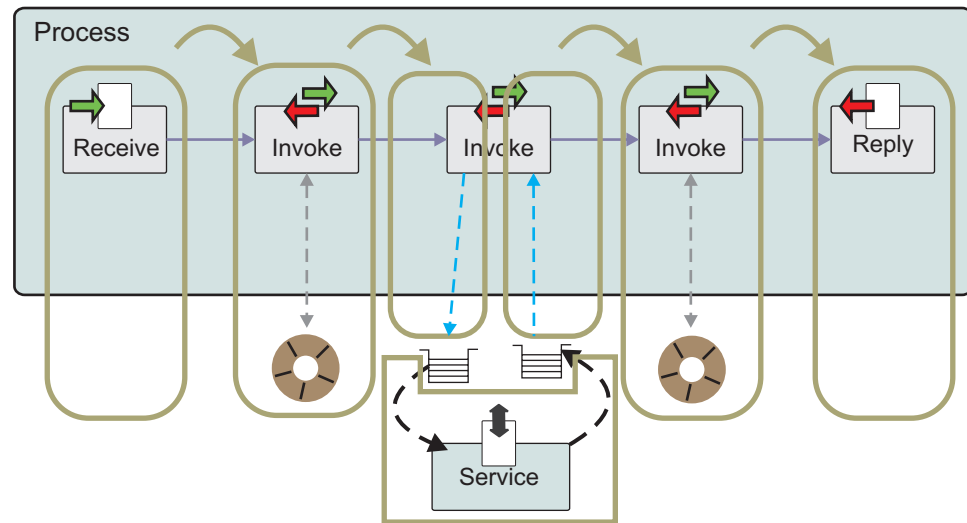
The following describes the transaction boundaries of a long-running process. You can influence transaction boundaries by using the transactional behavior attribute. However, Business Flow Manager can add or remove transaction boundaries at any time.

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.

- 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, factor out the business logic that needs to be executed in a single transaction into a microflow, and invoke it as a 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 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.

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 are 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.

You can also determine whether the transaction that initiates the process is committed after the receive activity, or the receive action of the receive-choice (pick) activity completes by changing the transactional behavior attribute of the corresponding activity. For initiating receive and receive-choice activities, the attribute can take one of the following values:

Commit after
> After the activity completes successfully, the transaction that initiates the

process is committed, and a new one is started. This setting is useful if you invoke the process instance using a synchronous API call.

Participates

The transaction that initiates the process continues after the activity completes. This setting is required if you want to invoke the process instance using the initiateAndClaimFirst API. With this API you can create a new process instance and immediately claim the first human task.

If your process invokes another BPEL process, ensure that the corresponding invoke activity is not part of the transaction that initiates the process. You can achieve by setting the transaction behavior attribute in one of the following ways:

- Set the attribute of the initiating receive or receive choice activity to `Commit after`
- Set the attribute of the invoke activity to `Commit before` or `Requires own`

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, for example, 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.

Concurrent navigation of branches of a parallel forEach activity

The processing of each branch of a parallel forEach activity is started in its own, separate transaction. Thus parallel execution of these branches is enabled.

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, for example, those for DB2 databases. However, asynchronous services on the branches of a parallel forEach activity are executed concurrently, thus parallelism forEach activities can be achieved.

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, 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 transactions.

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 fault can be returned from a service invocation, raised explicitly raised by the process, or it can be system fault raised by the runtime environment. A well-designed process should consider faults, and handle them whenever possible. Compensation is one way of handling faults.

Related concepts

"State transition diagrams for activities" on page 11
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 raising in business processes

You can raise faults using throw and rethrow activities, or programmatically using a Java snippet activity. The invocation of services can also raise faults.

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

Throw and rethrow activities for fault raising

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. A fault thrown by a throw activity should be caught and handled in 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 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.

Fault raising in Java snippets

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.

Fault propagation 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 method 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. The
navigation of the process continues until it reaches an end state.

Fault handling in business processes

When a fault occurs in a process, the navigation continues with the fault handler
or fault link.

Fault handlers can be specified for invoke activities, scopes, and on the process.
Fault links can be specified for generalized-flow activities. Scopes and all basic
activities, except for throw and rethrow activities, can be the source activity of a
fault link.

A fault handler or fault link can catch a specific fault name, fault type, or both.
When a fault occurs, Business Flow Manager uses the following rules to match the
fault with a fault handler or fault link on the enclosing scope, or on the activity
where the fault occurred.

- If an invoke activity without a fault handler or any other basic activity is the
  source of one or more fault links, Business Flow Manager tries to find a
  matching a fault link. If a fault link is not available, it then tries to find a
  matching fault handler on the enclosing scope.
- If an invoke activity or a scope with one or more fault handlers is the source of
  one or more fault links, Business Flow Manager tries to find a matching fault
  handler. If a fault handler is not available, it runs the default fault handler and
  then tries to find a matching fault link. If a matching fault link is not available, it
  tries to find a matching fault handler on the enclosing scope.
- If the fault does not have any associated fault data, Business Flow Manager uses
  a fault handler or fault link with the matching fault name. If a fault handler or
  fault link is not found, it uses the catch-all fault handler or fault link if one is
  available. A fault without any data cannot be caught by a fault handler or fault
  link that has a fault variable defined.

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

If a fault is raised that does not match any of the fault handler or fault link 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 enclosing scopes in the reverse order of completion of the corresponding scopes. If the scope is the source of one or more fault links, Business Flow Manager then tries to find a matching fault link. If a matching fault link is not available or the scope is not the source of any fault links, the default fault handler 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 or fault links that are available.

If the fault is not caught by any of the specific fault handlers or fault links, or by the catch-all fault handler or catch-all fault link, 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.

Fault handler considerations

When you define 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 response.
  - Invoke a human task to correct the issue.
  - For microflows, 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 **Continue On Error** setting on the process to handle the fault administratively.

Fault link considerations

When you use fault links in your business process, consider the following:
- A fault link is activated for faults that occur within the source activity only. The evaluation of conditions of normal links is not part of the execution of the activity.
- If the source activity of the fault link is a scope activity, the fault handler of the scope activity is evaluated first when a fault occurs inside the scope. However, the fault handler can rethrow the fault. In this case, a fault link of the scope can catch the fault and can be navigated.

- If an activity is the source of multiple fault links, only one of the fault links can be navigated when a fault occurs.
- The target activity of the fault link will be executed normally. Compensate and rethrow activities in fault handlers cannot be the target of a fault link.
- When a fault contains fault data, a variable of the fault data type needs to be declared on the surrounding scope. The fault link must reference this variable so that the target activity of the fault link has access to the fault data.

Transactional considerations

In general, if a runtime exception is raised by a called service that participates in the current transaction, the current transaction rolls back to the state that the process was in before the transaction started. If the transaction was triggered by Business Flow Manager, it is automatically retried and the service is called again. When the transaction is retried, the Business Flow Manager can overrule the transactional behavior attribute of the activity, and introduce additional transaction boundaries in order to determine the activity that is causing the runtime exception. If the called service raises the runtime exception repeatedly, the runtime exception is stored in a separate transaction. During the next retry the service is not called again but the stored exception is restored and the navigation continues with the fault handling.

Retrieval of fault data for business processes

Your process can handle runtime faults and BPEL standard faults. To handle these faults, you might need access to the information about the fault.

You can use one of the following constructs to retrieve this information:

- The getCurrentFaultAsException method

  You can use the getCurrentFaultAsException method in a fault handler to retrieve data for runtime faults, BPEL standard faults, and business faults. This mechanism is useful in combination with a catch-all fault handler because this type of fault handler does not have an associated variable to capture fault data, or if you are catching the runtimeFailure fault.

  The getCurrentFaultAsException method can be called in a Java snippet activity. The 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. You can therefore 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()
  ```

- A typed fault variable for the fault handler or fault link

  For runtime faults and BPEL standard faults, you can define a typed fault variable for your fault handler or fault link to capture the fault data. The fault variable must be typed by the StandardFaultType complex type.

Related reference

"A fault is not caught by the fault handler" on page 759
A fault handler is attached to an invoke activity to catch specific faults that are
thrown by the invoked service. However, even if the invoked service returns the
expected fault, the fault handler is not run.

Continue-on-error behavior of activities and business processes

When you define a business process, you can specify what should happen if an
unexpected fault is raised and a fault handler is not defined for that fault. You can
use the **Continue On Error** setting when you define your process to specify that it
is to stop where the fault occurs.

For most activities, the continue-on-error behavior is the same as for the process.
You can specify the continue-on-error behavior explicitly for invoke, Java snippet,
custom, and human task activities. By default, the continue-on-error behavior of
these activities is also the same as for the process. If an unexpected fault is
detected, fault handling of the activity is started. If the **Continue On Error** setting
is set to Yes, then the standard fault handling is applied. If the **Continue On Error**
setting for the activity or the process is set to No and the fault is not handled by a
fault handler on the immediately enclosing scope or by a fault link leaving this
activity, the activity is stopped. If the activity has an associated fault handler or
compensation handler, the immediately enclosing scope is the activity itself. In all
other cases, the immediately enclosing scope is the next surrounding scope in
which the activity is contained. If a catch-all fault link or fault handler is defined
on the immediately enclosing scope, the value of the **Continue On Error** setting
does not have any effect, because the fault is always handled and the activity is
never stopped.

For activities that stopped because of an unexpected fault, you can use the
stopReason property of the activity to determine the cause of the fault and the
actions that you can take to repair it. The following table shows the values the
stopReason property can take in fault situations.

Value of the stopReason property	Cause	Allowed actions
STOP_REASON_ ACTIVATION_ FAILED	The evaluation of the join condition of the activity failed. You can use the force join condition to set the condition to true or false or to reevaluate it.	• Force retry • Force the join condition

Value of the stopReason property	Cause	Allowed actions
STOP_REASON_ IMPLEMENTATION_ FAILED	The implementation of the activity threw a fault.	
	This value is set if the implementation of the activity failed, for example: • A service called by an invoke activity returned a fault that is not handled by a fault handler. • A timeout expression failed when a wait activity was activated. • The evaluation of an exit condition of an activity failed.	• Force retry • Force complete
	If the evaluation of a while or repeatUntil condition failed, the value of the kind property is either or .	• Force retry • Force complete • Force the loop condition
	If the evaluation of a forEach counter or condition failed, the value of the kind property is either KIND_FOR_ EACH_SERIAL or KIND_FOR_ EACH_PARALLEL.	• Force retry • Force complete • Force the values of the forEach counter
	The evaluation of the branch expression in choice activity (also known as a switch activity) failed.	Force the navigation of the branch
STOP_REASON_ FOLLOW_ON_ NAVIGATION_ FAILED	The evaluation of a transition condition of an outgoing link failed. This value is set in one of the following situations: • In a parallel flow (also known as a parallel activities activity), after an activity completed, the transition conditions of the outgoing links were evaluated, and one of them produced a fault. • In a cyclic flow, if none of the outgoing links qualify for follow-on navigation.	• Force the navigation of the outgoing link • Force complete
STOP_REASON_ EXIT_CONDITION_FALSE	The criteria for the exit condition were not met. This value is set only when the exit condition is evaluated on exit of an activity and the condition evaluates to `false`.	• Force retry • Force complete

Stopped activities can be repaired with the one of the Business Process Choreographer API force or skip methods, or with the corresponding actions in Business Process Choreographer Explorer. You can also modify the activity variables before you carry out the repair action. The following table summarizes the API methods and the Business Process Choreographer Explorer actions.

Table 3.

Repair action	API method	Action in Business Process Choreographer Explorer
Force retry	forceRetry	**Restart**
Force complete	forceComplete	**Force Complete**
Force the join condition	forceJoinCondition	**Repair Join**
Force the loop condition	forceLoopCondition	**Next Iteration** or **End Loop**
Force the values of the forEach counter	forceForEachCounterValues	**Repair For Each**
Force the navigation a branch within a choice activity	`forceNavigate(..., int positionOfBranch)`	**Force Case Navigation** or **Force Case Execution**
Force the navigation of an outgoing link	`forceNavigate(..., ... linksToBeFollowed)`	**Force Navigation**
Force complete and jump	forceCompleteAndJump	**Force Complete Source Activity and Jump**
Skip	skip	**Skip** or **Skip Activity**
Skip and jump	skipAndJump	**Skip Source Activity And Jump**

Related concepts

"State transition diagrams for activities" on page 11
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.

Related tasks

"Processing human task activities" on page 553
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.

"Forcing the completion of an activity" on page 559
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.

"Retrying the execution of a stopped activity" on page 561
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.

Compensation handling in business processes

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.

Related concepts

"Transactional behavior of microflows" on page 26
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.

Compensation handling in 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.

Typically, undo actions are specified in the process model 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, 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.

Compensation handling in long-running processes

Compensation for long-running processes is also known as *business-level compensation*. The compensation logic can be defined on the scope level, for invoke activities, or for human task activities. This means that either a single invoke activity or human task activity, or a set of activities in a scope can be compensated.

In BPEL processes, compensation is triggered by fault handlers of a scope or the process, or by the compensation handler of a scope or an invoke activity.

In BPMN processes, compensation can be triggered from outside compensation and fault handlers via the compensate activity. This support is available for generalized flow activities and collaboration scopes, where compensation can be triggered from within the flow or scope. In this case, the compensate activity triggers the compensation of all the activities in the current scope that completed successfully before the compensate activity started. When all of these activities are compensated, the compensate activity completes and the navigation of the process continues. If the compensation of a completed activity fails, the compensate activity continues to trigger the compensation of the remaining activities. When the compensation of these activities finishes, the compensate activity passes the fault from the failed compensation to the enclosing scope

Compensation of child processes

A long-running process can also compensate child processes that have successfully completed. A child process is called by an invoke activity. This invoke activity can have a compensation handler associated with it. If a compensation handler is defined, its logic determines whether compensation for the child process is triggered by the compensate activity. If the compensation handler does not use the compensate activity, compensation for the child process does not occur. If the invoke activity does not have a compensation handler, then compensation for the child process is triggered automatically.

For long-running child processes, compensation is run for all of the successfully completed, directly nested scopes on the process level in the reverse order of their completion. For microflow child processes, compensation runs the undo actions of all of the invocations in the reverse order of the forward execution of the invocations.

In general, only human task activities, invoke activities (including those activities that called a child process), and scope activities, that complete successfully are compensated. Compensation of these activities can be done in one of the following ways:

- Compensate everything that is enclosed in the scope

  The compensation of activities that started a child processes is integrated into the sequence of compensating the other activities in the scope. Compensation of everything in the scope is activated by the default compensation handler of the current scope, or by a compensate activity that does not specify a target.

  For example, the activities A, B, and C are enclosed in a scope. Activity B is the invoke activity for a child process, and activities A and C are scope activities. All of the activities complete successfully in sequence. When the parent process is

compensated, these activities are compensated in the reverse order in which they completed: activity C, activity B, and then activity A.

- Compensate specific activities in the scope

  Compensation is activated by compensate activities that target a scope or an activity. An invoke activity that started a child process can be the target of a compensate activity, even if the invoke activity does not have a compensation handler defined.

Recovery from infrastructure failures

A long-running process spans multiple transactions. If a transaction fails because of an infrastructure failure, Business Flow Manager provides a facility for automatically recovering from these failures.

In a long-running process, the Business Flow Manager sends itself request messages that trigger follow-on navigation. For each incoming request message, a new transaction is started and the request message is passed to the Business Flow Manager for processing. Each transaction consists of the following actions:

- Receive a request message
- Navigate according to the request
- Store the state in the database
- Send request messages that trigger follow-on transactions.

Business Flow Manager uses the following queues for coping with infrastructure failures:

- The retention queue stores failed messages that will be automatically retried
- The hold queue stores messages that have failed more times than the retry limit, and can indicate a more serious infrastructure failure or a damaged message that cannot be processed

When messages are processed successfully, it is inferred that the infrastructure is available. However, Business Flow Manager might fail to process a message in the following situations:

Cause	Response
Unavailable infrastructure	In normal processing mode, for a specified time, all messages are kept 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.

If the infrastructure is unavailable, and the retention queue is full, message processing is switched from normal processing to *quiesce mode*. In quiesce mode, the message processing is slowed down until the infrastructure is available again. When the infrastructure becomes available, message processing switches back to normal mode.

Normal message processing

During normal processing, a message is processed as follows:

- If a message fails three times, it is stored in the retention queue.
- When a message is in the retention queue, the options are as follows:

– When a subsequent message is processed successfully, all of the messages from the retention queue are moved back to the input queue. For each message, a count is maintained of how often the message is sent to the retention queue. This count is referred to as the *retention queue traversal count*. If this count exceeds the retry limit for a given message, the message is put in the hold queue.

– If the next message fails, 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, all of the messages are moved from the retention queue to the input queue, and message processing is put into quiesce mode.

Message processing in quiesce mode

In quiesce mode, processing a message is attempted periodically. Messages that fail to be processed are put back in the input queue, without incrementing either the delivery count or the retention queue traversal count. As soon as a message can be processed successfully, message processing is switched back to normal mode.

Retry limit

The retry limit defines the maximum number of times that a message can be transferred to the retention queue before it is 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 is started. If the last retry 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.

To change this Business Flow Manager property, click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and 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 Business Flow Manager more tolerant of infrastructure failures, increase the value.

To change this Business Flow Manager property, click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Flow Manager**.

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, administrative scripts, or failed event manager.

Authorization for business 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. Business Flow Manager uses the activity roles for navigation and authorization. Each activity role matches exactly one human task role. 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.

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 Java EE role or an instance-based role.

Java EE roles for business processes

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

The following Java Platform, Enterprise Edition (Java EE) 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 authorization roles for business processes and activities

A set of predefined authorization roles is provided for business processes and activities. You can assign these roles 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, create, delete, and transfer work items, change the navigation of the process at runtime, for example, by skipping activities. 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	Repair activities in a process.
Scope reader	View the properties of the activities and variables in the scope. Members of this role also automatically become the reader for the properties of activities, and the inline to-do tasks (including the subtasks, follow-on tasks, and escalations) that are associated with human task activities in the scope.
Scope administrator	Administer the activities in the scope, including updating variables for activities, skipping activities, and canceling skip requests. 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 in the scope.

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, unless you have transferred the process ownership to another user.

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.

Role	People assignment
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. 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.
Scope reader	The scope reader is specified by setting the reader role on the administration task that is associated with the scope. This role is inherited by all of the activities in the scope.
Scope administrator	The scope administrator is defined by an administration task that is assigned to the scope. This role is inherited by all of the activities in the scope.

Note: If process administration is restricted to system administrators, then instance-based administration is disabled. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. In addition, reading, viewing, and monitoring a process instance or parts of it can only be performed by users in the BPESystemAdministrator or BPESystemMonitor roles. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

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.

In addition, when the process is called by a Service Component Architecture (SCA) client, you can restrict the set of users that is allowed to start the process by setting specific SCA security qualifiers when you install the process.

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 Manager API, or by initiating the process using the Business Flow Manager API. Both methods 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 is 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 is called directly by Business Flow Manager.

– 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.

If an administration task is assigned to the process, the administrator role of the administration task is inherited by the process. A process administrator can perform various actions on the process, including creating and starting a process instance.

Related concepts

"Authorization for interacting with a business process"
A long-running process can have multiple receive activities, pick (receive choice) activities, and event handlers. These are served by submitting a request to the appropriate operation of the corresponding process instance. The process instance is identified implicitly by providing a unique correlation set instance in the request according to the correlation set defined in the process model.

Authorization for interacting with a business process

A long-running process can have multiple receive activities, pick (receive choice) activities, and event handlers. These are served by submitting a request to the appropriate operation of the corresponding process instance. The process instance is identified implicitly by providing a unique correlation set instance in the request according to the correlation set defined in the process model.

A receive or pick activity can be used to create a process instance. So, interacting with an existing process instance by submitting a request to a process is similar to starting a new process instance.

The set of users that are authorized to submit a request to a process instance is determined by the invocation task that is associated with the receive or pick activity, or the event handlers, and by the administration task that is associated with the process.

You can use human tasks in the following ways to interact with a process instance:

• Assign an inline invocation task to the receive activity, pick activity, or event handler.

  The potential starters that are defined for the inline invocation task submit a request to the corresponding operation of the process. The invocation task is optional. If an invocation task is not defined, everyone is authorized to submit a request.

• You can also use a stand-alone human task to secure inbound operations of a business process. The same rules and restrictions apply as for stand-alone invocation tasks for process-creating operations.

• Assign an administration task to the process.

  The administrator role of the administration task is inherited by the process. A process administrator can interact with a process using its operations.

If an administration task is not specified for the process, the starter of the
process becomes the process administrator. In this case, the process starter is
allowed to submit requests to operations of the process instance.

If a process uses the same operation in different receive, pick (receive choice)
activities, or event handlers, and the receiving process instance is currently not
expecting a request, because the corresponding receive or pick (receive choice)
activity is not yet waiting or the event handler is not yet active, then the user
sending the request must be authorized to send a request to all of these activities
and event handlers, otherwise the request will be rejected.

Related concepts

"Authorization for creating and starting business processes" on page 47
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.

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

Note: If process administration is restricted to system administrators,
instance-based administration is disabled, and all administrative actions on
processes, scopes, and activities are limited to users in the
BPESystemAdministrator role. For more information about this administration
mode, see "Alternate process administration authorization mode" on page 50.

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, terminate the process instance.

An administration task is associated with every business process. If an
administration task is not modeled 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.

Scope administration

You can model an administration task for the scope that defines scope readers and
scope administrators. A scope reader is allowed to view local variables. Scope
administrators are allowed to repair activity instances in the scope, and to view
and update local variables. If the scope is enclosed in another scope, the scope
reader and administration rights are inherited by the enclosing scope. Scope
readers and administrators also become readers and administrators of the activities
in the scope.

Activity administration

The administrator role for an activity administration task determines who is
allowed to administer the corresponding activity. The activity administrator can,

for example, restart the activity. The administration task is created as soon as administrative actions, that is, restart or complete, can be performed on the activity instance. Reader and administrator roles of the process, and reader and administrator roles of the enclosing scopes are automatically propagated to the activities.

In addition, you can model administration tasks for activities in the following ways:

- For each invoke or snippet activity. This administration task determines who is allowed to administer the activity in addition to the process administrators.
- A default administration task for activities on the process level that applies to any activity that does not have an administration task assigned to it.

Alternate process administration authorization mode

If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

When the Business Flow Manager custom property ProcessAdministration is set to the value useSystemAdminAuthorizationOnly, a different set of rules apply to what happens when a new process instance is started and which user IDs are allowed to view, monitor, or perform administrative actions on process instance, scope instances, and activity instances.

The purpose of this alternate administration mode is to improve performance by reducing the number of objects created in the Business Process Choreographer database, which can reduce the response times for task list and process list queries.

This administration mode only affects newly created objects, existing instances are subject to the administration rules that applied when the instances were created. Take care to avoid any disruption to people or automated processes that perform administrative actions using user IDs that are not in the appropriate role.

The alternate administration authorization mode implements the following changes:

Process instances

When a new process instance in started, whether by a user or another component, no administration task instance is created for the process instance. The only work item created for the process instance is the process starter work item.

Administration

Only users in the BPESystemAdministrator role can perform administration actions on the process instance. For example, terminating or restarting a failed process instance, or updating the contents of a global or local variable.

Viewing and monitoring

Only users in either the BPESystemAdministrator role or BPESystemMonitor role can view or monitor the process instance or parts of it. For example, viewing the progress of a process instance in the Business Process Choreographer Explorer, or using Business Flow Manager APIs to read the contents of variables that belong to a process instance.

Scope instances

When a scope that has an associated administration task is activated, no administration task instance is created and no work item is created.

Administration

Only users in the BPESystemAdministrator role can perform administration actions on the scope instance. For example, jumping from one activity in the scope to another activity, or performing a skip, force-retry, or force-complete on an activity inside the scope.

Viewing and monitoring

Only users in either the BPESystemAdministrator role or BPESystemMonitor role can view or monitor the scope instance or parts of it.

Activity instances

When an activity instance needs an administrative action performed on it, for example, because it stops due to a failure in its implementation, no administration task or work items are created.

Administration

Only users in the BPESystemAdministrator role can perform administration actions on the activity instance. For example, performing a force-complete, force-navigate, or force-retry on an activity instance that stopped.

Viewing and monitoring

Only users in either the BPESystemAdministrator role or BPESystemMonitor role can view or monitor the activity instance or parts of it. For example, reading the contents of variables that belong to an activity instance.

This administration mode might not be suitable for your needs if any of the following conditions apply to your system:

- If your process template was modeled to include necessary administration tasks, enabling this administration mode will deactivate the administration tasks and all features relating to them, for example escalations.
- Adding all users that need to perform the above administration tasks in the BPESystemAdministrator role might grant them more rights than they need.

Related tasks

Restricting process administration to system administrators
Switching off the default instance-based administration means that only users in the BPESystemAdministrator role can administer process instances, scope instances, and activity instances. Also, only users in the BPESystemMonitor role will be able to view or monitor process instances. This can improve performance because it reduces the database load and the rate that the database grows.

Human tasks overview

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.

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).

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 tasks
> This is where a service component (such as a business process) assigns a task to one or more people as something for them to do. A to-do task can be implemented either stand-alone or inline.
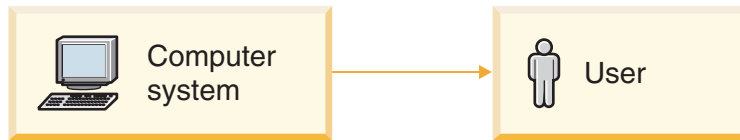
Invocation tasks

This is where a person can "assign" a task to a service component. In such a case, a person invokes an automated service, such as a business process.
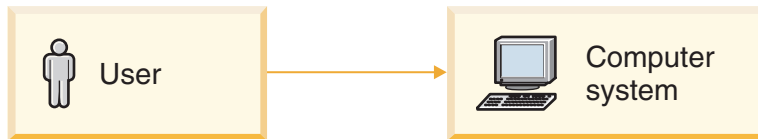
An invocation task can be implemented either stand-alone or inline. When it is inline, an invocation task allows a person to invoke the operations that a business process exposes through activities, such as receive or pick activities, or event handlers.

Collaboration tasks

This is where a person assigns a task to one or more people. This task kind enables a person to share work with other people in a structured and controlled way.

A collaboration task is stand-alone, in that there is no interaction between it and any other component. It is self-contained and implements a stand-alone human interaction without any reference or interface to another service.

Administration tasks

This type of task grants a person administrative powers such as the ability to suspend, terminate, restart, force-retry, or force-complete a business process. Administration tasks can be set up on either an invoke activity, or the process as a whole.
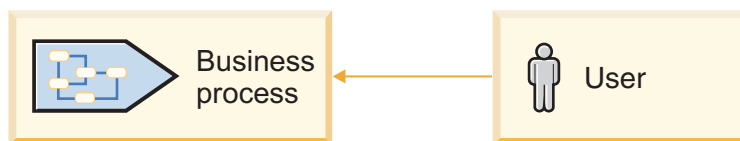
This type of task is only available within a business process (inline task).

Related concepts

"State transition diagrams for to-do tasks" on page 86
To-do tasks are created automatically by a client application, or calling component. They 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). During the life cycle of a to-do task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

"State transition diagrams for collaboration tasks" on page 90
Collaboration tasks support people when they perform work for other people. During the life cycle of a collaboration task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

"State transition diagrams for invocation tasks" on page 93
Invocation tasks support people when they invoke services. During the life cycle of an invocation task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

"State transition diagrams for administration tasks" on page 95
Administration tasks support people in administering business processes and their activities. During the life cycle of an administration task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

"Stand-alone and inline tasks" on page 68
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*.

Versioning of human tasks

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

You can include versioning information when you model the stand-alone human task in WebSphere Integration Developer. The version of a task is determined by its valid-from date. This means that different versions of a task can have the same task name, but they have different valid-from dates. The version of a task 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 is used is made either during modeling, or when the task model is deployed. The calling component invokes a dedicated, statically-bound task according to the Service Component Architecture (SCA) wiring. Even if another version of the task exists that is valid according to its valid-from dates, the current statically wired task 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 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 is used is determined when the task instance is created. In this case, the version of the task that is currently valid is used. A newer version of a task supersedes all of the previous template versions. Existing task instances

continue to run with the task with which they were associated when they started. This leads to the following categories of tasks:

- Currently valid tasks are used for new task instances
- Tasks that are no longer valid might still be valid for running task instances
- Tasks 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 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.

Generally, a task instance inherits all of its properties from the corresponding task template with the following exceptions:

Column name in TASK_TEMPL view	Inherited by task instance	Comments
VALID_FROM	No	Not needed by the task instance.
CONTAINMENT_CTX_ID	No	Task instances are deleted according to a different set of rules than their corresponding task templates.
IS_AD_HOC	No	Not needed by the task instance: • An ad hoc task template creates a task instance that is not ad hoc. • An ad hoc task instance does not have a task template.
IS_INLINE	Usually	The property is not inherited in the following situations: • A subtask instance cannot be inline, even if its template is defined as inline. • A follow-on task instance cannot be inline, even if its template is defined as inline. • A human task activity instance is always related to an inline task instance.
STATE	No	A task template must be in the STATE_STARTED state to create and start task instances. The instances are then in the STATE_READY state.

In addition, all of the custom properties of a task template (TASK_TEMPL_CPROP view) are inherited by the custom property instances of a task instance

(TASK_CPROP view). The multilingual description of a task template (TASK_TEMPL_DESC view) has a row for each locale. A task instance (TASK_DESC view) inherits these rows.

Modification of task instance properties at runtime

At runtime, the properties of a task instance, such as its name, who can work on the task, and its duration settings are copied from its template. Typically, a task instance keeps the predefined values of these properties throughout its life cycle. However, sometimes you might need to override these predefined values at runtime in response to your business needs.

In addition, the predefined values of some properties can contain references to other properties: they are then called replacement variable expressions.

Business Process Choreographer Explorer, Business Space, and the Business Process Choreographer APIs support the modification of task instances at runtime.

Properties that can be changed at runtime

You can use the Business Process Choreographer APIs to change the following properties of a task instance. Some of these properties can also be changed using Business Process Choreographer Explorer, or Business Space.
- Business relevance
- Context authorization of owner
- Deletion time
- Description
- Display name
- Due time
- Escalation state
- Event handler name
- Expiration time
- Name
- Namespace
- Parent context ID
- Priority
- Read
- Supports claim if suspended
- Supports delegation
- Supports follow-on tasks
- Supports subtasks
- Type
- Work basket

Changes to the expiration, deletion, and due times for tasks at runtime

Sometimes a business situation requires that you change the due, expiration, or deletion time that was originally defined for a task. The task state determines which of these times you can reschedule, cancel, and start at runtime, and when these actions can be taken. You can use Business Process Choreographer Explorer to change these times, or you can use the update method of the Human Task Manager API to modify the appropriate task property.

You can change the due time, expiration time, or deletion time of a task in one of the following ways:

- Using a specific time
- Using a duration, for example, 2 days, based on a calendar, such as a business calendar. You can also use the following constants in duration expressions:

DURATION_ZERO
> The task becomes due or it expires immediately after it starts, or it is deleted immediately after it completes (depending on the auto delete setting of the task and the task end state). If the task has started, it becomes due immediately, or it expires immediately. If the task is in an end state, it is deleted immediately if auto deletion is set for the task.

DURATION_INFINITE
> The task does not become due, it does not expire, or it is not deleted.

For a chain of tasks, each task in the chain has its own due time. However, only the first task in the chain has an expiration time and this is shared by all of the other tasks in the chain. Subtasks have their own due time and expiration time. If deletion is supported for the task kind, you can change the deletion time.

Due time

You can use the dueTime and the durationUntilDue properties to change the due time of a task. The task state determines which of these properties you can use.

Inactive state
> In this state, you can override the duration until due value that was defined for the task by setting the durationUntilDue property to a valid value for the calendar that is used by the task, or to one of the constant values. To specify that the task becomes due immediately after it starts, for example, because the task has a high priority, set the durationUntilDue property to DURATION_ZERO. To prevent the task from becoming due, set the durationUntilDue property to DURATION_INFINITE.

An active state (ready, claimed, running, or forwarded)
> If the task is in any of these states, you can reschedule or cancel the due time. In addition, if the task does not have a due time, for example, because DURATION_INFINITE was set while the task was inactive, you can set a due time.
>
> You can set a due time or reschedule it by setting either the dueTime property or the durationUntilDue property to a valid value for the calendar that is used by the task. To cancel the due time, set the durationUntilDue property to DURATION_INFINITE. To force the task to become due immediately, for example, because it is associated with an urgent customer request, set the durationUntilDue property to DURATION_ZERO.

Expiration time

You can use the expirationTime and the durationUntilExpires properties to change the expiration time of a task. The task state determines which of these properties you can use.

Inactive state
> In this state, you can override the duration until expires value that was defined for the task by setting the durationUntilExpires property to a valid

value for the calendar that is used by the task, or to one of the constant values. To specify that the task expires immediately after it starts, set the durationUntilExpires property to DURATION_ZERO. To prevent the task from expiring, set the durationUntilExpires property to DURATION_INFINITE.

An active state (ready, claimed, running, or forwarded)
If the task is in any of these states, you can reschedule or cancel the expiration time. In addition, if the task does not have an expiration time, for example, because DURATION_INFINITE was set while the task was inactive, you can set an expiration time.

You can set an expiration time or reschedule it by setting either the expirationTime property or the durationUntilExpires property to a valid value for the calendar used by the task. To cancel the expiration time, set the durationUntilExpires property to DURATION_INFINITE. To force the task to expire immediately, for example, because it is no longer required, set the durationUntilExpires property to DURATION_ZERO.

Deletion time

The auto delete setting of the task and the task end state determine whether a task is deleted when the deletion time is reached. You can use the deletionTime and the durationUntilDeleted properties to change the deletion time of a task. The task state determines which of these properties you can use.

Inactive state or an active state (ready, claimed, running, or forwarded)
If the task is in any of these states, you can override the duration until deleted value that was defined for the task by setting the durationUntilDeleted property to one of the constant values. You can delete the task immediately after it reaches one of the end states by setting the durationUntilDeleted property to DURATION_ZERO. To cancel the deletion time, set the durationUntilDeleted property to DURATION_INFINITE.

An end state (finished, failed, forwarded, terminated, or expired)
If the task is in any of these states, you can reschedule or cancel the deletion time. In addition, if the task does not have a deletion time, for example, because DURATION_INFINITE was set while the task was inactive, you can set a deletion time.

You can set a deletion time or reschedule it by setting either the deletionTime property or the durationUntilDeleted property to a valid value for the calendar used by the task. To cancel the deletion time, set the durationUntilDeleted property to DURATION_INFINITE. To force the task to be deleted immediately, set the durationUntilDeleted property to DURATION_ZERO.

Changes to the timing of an escalation at runtime

Sometimes a business situation requires that you change the timing of an escalation that was specified when the escalation was defined. The escalation state determines which of these times you can change, and when these actions can be taken. You can use the update method of the Human Task Manager API to modify the appropriate escalation property. You can also use the Tasks List widget or the Escalations List widget in Business Space to override the scheduled escalation time, and start an escalation immediately.

You can change the timing of the initial escalation and when the escalation repeats in one of the following ways:

- Using a specific time
- Using a duration, for example, 2 days, based on a calendar, such as a business calendar. You can also use the following constants in duration expressions:

  DURATION_INFINITE
  > The initial escalation or the repeated escalations are cancelled.

Initial escalation

You can use the escalationTime and the durationUntilEscalated properties to change the timing of an escalation. The escalation state determines which of these properties you can use.

Inactive state
> In this state, you can override the duration until escalated value that was defined for the escalation by setting the durationUntilEscalated property to a valid value for the calendar that is used by the task. To prevent the escalation from starting, set the durationUntilEscalated property to DURATION_INFINITE.

Waiting state
> In this state, you can reschedule the initial escalation or cancel it. You can reschedule the escalation by setting either the escalationTime property or the durationUntilEscalated property to a valid value for the calendar that is used by the task. To cancel the escalation, set the durationUntilEscalated property to DURATION_INFINITE.
>
> You can also trigger the escalation manually using the Human Task Manager API triggerEscalation method.

Repeated escalations

If the task is in the escalated state and the expected task state has not yet been reached, you can use the escalationTime and the durationUntilRepeated property to change the timing of repeated escalations depending on the escalation state.

Inactive and waiting state
> In this state, you can override the duration until repeated value that was defined for the escalation by setting the durationUntilRepeated property to a valid value for the calendar that is used by the task. To prevent the escalation from repeating, set the durationUntilRepeated property to DURATION_INFINITE.

Escalated state
> In this state, you can reschedule the repetition of the escalation or cancel it. You can reschedule the repetition of the escalation by setting either the escalationTime property or the durationUntilRepeated property to a valid value for the calendar that is used by the task. To cancel the repetition, set the durationUntilRepeated property to DURATION_INFINITE.

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.

Replacement variables in human tasks

Replacement variables are used in the definitions of human tasks to refer to a value of an element that is resolved at runtime. These variables represent task and process-related data, such as people assigned to a task or custom properties for the task. This data is available at runtime for all or part of the life cycle of a task instance.

Only certain task elements can contain replacement variables. These task elements are defined for the task template, or for a task model that was created at runtime using the Business Process Choreographer APIs. These definitions are inherited by the task instance when it is created. The following task elements can include replacement variables.

- Duration until delete
- Duration until expires
- Duration until overdue
- Escalation custom properties
- Escalation description
- Escalation duration
- E-mail subject and body for escalation notification e-mails
- People assignment for administrators
- People assignment for editors
- People assignment for escalation receivers
- People assignment for potential instance creators
- People assignment for potential owners
- People assignment for readers
- People assignment for potential starters
- Task custom properties
- Task description
- Task priority
- Task type

Actions that initialize or change replacement variables for human task elements:

This describes which Human Task Manager replacement variables can be considered to be initialized before the completion of specific task-related actions. This allows both replacement variable initialization and task element evaluation to be done in the same task action.

The following table summarizes the replacement variables that are available for use in human tasks. It includes the initialization sequences for both replacement variable calculation and task element evaluation during the execution of a task action.

Table 4. Actions that initialize or change replacement variables

Replacement variable	Initialize action	Change action
htm:input.*[part|part\XPath|\XPath]*	Any action that creates or starts a task, setInputMessage method	Any action that creates or starts a task, setInputMessage
htm:output.*[part|part\XPath|\XPath]*	complete method or setOutputMessage method	complete method or setOutputMessage
htm:task.administrators	Any action that starts a task	Transfer an administrator work item, rerun a people query for the administrator role
htm:task.description	Any action that creates or starts a task	Update a ask with a new description
htm:task.displayName	Any action that creates or starts a task	Update a task with a new display name
htm:task.editors	Any action that starts a task	Transfer an editor work item, rerun a people query for the editor role
htm:task.instanceID	Any action that creates a task	None
htm:task.originator	Any action that creates a task	Transfer an originator work item
htm:task.owner	Any action that claims a task	Transfer an owner work item, any action that cancels the claim of a task
htm:task.potentialInstanceCreators	Any action that starts a task	Rerun a people query for the potential instance creator role
htm:task.potentialOwners	Any action that starts a task	Transfer a potential owner work item, rerun a people query for the potential owner role
htm:task.potentialStarters	Any action that creates a task	Transfer a potential starter work item, rerun a people query for the potential starter role
htm:task.property.*customPropertyName*	Any action that starts a task, or the task.setCustomProperty method	task.setCustomProperty method
htm:task.readers	Any action that starts a task	Transfer a reader work item, rerun a people query for the reader role
htm:task.starter	Any action that starts a task	Transfer a starter work item
htm:task.URLPrefix	Start up of the human task container	Change the human task container configuration in the administrative console
htm:task.URLPrefixAdmin	Start up of the human task container	Change the human task container configuration in the administrative console
htm:task.URLPrefixBPCExplorer	Start up of the human task container	Change the human task container configuration in the administrative console
htm:escalation(*escalationName*).receivers	The escalation is put into the escalated state	Transfer an escalation receiver work item for the specified escalation, rerun a people query for the escalation receiver role
htm:escalation.activationState	Create escalation	None
htm:escalation.description	Create escalation	None
htm:escalation.displayName	Create escalation	None
htm:escalation.expectedTaskState	Create escalation	None

Replacement variable	Initialize action	Change action
htm:escalation.instanceID	Create escalation	None
htm:escalation.property. *customPropertyName*	Create escalation, or the escalation.setCustomProperty method	escalation.setCustomProperty method
htm:escalation.receivers	The escalation is put into the escalated state	Transfer an escalation receiver work item for the current escalation, rerun a people query for the escalation receiver role
htm:escalation.URLPrefix	Start up of the human task container	Change the human task container configuration in the administrative console
htm:escalation.URLPrefixBPCExplorer	Start up of the human task container	Change the human task container configuration in the administrative console
wf:activity(*activityName*).editor	Create an inline task that belongs to the named activity	Transfer an editor work item for the named activity, rerun a people query for the editor role
wf:activity(*activityName*).owner	Create an inline task that belongs to the named activity	Claim, cancel claim, or transfer an owner work item for the named activity
wf:activity(*activityName*). potentialOwners	Create an inline task that belongs to the named activity	Transfer a potential owner work item for the named activity, rerun a people query for the potential owner role
wf:activity(*activityName*).reader	Create an inline task that belongs to the named activity	Transfer a reader work item for the named activity, rerun a people query for the reader role
wf:process.administrators	Start a process	Transfer a process administrator work item for the process or the process administrative task, rerun a people query for the administrator role
wf:process.readers	Start a process	Transfer a process reader work item for the process or the process administrative task, rerun a people query for the reader role
wf:process.starter	Start a process	None
wf:property.*customPropertyName*	Start a process	setCustomProperty method
wf:variable.[*variableName*]\ *messagePartName*[*XPath*]	The result of an activity, or the processInstance.setVariable method	The result of an activity, or the processInstance.setVariable method

Evaluation of task elements and their replacement variables at runtime:

Task elements can be queried by client applications to expose task-related information. The definition of these elements can contain replacement variables. For these variables to be replaced with values at runtime, certain conditions must be met.

For the variable to be replaced by a value, both of the following conditions must be met:

1. The variables are initialized before the task element is evaluated
2. The task element is evaluated before the query from the client application

3.

The following table describes which Human Task Manager replacement variables can be considered to be initialized before the completion of specific task-related actions. This allows both replacement variable initialization and task element evaluation to be done in the same task action.

Table 5. Task-related actions and the sequence of evaluation of the task elements and replacement variables

Task-related action	Sequence of evaluation

Create a task	1. Task element: People assignment for potential instance creators is evaluated
	   a. Replacement variable: htm:task.potentialInstanceCreators is initialized
	2. Task element: People assignment for potential starters is evaluated
	   a. Replacement variable: htm:task.potentialInstanceStarters is initialized
	   b. Replacement variable: htm:task.instanceID is initialized
	   c. Replacement variable: htm:task.displayName is initialized
	   d. Replacement variable: htm:task.property.*customPropertyName* is initialized
	3. Task element: The task priority is evaluated
	   a. Replacement variable: htm:task.originator is initialized
	4. Task element: The task description is evaluated
	   a. Replacement variable: htm:task.description is initialized

Table 5. Task-related actions and the sequence of evaluation of the task elements and replacement variables (continued)

Task-related action	Sequence of evaluation

Start a task	1. Task element: No elements are evaluated
	  a. Replacement variable: htm:input.*[part|part\XPath|\ XPath]* is initialized
	2. Task element: The duration until the task is overdue is evaluated
	3. Task element: The duration until the task expires is evaluated
	4. Task element: People assignment for the administrator role is evaluated
	  a. Replacement variable: htm:task.administrators is initialized
	5. Task element: People assignment for potential owners is evaluated
	  a. Replacement variable: htm:task.potentialOwners is initialized
	6. Task element: People assignment for editors is evaluated
	  a. Replacement variable: htm:task.editors is initialized
	7. Task element: People assignment for readers is evaluated
	  a. Replacement variable: htm:task.readers is initialized
	  b. Replacement variable: htm:task.starter is initialized
	8. Task element: The custom properties for the task are evaluated
	  a. Replacement variable: htm:task.property.*customPropertyName* is initialized
	9. Task element: The task type is evaluated
	10. Task element: The task priority is evaluated
	11. Task element: The task description is evaluated
	  a. Replacement variable: htm:task.description is initialized

An escalation is created	No task elements are evaluated. The following replacement variables are initialized: • htm:escalation.instanceID • htm:escalation.activationState • htm:escalation.expectedTaskState • htm:escalation.displayName • htm:escalation.description • htm:escalation.property.*customPropertyName*

An escalation is activated	1. Task element: The escalation duration is evaluated
	2. Task element: The escalation properties are evaluated
	  a. Replacement variable: htm:escalation.property.*customPropertyName* is initialized
	3. Task element: The escalation description is evaluated
	  a. Replacement variable: htm:escalation.description is initialized or changed

Table 5. Task-related actions and the sequence of evaluation of the task elements and replacement variables (continued)

Task-related action	Sequence of evaluation

An escalation is triggered	1. Task element: Escalation receivers are evaluated.
	
	    a. Replacement variable: htm:escalation.receivers is evaluated
	
	    Note that the definition can refer to the escalation receivers of another escalation, for example, it can include a variable like htm:escalation(otherEscalationName).receivers.
	
	2. Task element: For e-mail notifications, the e-mail subject and body are evaluated

A task reaches an end state	1. Task element: The duration until deletion of the task is evaluated

This table highlights the constraints that you must observe when specifying Human Task Manager replacement variables for task elements:

- A task element should only include replacement variables that are initialized before the task element is evaluated:
  - Replacement variables that are initialized due to an earlier task action are the replacement variables that are located in rows that are above the row that contains the task element.

    For example, the replacement variable %htm:task.originator% is initialized when a task is created. Therefore, it can be included in the definition of the task element "People assignment for potential owners" that is evaluated when the task instance is started.

  - Replacement variables that are initialized as part of the same task action as the task element evaluation, but before the task element evaluation is done are shown on the same row. As indicated in the table, for selected task actions (task creation, task update, escalation activation, escalation timer expiration) an evaluation order is defined for both task elements and replacement variables. Based on this sequence, you can see which of the replacement variables are initialized before a task element is evaluated.

    For example, a replacement variable %htm:input.\param1% is initialized when the task starts, but before any of the task elements are evaluated. Therefore it can be included in the definition of the task element "People assignment for potential owners" that is also evaluated during the start of the task instance. If an evaluation order is not given in a table row, no specific evaluation order is guaranteed for that row.

- A task element evaluation can imply the initialization of a corresponding replacement variable:
  - Some task elements have corresponding replacement variables that can be used by other task elements. For example, the task element "People assignment for potential owners" has a corresponding replacement variable %htm:task.potentialOwners%, that is initialized after the task element is evaluated. This means that task elements can be defined in terms of other task elements.

Replacement variables: Examples of usage patterns in task elements:

You can use replacement variables in many different ways. At runtime the values used for the variables can come from many sources. For example, they can

originate from previous staff resolutions, custom properties and, in the case of inline tasks, from the surrounding business process.

A task description includes runtime-specific information

Use this pattern for client applications that query task descriptions. These descriptions include replacement variables that are initialized when a task is created or started. This pattern requires the following definitions for the task elements in the task template.

Table 6. Task template definitions to include runtime-specific information in the task description

Task element	Definition
Task custom property	name: 'property1' value: 'a default value'
Task description	"This task instance has as ID: %htm:task.instanceID% as originator: %htm:task.originator% as administrators: %htm:task.administrators% as owner: %htm:task.owner% a custom property with the name 'property1' which is set to %htm:task.property.*property1*%

If you use the setCustomProperty method on a task instance, you can set an individual custom property for the task instance. When the task starts, the task description is evaluated, and this value is included in the description that is displayed in the client application.

Control the duration of task with a custom property

Use this pattern so that a client application can control the duration of a task instance. This pattern requires the following definitions for the task elements in the task template.

Table 7. Task template definitions to control the duration of a task

Task element	Definition
Task calendar	'Simple'
Task custom property	name: 'property1' value: '2days 3hours'
Duration until overdue	%htm:task.property.*property1*%

If you use the setCustomProperty method on a task instance, you can set a custom property for the task instance in a format that is allowed by the simple calendar. When the task starts, the duration until overdue is evaluated, and this value is inserted for the duration.

Control the people assignment of an inline task

Use this pattern to control the people assignment for a task instance based on the people assignment for a previous task in the business process. This pattern requires the following definitions for task elements in the task template.

Table 8. Task template definitions to control the people assignment of an inline task

Task element	Definition
People assignment for potential owners role	Users by User ID userId: %wf:activity(*activity1*).owner%

In the replacement variable, *activity1* is a human task activity in a business process that is in the claimed state. This means that the owner of the task is known. When the second task is started, the people assignment for the potential owners is evaluated. The owner of the first task is inserted as the parameter value in the people assignment expression.

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 9.

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 service-oriented architecture (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.

Stand-alone tasks have an autonomy setting of either peer or child. Stand-alone tasks with peer autonomy communicate with their partner components exclusively by 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 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 task components they have no SCA references or SCA interfaces and therefore cannot be wired to other service components. Instead they provide interfaces so that people can start them and work with them using the Human Task Manager APIs.

Inline tasks

Inline tasks are an integral part of the business process. Inline tasks can be to-do tasks, invocation tasks, or 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, 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 can emit their CEI and audit log events as business process activity events and human task events. Their subtasks and follow-on tasks emit events as human task events.

The following rules apply to inline tasks:

- To-do tasks are human task activities in a process. They share the same state, but the human task activity does not reflect the forwarded state or 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.
  - Inline to-do tasks do not have their own expiration, the expiration is defined on the corresponding human task activity. When the human task activity expires, the to-do task is terminated. Updating or rescheduling of inline to-do tasks is supported by both the Human Task Manager and Business Flow Manager APIs. If the Human Task Manager API is used, the request is forwarded to the human task activity.
  - 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 started using the Human Task Manager API, their life cycle is independent of the process regardless of how they were created, and their results can be displayed even after the process is deleted.
  - Instances of inline human tasks can be migrated with the process instance they are related to.
- To-do and invocation task descriptions, display names, and documentation support only one language.
- Inline invocation tasks can be modeled with values for both the duration until expiration and the duration until deletion. These settings are available only if the tasks are created using the Human Task Manager API. These durations can be updated before the task starts, and rescheduled after the task has started.
- 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 API documentation for the HumanTaskManager interface in the com.ibm.task.api package, and the information on which roles are authorized to make specific update actions on tasks.

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 scope reader and the scope administrator authorization is inherited by all of the activities in the scope.

Note: If process administration is restricted to system administrators, then instance-based administration is disabled. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. In addition, reading, viewing, and monitoring a process instance or parts of it can only be performed by users in the BPESystemAdministrator or BPESystemMonitor roles. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

Relationship between human tasks and business processes

Inline tasks know the process they are related to, and the process knows about its inline tasks. With stand-alone to-do tasks this relationship can be defined with child autonomy. A child task that is directly instantiated by an invoke activity of a business process, participates in the life cycle of that business process. That is, life cycle operations, such as termination or deletion, are propagated from the business process to its child tasks. When the activity expires, the to-do task is terminated.

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 or pick 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 or pick 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, an administration task is created at runtime. This task ensures the default authorization for the business process; the process starter becomes the only administrator of the process, and readers are not assigned 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.

Note: If process administration is restricted to system administrators, then instance-based administration is disabled, and no administration task instances are created. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

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 a subtask when it is started, the subtask can also inherit authorization roles from its parent task. The inherited roles depend on the role propagation settings that are defined for the subtask in WebSphere Integration Developer:

All   The readers, editors, originator, potential owners, and owner of the parent task become readers of the subtask and its escalations

All or Administrator
> Administrators of the parent task become administrators of the subtask and its escalations

Life cycle considerations

When the first subtask is started, the parent task enters the waiting-for-subtask substate. It remains in this substate until all of the subtasks reach 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 a parent task 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.

The following operations can be performed on subtasks:
- 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.

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 task. Auto-deletion settings are ignored for tasks that are started as subtasks. Subtasks are deleted when their parent task is deleted or restarted. 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.
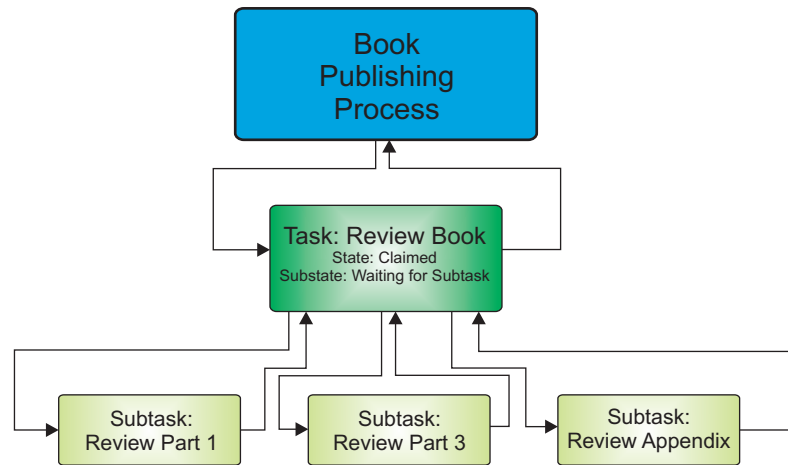
In a book publishing process, the "Review Book" task is claimed by Linda. She realizes that the book is too large for her to review alone, and specialized knowledge is required for some parts of it. She decides to deviate from the standard publishing process, and assigns parts of her task to some of 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 the subtask is complete, the parent task leaves the waiting-for-subtask substate and it can be completed.

Related concepts

"Authorization roles for human tasks" on page 105
Actions that you can take on human tasks depend on your authorization role. This role can be a system-level Java EE role or an instance-based role. Role-based authorization requires that administration and application security is enabled for the application server.

"Life cycle of human tasks" on page 86
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, an 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. You can start a follow-on task from a to-do task or a collaboration task that has the **supportsFollowOnTask** attribute set to true. A follow-on task can have follow-on tasks of its own resulting in a chain of tasks.

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 first 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 and the changes are copied to the parent task.

Authorization considerations

Follow-on tasks inherit the authorization roles from the predecessor task.
- The readers, editors, originator, potential owners, 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 and its escalations

Life cycle considerations

When the follow-on task is started, the predecessor task enters the forwarded state. A chain of follow-on tasks is handled as though it were a single task. This means that you can perform some life-cycle operations on any task in the chain and the correct behavior is applied. For example:
- When any task in the chain is suspended, the entire chain is suspended. Each task is put into the suspended substate.

- A suspended chain of follow-on tasks can be resumed by any task in the chain.
- If a task in the chain escalates, all of the follow-on tasks in the chain are escalated.
- When any task in the chain is terminated, the entire chain is terminated.
- When the first task in the chain expires, the last task in the chain is put into the expired state.

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 first task in the chain.
- Top-level tasks and follow-on tasks can be suspended and resumed. This action suspends and resumes all of the tasks in the chain.
- 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.
- Follow-on tasks are deleted when their parent task is deleted or restarted. 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.
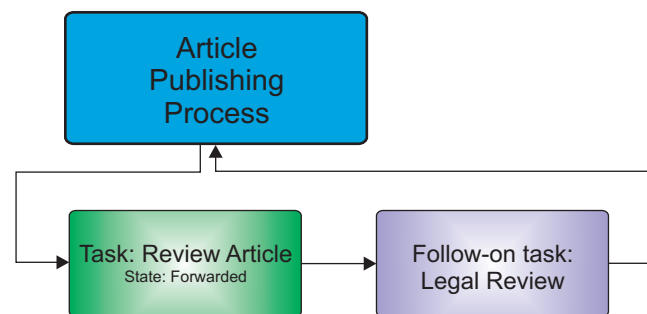
In an article publishing process, the "Review Article" task is claimed by John. He is empowered by the process to review and approve the legal aspects of articles as well. However, this article describes the collaboration with a competitor product, and is thus very sensitive from a legal point-of-view. 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 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.

To-do tasks and collaboration tasks with parallel ownership

Tasks with parallel ownership allow potential owners to work simultaneously on the task. A common example of parallel ownership is when a set of potential owners need to approve a to-do task in a business process. Parallel ownership can be specified for to-do tasks and collaboration tasks.

When a task with parallel ownership is started, a subtask for each potential owner is created and started, and the parent task goes into the running state. The subtasks are always collaboration tasks. The input message and all other relevant information for the parent task are copied to each subtask.

After the subtasks are started they go into either the ready state or, if automatic claim is specified for the parent task, the claimed state. When the subtasks are created, the parent task goes into the waiting-for-subtask substate. The subtasks then go through the normal life cycle of a collaboration task; the parent task remains in the waiting-for-subtask substate until all of its subtasks reach an end state. If the completion condition for the parent task becomes true, all of the subtasks, which are not yet in an end state, are terminated.

Because parent tasks do not have an owner, you cannot use the API operations, such as claim or cancelClaim on them. If the parent task is modeled so that its subtasks are claimed automatically, the subtasks are automatically assigned to each of the potential owners.

Authorization considerations

The following restrictions apply to the people assignment criteria that you assign to the task with parallel ownership so that a subtask can be created for each potential owner:
- Do not use the Nobody or Everybody people assignment criteria
- Do not use people assignment criteria that return a group, for example, Group

The subtasks that are created have the following authorization roles:
- The administrator of the task with parallel ownership becomes the administrator of each of the subtasks
- The person who starts the task with parallel ownership becomes the originator of each of the subtasks
- One potential owner

In addition, the subtask can also inherit authorization roles from the task with parallel ownership. The inherited roles depend on the role propagation settings that are defined for this task in WebSphere Integration Developer:

All The readers, editors, originator, potential owners, and owner of the parent task become readers of the subtask and its escalations

All or Administrator
Administrators of the parent task become administrators of the subtask and its escalations

Completion conditions

Generally, a parent task waits in the waiting-for-subtask substate until all of its subtasks are in an end state. However, in certain parallel ownership situations, you might want the parent task to finish without waiting for all of the subtasks to enter an end state. For example, if the subtasks are for the approval of a document, you might want the parent task to end even if not all of the subtask owners approve the document. To enable this type of scenario, you can specify completion conditions for the task with parallel ownership in WebSphere Integration Developer. The following types of completion conditions are available:

XPath-based completion condition
This completion condition can exploit both the completion condition functions and result construction functions. The condition is evaluated before the subtasks are created and after each subtask enters the finished, expired, terminated, or failed state. It must evaluate to true for a task with parallel ownership to finish.

For example, the completion condition for a task with parallel ownership that should finish when at least 50% of the subtask owners have provided their data might look similar to the following code snippet:

```
tel:getCountOfFinishedSubtasks() div tel:getCountOfSubtasks() > 0.5
```

Calendar-based completion duration
A duration that specifies when the task with parallel ownership should finish at the latest. The duration syntax is determined by the calendar that is specified for the task definition.

If one of the completion conditions applies, the task with parallel ownership finishes and the aggregated result for all of the subtasks is constructed. If subtasks exist that are not yet finished, they are automatically terminated.

Result construction

The result of the task with parallel ownership is constructed by aggregating the results of its subtasks. You can use XPath expressions in the task definition to specify how fields in the output message of the task with parallel ownership are to be filled based on the outcome of the subtasks. The result for a specific field consists of the following attributes that are defined by an XPath expression identifying a value in the context of the output message of the task.

part
This attribute identifies the part in the output message that contains the location field that is to be used. This field must be omitted for message definitions that use the document literal wrapped binding style.

location
This attribute identifies the following fields:
- The field in the output message of each of the subtasks that is the source for result aggregation

- The field in the output message of the task with parallel ownership that is the target for the result of the aggregation

condition

This attribute specifies that the subtask field is relevant for result construction. The field is identified by the location attribute. If a subtask field is not relevant, it is ignored when the results are constructed.

aggregationFunction

This attribute defines how the values of the subtask fields are combined into one aggregated result. The fields are identified by the part, the location, and the condition attributes. The aggregated result is stored in the output message of the task with parallel ownership, in the field identified by the location attribute.

Example of a TEL definition of a task with parallel ownership

The following example shows the TEL definition of a task with parallel ownership:

```
<tel:result>
   <tel:aggregate location="/reviewresult" function="tel:and()"/>
   <tel:aggregate location="/reviewcomments"
                   condition="/reviewresult=true()"
                   function="tel:concatWithDelimiter('|')"/>
</tel:result>
```

In this example, result aggregation is specified for the `reviewresult` and `reviewcomments` fields in the output message. The location `/reviewcomments` specifies that the corresponding field in the output message of the subtask is used as the source for aggregation. The XPath indicator "/" denotes the root of the output message definition. The `/reviewresult=true()` condition specifies that a subtask is considered only if the value of the `reviewresult` field in its output message is set to true. The aggregation function specifies that the values of the qualifying output messages are concatenated into an aggregate String using the specified delimiter.

Related concepts

"State transition diagrams for to-do tasks" on page 86
To-do tasks are created automatically by a client application, or calling component. They 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). During the life cycle of a to-do task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

"State transition diagrams for collaboration tasks" on page 90
Collaboration tasks support people when they perform work for other people. During the life cycle of a collaboration task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

XPath extension functions for human tasks with parallel ownership

For human tasks with parallel ownership, you can use XPath extension functions to control how individual responses are combined to produce a result.

In addition to standard XPath functions described in the XPath 1.0 specification (http://www.w3.org/TR/xpath), you can use the following functions, called extension functions, in your XPath expressions and conditions.

XPath extension functions for aggregating the results of human tasks with parallel ownership

Use result aggregation functions for human tasks with parallel ownership to specify how each field of the output message should be filled based on the outcome of the individual subtasks.

Aggregation functions for fields of type string

The following functions aggregate the output messages for all of the subtasks of the parallel-ownership task and return values of type String. The output messages are converted to strings according to the XPath string function before the function runs.

Table 10. Aggregation functions that return string values

Function name	Parameters	Description
concat	None	This function concatenates the values of the output messages for all of the subtasks. Output messages are converted to strings according to the XPath string function. If no output messages exist, an empty string is returned.
concatWithDelimiter	Delimiter	This function concatenates the values of the output messages for all of the subtasks. Each of the values in the concatenated output is separated by the specified delimiter string. If no output messages exist, an empty string is returned.
leastFrequentOccurence	None	This function returns the value that occurs least frequently. If no output messages exist or more than one message has the least frequently occurring value (a tie), an empty string is returned.
mostFrequentOccurence	None	This function returns the value that occurs most frequently. If no output messages exist or a tie occurs, an empty string is returned.
voteOnString	Percentage	This function returns the value that occurs most frequently if its occurrence is above the specified percentage, and there is no tie. The result of the vote is determined in the following way. 1. The percentage is multiplied by the total number of subtasks for the task with parallel ownership to determine the minimum number of occurrences. 2. The string values are filtered so that only those values which exceed the minimum number of occurrences determined in step 1 remain. 3. From the values that qualify from step 1, the value that occurs most frequently is returned. If no output messages exist or a tie occurs, an empty string is returned.

Aggregation functions for fields of type boolean

The following functions aggregate the output messages for all of the subtasks of the parallel-ownership task and return values of type Boolean. The output messages are converted to Boolean values according to the XPath Boolean function before the function runs.

Table 11. Aggregation functions that return Boolean values

Function name	Parameters	Description
and	None	This function returns the conjunction of all of the values. The result is true if all of the values are true. If at least one of the values is false or no output messages exist, the result is false.
or	None	This function returns the disjunction of all of the values. The result is true if one of the values is true. If all of the values are false or no output messages exist, the result is false.
vote	Percentage	This function returns the Boolean value that occurs most frequently if its occurrence is above the specified percentage, and there is no tie. The result of the vote is determined in the following way. 1. The specified percentage and the count of the subtasks result in a minimum value that must be exceeded by the occurrence of a single Boolean value. 2. From the values that qualify from step 1, return the value that occurs most frequently. If no output messages exist or a tie occurs, false is returned.

Aggregation functions for numeric, duration, and dateTime values

The following functions aggregate output message fields for all of the subtasks of the parallel-ownership task and return the value as an object. They can be applied to output message fields of the following types:

Numeric
> xsd:decimal, xsd:float, xsd:double, xsd:integer, byte, xsd:int, xsd:long, xsd:short

Calendar
> xsd:duration, xsd:dateTime

Table 12. Aggregation functions that return objects

Function name	Parameters	Description
avg	None	This function returns the average value of the numeric, duration, or dateTime values. The calculation mode and the return type are determined by the first message in the set that has a value. The type of the returned value is the same as the type of the nodes in the node set. If no output messages exist, null is returned and the output for this field is not set.
max	None	This function returns the maximum value of the numeric, duration, or dateTime values. The comparison mode and the return type are determined by the first message in the set that has a value. If no output messages exist, null is returned and the output for this field is not set.

Table 12. Aggregation functions that return objects (continued)

Function name	Parameters	Description
min	None	This function returns the minimum value of the numeric, duration, or dateTime values. The comparison mode and the return type are determined by the first message in the set that has a value. If no output messages exist, null is returned and the output for this field is not set.
sum	None	This function returns the sum of the numeric or duration values. The dateTime type is not supported. The calculation mode and the return type are determined by the first message in the set that has a value. For duration values, the duration time in milliseconds is used to calculate the sum. The algorithm converts all of the numeric values to doubles, and then it adds the resulting values. All of the calculations are done based on doubles. The return type is the same as the type of the output messages. If no output messages exist, 0 is returned for numeric values and P0S for duration values.

Completion conditions for XPath functions for human tasks with parallel ownership

A completion condition defines when the set of subtasks that are associated with a parent task is considered to be complete. It is specified as a boolean XPath expression, which can refer to output data from finished subtasks, or other data obtained from helper functions, for example, the number of subtasks.

Node-set functions for accessing the output data for a set of subtasks

You can use the following functions to access the output data of finished subtasks. To aggregate the result, you can use the returned node-set to call one of the XPath extension functions.

node-set tel:getSubtaskOutputs(string partName, string locationPath)
> This function returns a node set of simple-typed or complex-typed elements that is constructed from the output documents of the subtasks in the routing pattern.

> **partName**
>> This string parameter contains the name of the part in the output document for a subtask.

> **locationPath**
>> This string parameter contains the location path in the output document for a subtask. The value of this parameter determines the number of elements returned in the node set:

>> **maxOccurs=1**
>>> The function returns a node set with one node per completed subtask.

>> **maxOccurs>1**
>>> The function returns a node set that combines the node set of each leaf.

>> **minOccurs=0**
>>> The function returns a node set that can contain fewer

elements than the number of completed subtasks because the node set can contain only non-empty elements.

node-set tel:getSubtaskOutputs(string locationPath)

This function returns the node set of simple-typed or complex-typed elements that is constructed from the output documents of the subtasks in the routing pattern. The behavior of this function is identical to **tel:getSubtaskOutputs(string partName, string locationPath)**. This function (**tel:getSubtaskOutputs(locationPath**) is only applicable to output documents that use the document/literal wrapped binding style. All other formats for the output documents (multipart messages, but also non-document/literal wrapped) must use the **tel:getSubtaskOutputs(partName, locationPath)** function.

Helper functions

You can use the following functions to access information about the subtasks of a task with parallel ownership. The result of these functions can be used in XPath expressions for completion conditions only.

number tel:getCountOfSubtasks()

This function returns the number of subtasks that were created for the routing pattern.

number tel:getCountOfCompletedSubtasks()

This function returns the number of subtasks for the routing pattern that are complete.

number tel:getCountOfFinishedSubtasks()

This function returns the number of subtasks for the routing pattern that are in an end state.

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 during modeling or, dynamically, when you create a task at runtime.

Escalations are activated at a certain task state. The task will normally only be escalated because the expected task state has not been reached when the time limit for the escalation expires. However, it can also be triggered manually any time before the time limit expires by a user who has the appropriate authority. The time limit for the escalation is interpreted by the calendar that is specified for the task. You can specify multiple escalations (or escalations chains) that have the same activation state. An escalated task is put into the escalated substate.

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

To define an escalation for a to-do task or a collaboration task that is in the claimed state when the task is not completed in time, you must specify that the expected task state is ended.

Running

To define an escalation for an invocation task that is in the running state when the invoked service does not return in time, you must specify that the expected task state is ended. If a task that is in the running state has an escalation defined for it, authorized users can trigger the escalation manually before the defined time limit expires.

Subtask started

For to-do tasks or collaboration tasks, you can define an escalation that starts when the first subtask is started. This state can be used for tasks with single ownership, but is mostly used for tasks with parallel ownership. For these tasks, the expected end state must be 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 individual user IDs.

You can define the escalation so that the priority of the task is automatically increased either for the first iteration only, or for every iteration of the escalation.

Life cycle of an escalation

The following diagram shows the state transitions that can occur during the life cycle of an escalation.
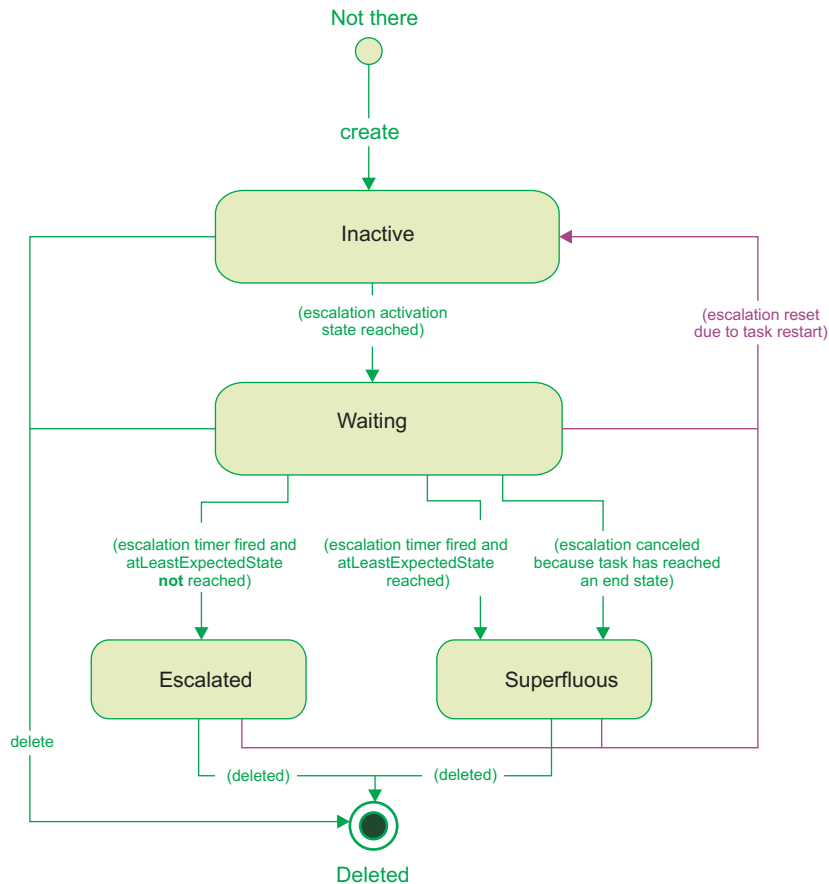
- When a task is created, any predefined escalations are created and are put into the inactive state.
- When the task reaches the activation state for the escalation, the escalation is put into the waiting state and the timer is started.
- A waiting escalation becomes escalated, the escalation action is performed, and the associated task is put into the escalated substate if one of the following occurs:
  - The task has not yet reached the expected state and an authorized user triggers the escalation manually.
  - The task has not yet reached the expected state and the timeout is triggered.
- A waiting escalation becomes superfluous, and is deleted if one of the following occurs:
  - The task has reached the expected state and an authorized user triggers the escalation manually.
  - The task has reached the expected state and the timeout is triggered.
  - The task has reached an end state and the escalation is canceled.
- You can change the escalation duration and repetition duration.

The escalation action can be performed repeatedly.

Chained escalations

A chain of escalations is a series of escalations with the same activation state and expected end state of the task. These escalations are processed sequentially so that only one escalation is waiting at any one time. A chain of escalations is activated

when the task reaches the activation state for the first escalation in the chain. 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 relative 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%*
- For inline tasks, 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. If the variable resolution fails, the duration is not set correctly. A CWTKE0038E error appears in the `SystemOut.log` file and your escalation is not set up.

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. For chained escalations, the duration of each escalation is evaluated when it is started.	The task activation state of the escalation.
Escalation repetition	The escalation is raised.	Escalated

Related concepts

"Changes to the timing of an escalation at runtime" on page 59
Sometimes a business situation requires that you change the timing of an
escalation that was specified when the escalation was defined. The escalation state
determines which of these times you can change, and when these actions can be
taken. You can use the update method of the Human Task Manager API to modify
the appropriate escalation property. You can also use the Tasks List widget or the
Escalations List widget in Business Space to override the scheduled escalation time,
and start an escalation immediately.

"Replacement variables in human tasks" on page 61
Replacement variables are used in the definitions of human tasks to refer to a
value of an element that is resolved at runtime. These variables represent task and
process-related data, such as people assigned to a task or custom properties for the
task. This data is available at runtime for all or part of the life cycle of a task
instance.

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, an 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.

Related concepts

"Subtasks" on page 71
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.

State transition diagrams for to-do tasks

To-do tasks are created automatically by a client application, or calling component.
They 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).
During the life cycle of a to-do task, certain interactions are possible only in certain
task states, and these interactions, in turn, influence the state of the task.

The state transitions that occur during the life cycle of the task also depend on
whether the task has single ownership or parallel ownership.

To-do tasks with single ownership

The following diagram shows the state transitions that can occur during the life
cycle of to-do tasks that have a single owner. For stand-alone to-do tasks, it
assumes that the autonomy attribute of the task is set to peer.
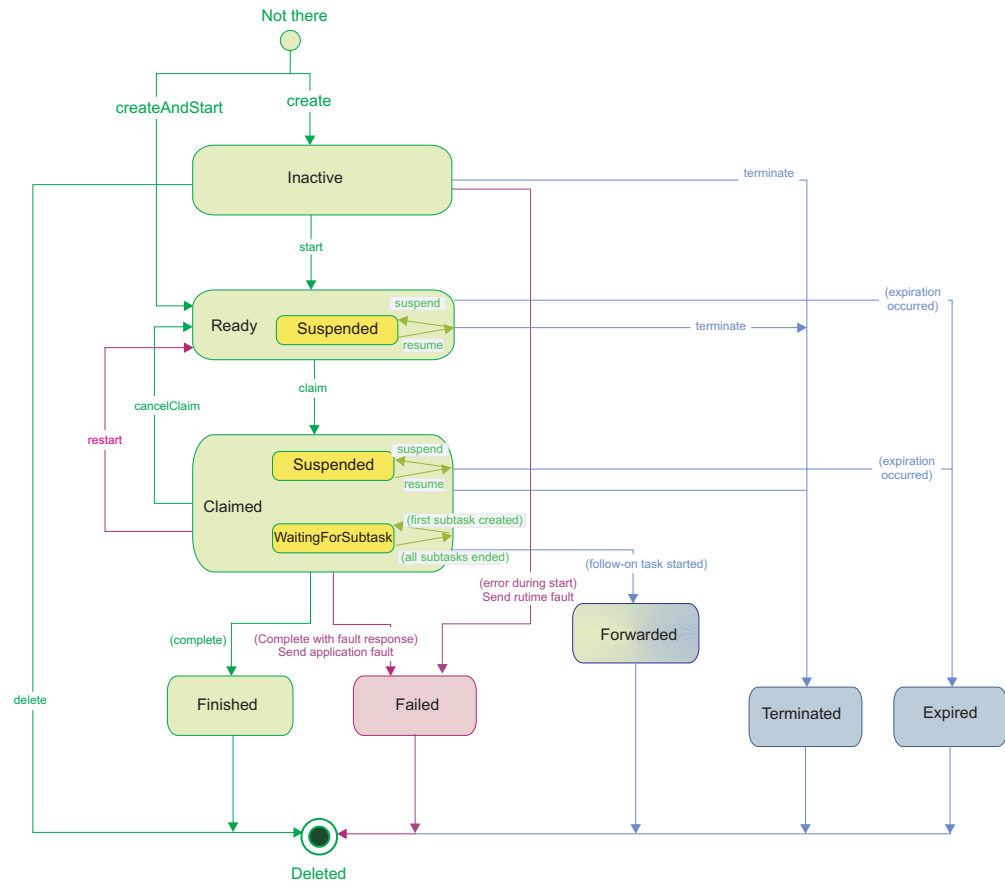
After the task is created, it is put into the inactive state. In this state you can update task properties or set custom properties, for example, to change the duration until the task is due, expires, or is deleted. To work on a to-do task, it must be started.

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 is escalated if it is not claimed or completed in time, or an authorized user triggers the escalation manually. The task 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.

- To reschedule a task when it is due, expires, or is deleted, the originator, starter or administrator of the task can edit the appropriate property for the duration or point in time.

- 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 the parent task is suspended, all of its subtasks are also suspended.
- If they want to delegate the completion of the work to someone else, they can create, for example, a collaboration task as a follow-on task to complete the work. 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. Note that if the claim of the task is cancelled, this action does not affect the due time or expiration time of the task, or the timing of the escalations.

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

- The task is escalated if it is not completed in time or if it waits too long for subtasks to complete. An authorized user can also trigger the escalation manually. The task is put into the escalated substate.
- 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.
- To reschedule a task when it is due, expires, or is deleted, the originator, starter or administrator of the task can edit the appropriate property for the duration or point in time.
- The task can be terminated manually using the terminate action. This is a state change that ends the task.
- The task can be restarted. The task is put back into the ready state. If the task has substates, they are cancelled. Escalations associated with the task are reset to the inactive state, and begin their normal life cycle. If the task has subtasks, these are terminated and deleted.

When the work is finished on a task, the owner completes the task. 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 and follow-on tasks are also deleted.

The forwarded state indicates that work is still required on the follow-on task. Automatic deletion of the parent task applies as soon as the follow-on task reaches an end state. Without automatic deletion, both the parent and the follow-on task remain in their states until the parent task is explicitly deleted. When the parent task is deleted, the follow-on task is also deleted.

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.

To-do tasks with parallel ownership

The following diagram shows the state transitions that can occur during the life cycle of to-do tasks with parallel ownership.

The parent task cannot be claimed or manually completed. The parent task goes into the running state and stays there until its completion condition becomes true, or expiration is reached.

Related concepts

"To-do tasks and collaboration tasks with parallel ownership" on page 76
Tasks with parallel ownership allow potential owners to work simultaneously on the task. A common example of parallel ownership is when a set of potential owners need to approve a to-do task in a business process. Parallel ownership can be specified for to-do tasks and collaboration tasks.

"Changes to the expiration, deletion, and due times for tasks at runtime" on page 57
Sometimes a business situation requires that you change the due, expiration, or deletion time that was originally defined for a task. The task state determines which of these times you can reschedule, cancel, and start at runtime, and when these actions can be taken. You can use Business Process Choreographer Explorer to change these times, or you can use the update method of the Human Task Manager API to modify the appropriate task property.

State transition diagrams for collaboration tasks

Collaboration tasks support people when they perform work for other people. During the life cycle of a collaboration task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

The state transitions that occur during the life cycle of the task also depend on whether the task has single ownership or parallel ownership.

Collaboration tasks with single ownership

The following diagram shows the state transitions that can occur during the life cycle of collaboration tasks that have a single owner.
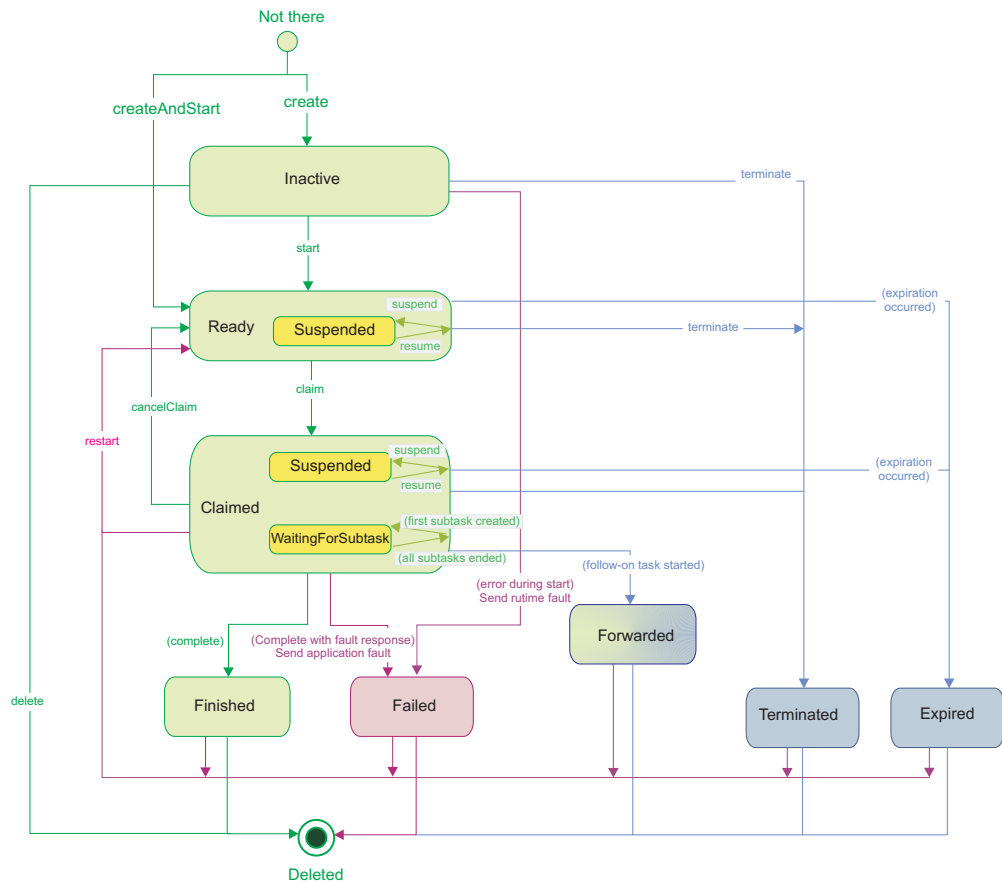
After the task is created, it is put into the inactive state. In this state it cannot be claimed, but you can update task properties or set custom properties. To work on a collaboration task, it must be started.

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 is escalated if it is not claimed or completed in time, or an authorized user triggers the escalation manually. The task 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.
- To reschedule a task when it is due, expires, or is deleted, the originator, starter or administrator of the task can edit the appropriate property for the duration or point in time.
- The task can be terminated manually using the terminate action. This state change ends the task.
- The task can be restarted. If the task is suspended, the suspended substate is cleared. If the task is escalated, the escalated substate is cleared. If the task has escalations, all of the escalations are put back into the inactive state and all of the running escalations are canceled. If the expiration timer is set, it is canceled and restarted, and the due time is recalculated. If the task is waiting for subtasks, the waiting-for-subtask substate is cleared, and the subtasks are deleted.

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 create subtasks to delegate parts of the work to other people. 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 the parent task is suspended, all of its subtasks are also suspended.
- If they want to delegate the completion of the work to someone else, they can create, for example, a collaboration task as a follow-on task to complete the work. 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. Note that if the claim of the task is cancelled, this action does not affect the due time or expiration time of the task, or the timing of the escalations.

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 to complete. An authorized user can also trigger the escalation manually. The task is put into the escalated substate.

- 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.
- The task can be restarted. The task is put back into the ready state. If the task has substates, these are cancelled. Escalations associated with the task are reset to the inactive state, and begin their normal life cycle. If the task has subtasks, these are terminated and deleted.

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 and follow-on tasks are also deleted.

The forwarded state indicates that work is still required on the follow-on task. Automatic deletion of the parent task applies as soon as the follow-on task reaches an end state. Without automatic deletion, both the parent and the follow-on task remain in their states until the parent task is explicitly deleted. When the parent task is deleted, the follow-on task is deleted as well.

A task in one of the end states can be restarted, if it is not a follow-on task of a to-do task. The task is put back into the ready state. Escalations associated with the task are cancelled and put input inactive state, and the deletion timer is also cancelled.

Collaboration tasks with parallel ownership

The following diagram shows the state transitions that can occur during the life cycle of collaboration tasks with parallel ownership.

The parent task cannot be claimed or manually completed. The parent task goes into the running state and stays there until its completion condition becomes true, or expiration is reached.

Related concepts

"To-do tasks and collaboration tasks with parallel ownership" on page 76
Tasks with parallel ownership allow potential owners to work simultaneously on the task. A common example of parallel ownership is when a set of potential owners need to approve a to-do task in a business process. Parallel ownership can be specified for to-do tasks and collaboration tasks.

"Changes to the expiration, deletion, and due times for tasks at runtime" on page 57
Sometimes a business situation requires that you change the due, expiration, or deletion time that was originally defined for a task. The task state determines which of these times you can reschedule, cancel, and start at runtime, and when these actions can be taken. You can use Business Process Choreographer Explorer to change these times, or you can use the update method of the Human Task Manager API to modify the appropriate task property.

Related tasks

"Processing to-do tasks or collaboration tasks" on page 570
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.

State transition diagrams for invocation tasks

Invocation tasks support people when they invoke services. During the life cycle of an invocation task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

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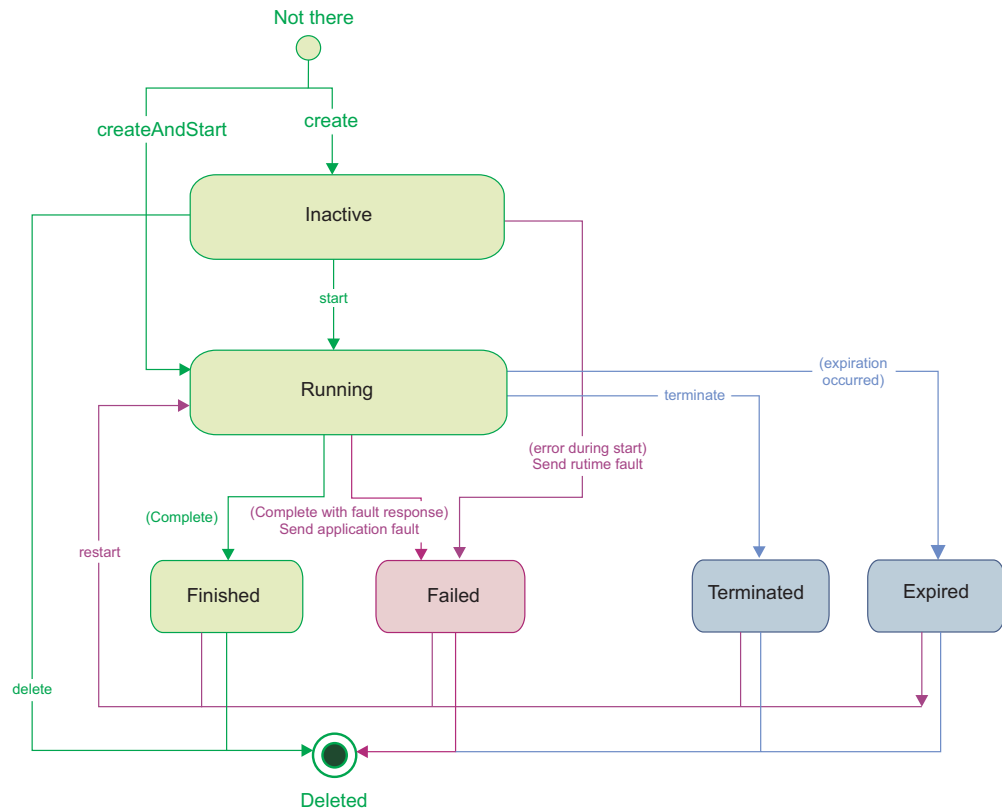
After creation, the task reaches the inactive state. In this state, you can update task properties, or set custom properties. To invoke the service, the task must be started. It can be started by the originator or one of the potential starters.

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.

A task in one of the end states can be restarted. The task is put back into the running state. Escalations associated with the task are cancelled and the deletion timer is also cancelled.

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.
- 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 started using the Human Task Manager API, their life cycle is independent of the process regardless of how they were created, and their results can be displayed even after the process is deleted.
- Regardless of how an inline invocation task is started, the due time for it can be rescheduled.
- Inline invocation tasks can be modeled with values for both the duration until expiration and the duration until deletion. These settings are available only if the tasks are created using the Human Task Manager API. These durations can be updated before the task starts, and rescheduled after the task has started.

State transition diagrams for administration tasks

Administration tasks support people in administering business processes and their activities. During the life cycle of an administration task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

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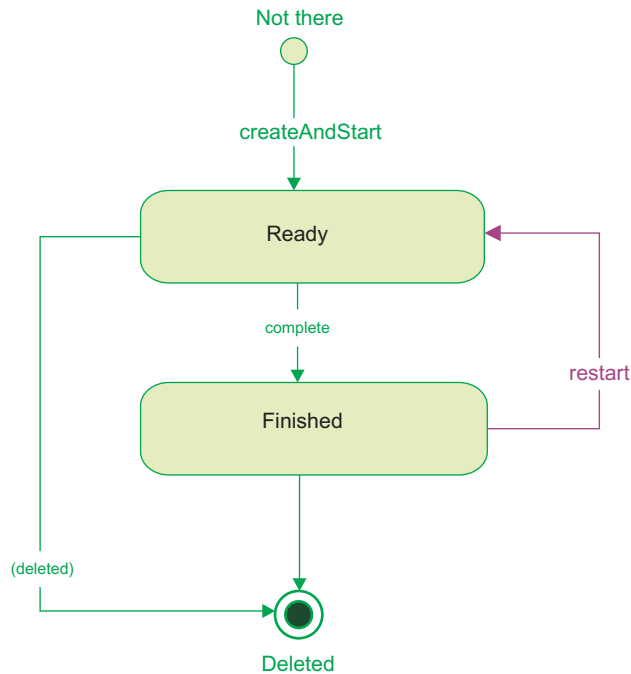
Business Flow Manager creates and starts an administration task implicitly in a single transaction. The inactive state is therefore not visible externally, and the task directly reaches the ready state.

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.

Note: If process administration is restricted to system administrators, then instance-based administration is disabled, and no administration task instances are created. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

How task states in Business Process Choreographer relate to the task status in Business Space

Business Process Choreographer includes the concept of a *task state*, which is defined as a stage in the life cycle of a task. For example, a task can be in the inactive or running state. The same concept in the Human Task Management widgets in Business Space powered by WebSphere is called the *task status*. In addition, the names and number of the individual task states differ in Business Process Choreographer and in Business Space.

The following table shows how the task states that are available in Business Process Choreographer map to the task status that is shown in Business Space.

Table 13. Mapping of task state in Business Process Choreographer to task status in Business Space

Task state in Business Process Choreographer	Task type	Task status in Business Space
Inactive	All task types	Not active
Ready	Collaboration and to-do tasks	Available
Running	Invocation tasks, collaboration and to-do tasks with parallel ownership	In progress
Finished	All task types	Successful
Failed	All task types	Failed
Terminated	All task types	Cancelled
Claimed	Collaboration and to-do tasks with single ownership	In progress
Terminating	All task types	Cancelled
Failing	All task types	Failed
Expired	All task types	Expired
Forwarded	Collaboration and to-do tasks with single ownership	Forwarded
Skipped	Inline invocation and to-do tasks	Skipped
Stopped	All types of inline tasks	Stopped

Scenarios for invoking tasks

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

Invocation of 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, a Web service implementation, a JMS message implementation, and a REST implementation. The API methods are similar for all implementations, but differ in their functional scope.

Invocation of to-do tasks as SCA service components

A stand-alone to-do task represents a Service Component Architecture (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 that connect 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 the 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.

Invocation of 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.

Invocation of an SCA service with 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 in the same way as the task was invoked.

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.

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.

"Scenario: stand-alone invocation tasks that support the asynchronous invocations of services" on page 100
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.

"Scenario: stand-alone invocation tasks that support asynchronous and synchronous invocations of services" on page 102
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.

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 97
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)

Qualifier type: attribute type	Value
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:

- *: use global if you use transactions settings, and local if you use activity session settings.
- **: if the transaction is set to global, only the transaction settings are used
- ***: 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) global
Interface qualifier**: JoinTransaction	false (default) 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.
- **: if the transaction is set to global, only the transaction settings are used
- ***: 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 97
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:	
• *: use global if you use transactions settings, and local if you use activity session settings.	
• **: if the transaction is set to global, only the transaction settings are used	
• ***: 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) global
Interface qualifier**: JoinTransaction	false (default) 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.	
• **: if the transaction is set to global, only the transaction settings are used	
• ***: 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 97
The various ways in which tasks can be invoked is described here.

Authorization and people assignment for human tasks

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 Java EE 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 Java EE role or an instance-based role. Role-based authorization requires that administration and application security is enabled for the application server.

Related concepts

"Subtasks" on page 71
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.

Java EE authorization roles for human tasks
System-level Java EE roles are set up when Human Task Manager is configured. The authority level implied by these roles is valid for all tasks and escalations.

The following Java Platform, Enterprise Edition (Java EE) 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 authorization roles for human tasks
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 for stand-alone tasks, the originator becomes the potential starter. For inline invocation 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 specified, then all users are considered to be members of this role. If people 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.

Note: If process administration is restricted to system administrators, then instance-based administration is disabled. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. In addition, reading, viewing, and monitoring a process instance or parts of it can only be performed by users in the BPESystemAdministrator or BPESystemMonitor roles. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

Task kinds and instance-based authorization roles

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 for a task.

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, scope, or activity					
2. This role also has authorization for read operations on the administered process, scope, or activity					
3. This role has authorization for read operations on the corresponding task					

Note: If process administration is restricted to system administrators, then instance-based administration is disabled, and no administration task instances are created. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

Task 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 object, for example, human task or business process, 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 from the people directory, for example, for composing e-mails. People assignment criteria are also used during runtime when task models are created programmatically.

You can use people assignment criteria definitions 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 people assignment criteria.

   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.

Replacement variables in people assignment criteria definitions

You can use replacement variables 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, a people assignment criteria definition might contain the `%htm:input.\name%` replacement variables as a parameter: 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 variable with the task input message value.

Related concepts

"Replacement variables in human tasks" on page 61
Replacement variables are used in the definitions of human tasks to refer to a value of an element that is resolved at runtime. These variables represent task and process-related data, such as people assigned to a task or custom properties for the task. This data is available at runtime for all or part of the life cycle of a task instance.

People resolution

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

People directories for use with Business Process Choreographer

People directories store user information that is used for resolving queries that assign people to human tasks.

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.

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.

Transformation of 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 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 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
- `Linux`   `UNIX`   On Linux® and UNIX® platforms, these files are in the *install_root*/ProcessChoreographer/Staff directory.
- `Windows`   On Windows® platforms, these files are in the *install_root*\ ProcessChoreographer\Staff directory.

People queries for a specific people directory provider

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 use this set of 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.

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.

To limit the substitution in time, you can specify a start and end point. If only a start point is specified, then the user will be considered to be absent from the specified starting point.

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

No substitution (default)
     The set of users remains unchanged

Replace absent users with their substitutes
- 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.

Prefer present users
- 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.

Default people assignments and inheritance rules

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.

Inheritance rules are applied to automatically assign people to a particular role, based on the fact that they are already assigned to another role. Inheritance rules effectively add users to a role in addition to the users that are determined by people assignment. These rules 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
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.

People assignment criteria and people query results

A people assignment criteria is associated with a task authorization role. 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 so that people can be assigned to the task.

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 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.

Shared 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. Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.

2. In the **Additional Properties** section, click **Custom Properties** and change the value of the **Staff.PostProcessorPlugin.EnableResultSharing** custom property to `true`, then save your changes

3. Restart the server or cluster to make the changes effective.

Related concepts

"Replacement variables in human tasks" on page 61
Replacement variables are used in the definitions of human tasks to refer to a value of an element that is resolved at runtime. These variables represent task and process-related data, such as people assigned to a task or custom properties for the task. This data is available at runtime for all or part of the life cycle of a task instance.

Part 2. Planning and configuring Business Process Choreographer

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 127.
   - For "Sample with organization", perform "Planning to create a sample Business Process Choreographer configuration including a sample organization" on page 127.
   - For "Non-production deployment environment", perform "Planning a non-production deployment environment configuration" on page 128.
   - For "Production deployment environment", perform "Planning to use the administrative console's deployment environment wizard" on page 130.
   - For "Flexible custom configuration", perform "Planning for a custom Business Process Choreographer configuration" on page 135.

Results

You have planned everything you need in order to configure Business Process Choreographer.

Related concepts

"Business Process Choreographer overview" on page 164
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
    - Business Process Choreographer Explorer with reporting function
    - A Business Process Choreographer event collector for the reporting function
- 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, which you can configure using its own configuration page in the administrative console, or by running a script. 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, which includes the Business Process Choreographer Explorer reporting function. For some database systems, it can also create a remote database. This configuration path is most suitable for creating production systems.

clientconfig.jacl configuration script
You can only use this script to configure a Business Process Choreographer Explorer, with or without the optional reporting function.

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 your message store uses either a file store or Derby Embedded data store, 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. If you were familiar with the Business Process Choreographer Observer up to version 6.1.2, be aware that it is now integrated in the Business Process Choreographer Explorer. It is now referred to as the Business Process Choreographer Explorer reporting function, and it can be accessed using the **Reports** tab in the Business Process Choreographer Explorer client. The reporting function uses the same URL as the Business Process Choreographer Explorer .

   When configuring the Business Process Choreographer Explorer in the administrative console, or using the bpeconfig.jacl configuration script or clientconfig.jacl configuration script there is an option to configure the Business Process Choreographer Explorer reporting function.

   If you migrated an existing Business Process Choreographer configuration, any Business Process Choreographer Observer configuration is not migrated. To use the Business Process Choreographer Explorer reporting function you must enable it, as described in "Enabling the Business Process Choreographer Explorer reporting function after migration" on page 282.

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

Table 14. Criteria for selecting a configuration path

Choices			Restrictions		Suitable configuration path
Are you planning a production system?	**What is the deploy- ment target?**	**Type of Business Process Choreographer configuration**	**Can use a separate BPEDB database?**	**Which message stores are supported for the messaging engine**	**Configuration path name, tools, and options**
No	Stand- alone server	Basic sample (without the sample organization)	Yes, but only Derby Embedded	Only Derby Embedded	**"Basic sample"** using one of: • Installer • Profile Management Tool Select the options: • **Stand-alone server profile** • **Typical** • **Enable administrative security**
		Sample including a 15 person organization, and substitution is enabled. This sample is identical to the sample that is available in WebSphere Integration Developer when you include the WebSphere Test Environment.		Derby Embedded, File Store, or WPRCSDB	**"Sample with organization"** using: • Profile Management Tool Select the options: • **Stand-alone server profile** • **Advanced** • **Create server from development template** • **Enable administrative security** • **Configure a sample Business Process Choreographer**
	Cluster	Choice of deployment environment patterns: • Remote Messaging and Remote Support • Remote Messaging • Single Cluster	No, it shares WPRCSDB, which can be any database except Derby Embedded and Microsoft® SQL Server	Shares WPRCSDB, which can be any supported database except File Store and Derby Embedded	**"Non-production deployment environment"** using one of: • Installer • Profile Management Tool Select: **Deployment environment**

Choices			Restrictions		Suitable configuration path
Are you planning a production system?	What is the deploy-ment target?	Type of Business Process Choreographer configuration	Can use a separate BPEDB database?	Which message stores are supported for the messaging engine	Configuration path name, tools, and options
Yes	Cluster	Choice of deployment environment patterns: • Remote Messaging and Remote Support • Remote Messaging • Single Cluster • Custom	Yes, any supported database except Derby Embedded	Any supported database except File Store and Derby Embedded	**"Production deployment environment"** using: • Administrative console 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: • bpeconfig.jacl script • 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.

   Restriction: If you use Microsoft SQL Server for the WPRCSDB database, the WPRCSDB database cannot be used for the Business Process Choreographer database because the SQL Server database is created as a case-insensitive database, but the BPEDB database used by Business Process Choreographer must be case-sensitive. Therefore, if you are using Microsoft SQL Server for the WPRCSDB database, you must have a separate case-sensitive 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 (populated with a sample organization) for people assignment and substitution enabled.

   Note: The sample people directory uses the default file registry configured for the federated repositories, 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. After the sample configuration has been created, you can use the administrative console to view which users and groups are available by clicking **Users and Groups**, then either **Manage Users** or **Manage Groups**.

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 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. Business Process Choreographer will use the same type of messages store that is used by SCA:
   - If SCA uses a FILESTORE, then Business Process Choreographer will also use a FILESTORE.
   - If SCA uses a Derby Embedded database, then Business Process Choreographer will use its own Derby Embedded database.
   - If SCA uses any other database, then Business Process Choreographer will use its own schema in the same database.

6. If you want to use the Business Process Choreographer Explorer reporting function, which is integrated in the Business Process Choreographer Explorer, 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 Explorer reporting function will also use the BPEDB database, or whether it will have its own, OBSRVRDB, database. Also plan the topology for the Business Process Choreographer Explorer reporting function components. To perform the detailed planning now, perform "Planning for the Business Process Choreographer Explorer reporting function" on page 159.

7. If you want WebSphere Portal Server or any custom WebSphere Process Server client to access Business Process Choreographer, perform "Planning for a remote client application" on page 162.

8. If you have application security enabled and you have a long-running process that calls a remote EJB method, Common Secure Interoperability Version 2 (CSIv2) identity assertion must be enabled when you configure CSIv2 inbound authentication.

9. If you will use human tasks, then WebSphere administrative security and application security must both be enabled.

Results

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

Related tasks

"Planning for a remote client application" on page 162
Planning for a remote Business Process Choreographer client application that uses the Business Process Choreographer APIs and runs on a WebSphere Process Server client installation.

Related information

➡ Profiles

➡ Deployment Environment Patterns

Planning to create a basic sample Business Process Choreographer configuration

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

Before you begin

You have performed "Planning the topology, setup, and configuration path" on page 121, and have selected the "Basic sample" configuration path.

Procedure

1. Decide whether you will create the sample using the Installer or Profile Management Tool.
2. If you decided to use the Profile Management Tool, 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.

Results

You have planned to create a basic sample Business Process Choreographer configuration.

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 when you include the WebSphere Test Environment.

Before you begin

You have performed "Planning the topology, setup, and configuration path" on page 121, and have selected the "Sample with organization" configuration path.

About this task

This sample Business Process Choreographer configuration requires minimal planning.

Procedure

1. Decide whether the Business Process Choreographer messaging engine will use file store, an embedded Derby database, or the common, WPRCSDB, database.
2. 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**
   - **Configure a sample Business Process Choreographer**

   If for example, you do not enable administrative security, the sample Business Process Choreographer configuration will not be created.

   Note: The sample people directory uses the default file registry configured for the federated repositories, 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. After the sample configuration has been created, you can use the administrative console to view which users and groups are available by clicking **Users and Groups**, then either **Manage Users** or **Manage Groups**.
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.

Results

You have planned to create a sample Business Process Choreographer configuration including a sample organization.

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

   Cleanup User Authentication User and Password
   The run-as user ID for the Business Flow Manager and Human Task Manager cleanup service. This user must be in the business administrator role.

5. 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 host name 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.

6. If you are going to use the Business Process Choreographer Explorer, the Business Space, or a client that uses the Representational State Transfer (REST) API or the JAX Web Services API, decide on the context roots for the REST API and the JAX Web Services API.

   - The defaults for the Business Flow Manager are `/rest/bpm/bfm` and `/BFMJAXWSAPI`.
   - The defaults for the Human Task Manager are `/rest/bpm/htm` and `/HTMJAXWSAPI`.
   - When configured on a server, or on a single cluster, or on multiple clusters that are mapped to different Web servers, you can use the default values.
   - When configured in a network deployment environment on multiple deployment targets that are mapped to the same Web server, do not use the default values. The context root for each Business Process Choreographer configuration must be unique for each combination of host name and port. You will have to set these values manually using the administrative console after configuring Business Process Choreographer.

7. If you want to use people assignment, perform "Planning for the people directory provider" on page 156.

Results

You have planned to create a non-production deployment environment configuration.

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 121, 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 (DBA) must be included in planning the databases. Your DBA will need a copy of the database scripts to customize and run.

2. Perform "Planning security, user IDs, and authorizations" on page 136.
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 the following:
      - Where you will deploy it.
      - If you want to use the Business Process Choreographer Explorer reporting function, also plan where you will deploy the Business Process Choreographer event collector.
   b. Plan the context root for the SCA bindings.
   c. 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, which can be changed in the wizard in a table row for the component WBI_BPC.
   - The BPEME database for the Business Process Choreographer messaging engine, which can be changed in the wizard in a table row for the component WBI_BPC_ME.
   - The OBSRVRDB database for the Business Process Choreographer Explorer reporting function, which can be changed in the wizard in a table row for the component WBI_BPCEventCollector.

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

   Database name
   > The name of the database, for example, BPEDB, BPEME, or OBSRVRDB 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 schema qualifier to be used for each database.

   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.

Also plan the database-specific settings, which you can set using the **Edit** button for the JDBC provider.

Table 15. Database-specific settings

Database / JDBC driver type	Database-specific settings
DB2 UDB – Universal driver	• User name • Password • Database name • Schema name • Server name • Server port number • Driver type • Description • Create tables
DB2 for i5/OS® – Toolbox driver	• User name • Password • Database name • Collection name • Server name • Description • Create tables
DB2 for z/OS® V8 and V9	• Implementation type – Connection pool data source or XA data source • User name • Password • Database name • Schema name • Server name • Server port number • Storage group • Description
Derby Network Server or Derby Network Server 40	• User name • Password • Description • Create tables • Server name • Server port number

Table 15. Database-specific settings (continued)

Database / JDBC driver type	Database-specific settings
Derby Embedded or Derby Embedded 40	• Description • Create tables
Microsoft SQL Server – Datadirect, and Microsoft drivers	• User name • Password • Database name • Server name • Server port number • Description • Create tables
Informix Dynamic Server – Universal and DataServer drivers	• User name • Password • Server name • Server port number • Description • Create tables
Oracle – oci driver	• User name • Password • Database name • Schema name • Driver type – oci • Description • Create tables
Oracle – thin driver	• User name • Password • Database name • Schema name • Server name • Server port number • Driver type – thin • Description • Create tables

For more details about planning the databases, see "Planning the databases for Business Process Choreographer" on page 142.

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

7. Plan the **Business Process Choreographer Explorer context root**, which defines part of the URL that browsers must use to reach the Business Process Choreographer Explorer.

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.

Cleanup User Authentication User and Password

The run-as user ID for the Business Flow Manager and Human Task Manager cleanup service. This user must be in the business administrator role.

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 host name 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 are going to use the Business Process Choreographer Explorer, the Business Space, or a client that uses the Representational State Transfer (REST) API or the JAX Web Services API, decide on the context roots for the REST API and the JAX Web Services API.

   • The defaults for the Business Flow Manager are `/rest/bpm/bfm` and `/BFMJAXWSAPI`.

   • The defaults for the Human Task Manager are `/rest/bpm/htm` and `/HTMJAXWSAPI`.

   • When configured on a server, or on a single cluster, or on multiple clusters that are mapped to different Web servers, you can use the default values.

   • When configured in a network deployment environment on multiple deployment targets that are mapped to the same Web server, do not use the default values. The context root for each Business Process Choreographer configuration must be unique for each combination of host name and port. You will have to set these values manually using the administrative console after configuring Business Process Choreographer.

11. If you want to use people assignment, perform "Planning for the people directory provider" on page 156.

Results

You have planned to use the administrative console's deployment environment wizard.

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 121, 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 (DBA) must be included in planning the databases. Your DBA will need a copy of the database scripts to customize and run.
3. "Planning security, user IDs, and authorizations" on page 136
4. "Planning the databases for Business Process Choreographer" on page 142
5. "Planning for the Business Flow Manager and Human Task Manager" on page 155
6. "Planning for the people directory provider" on page 156
7. "Planning for the Business Process Choreographer Explorer" on page 158
8. 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.
9. 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 180, and are described in detail in "bpeconfig.jacl script" on page 188.
   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 create a custom Business Process Choreographer configuration.

What to do next

Perform "Configuring Business Process Choreographer" on page 169.

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 the necessary rights for the configuration actions that the script will perform. Otherwise, you must specify user IDs as parameters for the script that have the necessary rights, in which 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 to invoke the wsadmin tool.

If you will use human tasks:
WebSphere administrative security and application security must both be enabled.

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 16. 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
The user who configures Business Process Choreographer	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" on page 188.	

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 17. 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: **BPEDB**: This is the default name for the database for Business Process Choreographer. **OBSRVDB**: This is the default name for the database for the Business Process Choreographer Explorer reporting function.	If you use the bpeconfig.jacl script to configure Business Process Choreographer: Read access to (or a copy of) the createSchema.sql script that bpeconfig.jacl generates in a subdirectory of the directory: • Linux UNIX On Linux and UNIX platforms: `profile_root/dbscripts/ProcessChoreographer/` • Windows On Windows platforms: `profile_root\dbscripts\ProcessChoreographer\`	
			If you want to review the database script files: Read access to (or a copy of the files in) the database scripts provided in the directory: • Linux UNIX On Linux and UNIX platforms: `install_root/dbscripts/ProcessChoreographer/database_type` • Windows On Windows platforms: `install_root\dbscripts\ProcessChoreographer\database_type` Where *database_type* is one of the following: • DB2 • DB2zOSV8 • DB2zOSV9 • DB2iSeries • Derby • Informix • Oracle • SQLServer	

Table 17. Planning access to the subdirectories of install_root (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
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.	Either read access to the `Staff` directory, or a copy of the files in the directory: • **Linux** **UNIX** On Linux and UNIX platforms: `install_root/ProcessChoreographer/Staff` • **Windows** On Windows platforms: `install_root\ProcessChoreographer\Staff` 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 18. 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. 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 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 Explorer reporting function, plan the user IDs to use to create, configure, and access the reporting database.

Table 19. Planning user IDs for the reporting 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 reporting database. For Oracle, to create the reporting database.	Create the database.	
Database administrator or an administrator	Configuring	Running the setupEventCollector tool, or SQL scripts to create the schema.	For the reporting database: Alter tables, connect, create procedure, insert tables, and create tables, table spaces, and views. 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.	
Event collector data source user name	Runtime	Connecting to the reporting database. If you are using the reporting database 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 20. Planning user ID for the preconfigured BPEME messaging engine database

User ID	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. If you use the bpeconfig.jacl script, this is the -medbUser parameter.	Configuring and runtime	This user name is used to connect to the BPEME database, 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 the Business Process Choreographer user IDs for the Java Message Service (JMS).

Table 21. Planning user IDs for JMS

User ID	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. If you use the bpeconfig.jacl script, this user IDs and its password are the parameters -mqUser and -mqPwd.	It must be a user name that exists in the WebSphere user registry. 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. If you use the bpeconfig.jacl script, this user IDs and its password are the parameters -jmsBFMRunAsUser and -jmsBFMRunAsPwd.	The user name must exist in the WebSphere user registry.	
Escalation authentication user	Runtime	Any Human Task Manager escalations will be processed using this user ID. If you use the bpeconfig.jacl script, this user ID and its password are the parameters -jmsHTMRunAsUser and -jmsHTMRunAsPwd.	The user name must exist in the WebSphere user registry.	

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

Table 22. 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. If you use the bpeconfig.jacl script, these users and groups correspond to the following parameters: • -adminUsers • -adminGroups • -monitorUsers • -monitorGroups	
Administrator group	Runtime		
Monitor user	Runtime		
Monitor group	Runtime		

9. Plan the user ID to use as the Java EE run-as role for administration jobs like the Business Flow Manager and Human Task Manager cleanup services and the process instance migration tool. This user ID must be a member of the administrator role user or group planned in Table 22.

Table 23. Planning the user ID for running administration jobs

User ID	When the user ID is used	What the user ID is used for	Planned user ID
Administration job user ID	Runtime administration	This user ID is used to run administration jobs. If you use the bpeconfig.jacl script, this user ID and its password correspond to the -adminJobUser and -adminJobPwd parameters.	

10. 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 24. 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. If you use the bpeconfig.jacl script, this is the -mailUser parameter. The password is the -mailPwd parameter.	Send e-mails.	

11. 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 a Java Authentication and Authorization Service (JAAS) alias and an associated user ID that will be used to connect to the LDAP server. If the LDAP server uses anonymous authentication this alias and user ID are not required.

Table 25. Planning the alias and user ID for the LDAP server

User ID or role	When the user ID is used	What the alias and user ID are used for	Which rights the user ID must have	Planned alias and user ID
LDAP plug-in property: Authentication Alias	Runtime	The alias is used to retrieve the user ID that is used to connect to the LDAP server. You specify this alias ID when customizing the properties for the LDAP plug-in, for example `mycomputer/My LDAP Alias`.	The JAAS alias must be associated with the LDAP user ID.	
LDAP user ID	Runtime	This user ID is used to connect to the LDAP server.	If the LDAP server uses simple authentication, this user ID must be able to connect to the LDAP server. This user ID is either a short name or a distinguished name (DN). If the LDAP server requires a DN you cannot use the short name.	

12. 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 or other Business Process Choreographer configurations. 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 have multiple Business Process Choreographer configurations, then each of these needs its own database or database schema. The Business Process Choreographer database tables cannot be shared between multiple Business Process Choreographer configurations.

Restriction: If you are using Informix, you cannot have multiple Business Process Choreographer configurations share the same database.

If you use the Business Process Choreographer Explorer reporting function, which until Version 6.1.2 was known as the Business Process Choreographer Observer, it can use the same BPEDB database, but using an additional database, gives better performance. Some of the scripts for setting up the reporting database already contain the suggested name OBSRVDB, though you are free to choose a different name.

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 14 on page 124.

Procedure
1. 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 144, otherwise, plan to use the WPRCSDB common database.
   b. If you will use the Business Process Choreographer Explorer reporting function:
      - If you want to minimize the impact that its queries have on the performance of your business processes, plan to use a separate database (OBSRVDB) as described in "Planning the reporting database" on page 149.
      - Otherwise, plan to configure it to use the BPEDB database.
   c. For high-load setups, for example, a large cluster with very high messaging rates, consider improving the performance by using a separate database for the Business Process Choreographer messaging engine. This allows the database logging to be parallelized, which can help to prevent it from becoming a bottleneck.

- If you use the administrative console to configure Business Process Choreographer, and you want a separate database for the Business Process Choreographer messaging engine, perform "Planning the messaging engine database" on page 154, otherwise plan to use the default database that is used by the Service Component Architecture (SCA).
- If you use the bpeconfig.jacl configuration script to configure Business Process Choreographer, Business Process Choreographer will use the same type of messages store that is used by SCA.

d. Optional: Use the database design tool to interactively create the database design file and the SQL script files that the database administrator can use to create all three databases that you planned in the previous steps. There are significant advantages to using this tool:

- You can run the tool as often as necessary to refine the database design parameters, without the risk of breaking them, rather than editing the provided template SQL files manually.
- If you have used a database design file, the next time that you migrate to a newer version of WebSphere Process Server, you can generate the schema update SQL scripts.
- If you create a database design file for a test configuration, it is convenient to be able to make of copy of the design file and make minor changes to it for the databases for your production system.
- Using the tool, you can also define the data sources for all three databases. Though you must configure the data source for the reporting database manually.

Important: When using the database design tool to create a deployment environment, after you have configured the common database, Business Process Choreographer is shown as being "complete". This is because there is a valid default, that causes the tables for Business Process Choreographer to be created in the common database. However, this default is not suitable for production systems. For a production system, make sure that you configure a dedicated database for each deployment target where Business Process Choreographer is configured.

2. 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" or the "Sample with organization" Business Process Choreographer configuration, a separate Derby Embedded BPEDB database is created, which is also used by the Business Process Choreographer Explorer reporting function. For the Business Process Choreographer messaging engine, the default is to have a separate Derby Embedded database (BPEME). If you use the Profile Management Tool, 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 Explorer reporting function, 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.

Results

You have planned all the databases 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. When 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 14 on page 124.

2. If you do not need the performance, scalability, and security that is normally required for a production system, you can have the database objects created in a single table space on a database server that is local to the WebSphere Process Server. This minimizes the planning and effort necessary to create the database, but requires that the user ID used to access the database also has database administration rights. The options that you need to plan depend on the configuration path that you choose:

   a. 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.

   b. If you use the administrative console **Deployment Environment wizard** to configure Business Process Choreographer, and it is sufficient for the default schema to be created in a single table space, plan to use a copy of the provided SQL script to create the BPEDB database.

c. If you use the **bpeconfig.jacl** tool to configure Business Process Choreographer, plan which of the following apply in your case.

- 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 database, the database must already exist. If you are using a DB2 for z/OS database, the database instance must already exist. For other database types, bepconfig.jacl will try to create the database.
  - 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" on page 188.

d. If you use the administrative console's **Business Process Choreographer configuration page**:

- To have the Business Process Choreographer database objects created in the common database WPRCSDB, plan to use the default database as the target for the Business Process Choreographer data source.
- To reuse an existing database, plan to specify the existing database 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 default table space, the first time that it uses the database. This option cannot be used for a DB2 on z/OS database nor for a remote Oracle database. To use this option for a DB2 UDB database, the database must have `AUTOMATIC STORAGE YES` enabled.
- To create the database using scripts, plan not to use the `Create tables` option.

e. Skip to step 12 on page 148.

3. Perform all of the following steps 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 is on a dedicated server, however it can also be local to the WebSphere Process Server system.
- 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.

4. If you have not already planned the user IDs for the database, perform Table 18 on page 138.

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.

For DB2 for z/OS, a table space is created for each table, and additional larger object (LOB) table spaces for LOB columns.

If you use one of the following:

- DB2
- Informix (table spaces are known as named dbspaces)
- Oracle

Plan where to locate the BPEDB database table spaces.

- The AUDITLOG table space stores audit events mainly for compatibility with earlier versions. It is not used much.
- The INDEXTS table space is only created for Oracle databases. It stores indexes, is used intensively and its growth rate correlates with the number of instances.
- The INSTANCE table space is only created for Oracle databases. It stores instance data for instances of business processes and human tasks. It is used intensively and its growth rate depends on your business applications.
- The LOBTS table space stores large data objects of instances of business processes and human tasks. It is used intensively and its growth rate correlates with the number of instances.
- The SCHEDTS table space stores scheduler information related to business processes and human tasks. It is used frequently and its growth rate correlates to the number of instances.
- The STAFFQRY table space stores authorization data for business process. It is used frequently and its growth rate depends on how you have modelled authorization.
- The TEMPLATE table space stores template information. It is used frequently and its growth rate correlates to the number and size of installed business process and human task applications.
- The WORKITEM table space stores authorization data for business process and human tasks. It is used intensively, and its growth rate correlates with the number of instances.
- For DB2 UDB, the following are also created:
    – BUFFERPOOL BPEBP8K
    – TEMPORARY TABLESPACE BPETEMP8K
    – TABLESPACE BPETS8K

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 1.5*N primary disks, 1.5*N mirror disks, and a stripe size of 64 kb.

If you are using a DB2 database, running on a four-processor server, 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 original 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.
- 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. The generated SQL files are based on the original SQL files that are provided for your database, but all configuration parameters that you enter in the administrative console are pre-filled in the generated SQL file, which minimizes the customization necessary.

For more information about using the generated SQL scripts, refer to "Using a generated SQL script to create the database schema for Business Process Choreographer" on page 208. If you want to preview the original 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 original SQL files are located in the following directory:

- `Linux` `UNIX` On Linux and UNIX platforms: *install_root*/`dbscripts/ProcessChoreographer/`*database_type*
- `Windows` On Windows platforms: *install_root*`\dbscripts\` `ProcessChoreographer\`*database_type*

Where *database_type* is one of the following:

- DB2
- DB2zOSV8
- DB2zOSV9
- DB2iSeries
- Derby
- Informix
- Oracle
- SQLServer

7. If the database server is remote to the WebSphere Process Server system, plan to install either a Java Database Connectivity (JDBC) driver or a database client on the WebSphere Process Server system:

   - 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, which is provided as part of your product installation, 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.

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. For an overview of the tools and scripts that are available, see Cleanup procedures for Business Process Choreographer.

Results

You have planned the database for Business Process Choreographer.

Planning the reporting database

Plan the database for the Business Process Choreographer Explorer reporting function.

About this task

Business Process Choreographer Explorer reporting function can use the same database, but using an additional database 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 Explorer reporting function 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 19 on page 139.

5. If you are **not** using a Derby database for the reporting database, decide whether you will use SQL-based or Java-based user-defined functions (UDFs).
   - The Java UDFs are more precise, but to be able to use them, you must install a JAR file in the database.
   - 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 Explorer reporting function" on page 270.

6. If you will not use the `bpeconfig.jacl` script to configure the Business Process Choreographer Explorer reporting function and event collector to use the BPEDB database, decide how you will create the reporting 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 290.

Unlike the other ways to create the database, this tool allows you to create either Java-based user-defined functions (UDFs) or SQL-based UDFs. You can also use it 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 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. If you configured Business Process Choreographer using the bpeconfig.jacl script in batch mode or using the administrative console, an SQL script is generated that has all necessary parameters substituted. Otherwise, you can use the database design tool to generate the SQL scripts interactively.

For a non-Derby database, all SQL scripts create the UDFs for the reporting database using the SQL implementation. 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 event collector 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. 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 Derby Network server data source, you must start the Derby network server from the directory *install_root*/derby/bin/networkServer, otherwise creating the tables will fail with the error CWWB04013E: The bpcodbutil.jar file could not be found on the Derby network server.

7. If you use a DB2 for Linux, UNIX, or Windows database, plan the following:
   - The database name. If performance in not a priority, you can use the value BPEDB, so that the reporting database uses the Business Process Choreographer database. For better performance, plan to use a separate database, for example, named 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 to use to create the database objects. 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 host name 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.
8. If you use a DB2 for i5/OS database, plan the following:
  - The database name. 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 host name of the database server. Typically this is always `localhost`. The port number is always 446.
    - The directory for the JDBC driver:
      - This is the path where the `jt400.jar` file is located.
9. 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.
  - Plan in which storage group the table spaces will be created:
    - 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 host name 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.

10. 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 or Embedded 40 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 or Network 40 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 host name of the database server. The default is localhost.
      - The port number used for the database. The default is 1527.

11. 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. For Oracle 10g use the ojdbc5.jar driver. For Oracle 11g use the ojdbc6.jar driver.
  - The host name of the database server. The default is localhost.
  - The port number used for the database. The default is 1521.

12. 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, and you can use the -reportSchemaName and -reportDataSource parameters to make the Business Process Choreographer Explorer reporting function use a different database rather than using the BPEDB database.
- 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 180.

13. 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 cause the tool to create 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.

14. 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:
    - `Linux` `UNIX` On Linux, UNIX platforms (and in the UNIX System Services (USS) on z/OS):*install_root*/dbscripts/ProcessChoreographer/*database_type*
    - `Windows` On Windows platforms: *install_root*\dbscripts\ProcessChoreographer*database_type*

    Where *database_type* is one of the following:
    - DB2
    - DB2zOSV8
    - DB2zOSV9
    - 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. Alternatively, you can use the database design tool to generate the SQL scripts.

Results

You have planned the reporting database.

Planning the messaging engine database

For high-load setups, where database logging might become a bottleneck, 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 the same messaging database for each messaging engine for the Service Component Architecture (SCA) system bus, each messaging engine for the SCA application bus, each messaging engine for the Common Event Infrastructure bus, and each messaging engine for 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, Business Process Choreographer will use the same type of messages store that is used by SCA.
    *   If SCA uses a FILESTORE, then Business Process Choreographer will also use a FILESTORE.
    *   If SCA uses a Derby Embedded database, then Business Process Choreographer will use its own Derby Embedded database.
    *   If SCA uses any other database, then Business Process Choreographer will use its own schema in the same database.
5. If you use the administrative console's Business Process Choreographer configuration page, if you want to use the default configuration that is based on the SCA message store settings, plan to select the **Use the default configuration** check box, otherwise, plan the following configuration parameters:
    *   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 a file store or the embedded Derby JDBC provider, the message stores will be created automatically.
7. If you are not using a 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 20 on page 139.

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 21 on page 140, 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 21 on page 140, 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 22 on page 140.

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. If you are going to use the Business Process Choreographer Explorer, the Business Space, or a client that uses the Representational State Transfer (REST) API or the JAX Web Services API, decide on the context roots for the REST API and the JAX Web Services API.
   - The defaults for the Business Flow Manager are /rest/bpm/bfm and /BFMJAXWSAPI.
   - The defaults for the Human Task Manager are /rest/bpm/htm and /HTMJAXWSAPI.
   - When configured on a server, or on a single cluster, or on multiple clusters that are mapped to different Web servers, you can use the default values.

- When configured in a network deployment environment on multiple deployment targets that are mapped to the same Web server, do not use the default values. The context root for each Business Process Choreographer configuration must be unique for each combination of host name and port. You will have to set these values manually using the administrative console after configuring Business Process Choreographer.

7. Decide whether you want to initially enable audit logging for the Business Flow Manager, or Human Task Manager, or both.

8. If you are going to use the Business Process Choreographer Explorer reporting function, 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 federated repositories for WebSphere Application Server security. You can associate federated repositories with one or more user repositories, based on a file, LDAP, or a database. For more information about this, see Managing the realm in a federated repository configuration. For more information about using federated repositories, 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 on page 157.

   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. Depending on the WebSphere security realm definition, the user registry can use one of the following repositories:

- Federated repository – which can use the following:
  - File registry
  - One or more LDAPs
  - One or more databases
- Standalone LDAP
- Standalone custom
- Local operating system

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 225.

   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 or if AuthenticationType is not set to `simple` then an 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 using two single quotes, `''`.
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. For a high availability configuration, where you have two or more LDAP servers that maintain mirrored data, plan to specify a URL for each LDAP server and use the space character to separate them.
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`
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 224.

4. If you want to use people substitution, consider the following:
   - You must use the VMM people directory provider. The LDAP, system, and user registry people directory providers do not support people substitution.
   - If you are going to use people substitution in a production environment, plan to use the VMM Property Extension Repository to store the substitution information. The Property Extension Repository and, implicitly, the selected database must be unique and accessible from within the whole 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.
   - To use people substitution in a single-server test environment, you can store people substitution information in the internal file registry that is configured for federated repositories.

Results

You have planned the people directory provider and people assignment options.

Planning for the Business Process Choreographer Explorer

Plan the configuration options and parameters for the Business Process Choreographer Explorer.

About this task

If you will use the Business Process Choreographer Explorer you can either configure it at the same time as you configure Business Process Choreographer, or you can do it later. The Business Process Choreographer Explorer reporting function is optional.

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 and considerations include:
   - Because each Business Process Choreographer Explorer instance can only connect to 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.
   - You might want to have two or more different customized versions of the Business Process Choreographer Explorer connecting to the same Business Process Choreographer configuration. You can customize each version independently, for more information about what you can customize, see "Customizing Business Process Choreographer Explorer" on page 377.
   - You can configure multiple Business Process Choreographer Explorer instances on each server or cluster.

- The instances can be created on any deployment target regardless of where there are Business Process Choreographer or Business Process Choreographer event collector configurations.
- Because each Business Process Choreographer Explorer instance's reporting function can only connect to one Business Process Choreographer event collector, plan to configure equal numbers of Business Process Choreographer Explorer instances with the reporting function as there are Business Process Choreographer event collectors.

2. For each Business Process Choreographer Explorer instance that you want, plan the following:
   a. The context root for the Business Process Choreographer Explorer. It must be unique within the cell. The default is /bpc.
   b. The URL for the Business Process Choreographer Explorer that will be inserted in escalation e-mails.
   c. The URL for the Business Flow Manager and Human Task Manager representational state transfer (REST) APIs endpoints. They must match the values for the context roots that you planned for the REST APIs. For example, if the context root for the Human Task Manager Web service is /rest/bpm/htm, the endpoint URL for the Human Task Manager REST API endpoint would be http://*hostname*:*port*/rest/bpm/htm.
   d. The maximum number of results to be returned for a query - the default is 10000.
   e. The deployment target (server or cluster) of the Business Process Choreographer instance that this Business Process Choreographer Explorer will manage.
   f. Optional: If you will use the Business Process Choreographer Explorer reporting function, perform "Planning for the Business Process Choreographer Explorer reporting function." You can also plan and configure it later.

Results

You have planned the configuration options for the Business Process Choreographer Explorer.

Planning for the Business Process Choreographer Explorer reporting function
Plan to configure the Business Process Choreographer Explorer reporting function and event collector.

About this task

If you will use the Business Process Choreographer Explorer reporting function, you can either configure it when you configure Business Process Choreographer Explorer, or you can do it later.

Procedure
1. Because security roles are not used to restrict access to the Business Process Choreographer Explorer reporting function, if you do not want all Business Process Choreographer Explorer users to have access to the reporting function, plan to configure a separate Business Process Choreographer Explorer instance for the reporting function, and make it inaccessible to normal users.
2. Understand the purpose and relationships between the different Business Process Choreographer Explorer reporting function topology elements.

The Business Process Choreographer Explorer reporting function.
Before version 6.2, this feature was available as the Business Process Choreographer Observer. Since version 6.2, this feature is integrated in the Business Process Choreographer Explorer, and is available on the **Reports** tab. You must configure the Business Process Choreographer Explorer reporting function before you can use it.

The event collector application.
This application must be deployed on a server or cluster where the Common Event Infrastructure (CEI) server is configured. You can only have one event collector on each CEI deployment target. 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 reporting database.

The reporting database.
The event collector and Business Process Choreographer Explorer reporting function 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 "Business Process Choreographer Explorer reporting function overview" on page 165.

3. Identify the purpose of your setup, your system requirements, and the topology implications.

Simple setup
For simpler configuration and administration, but lower performance, deploy the event collector application on the same deployment target as you have the Business Process Choreographer Explorer 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 Business Process Choreographer Explorer 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.

4. If you have not already planned the database for the Business Process Choreographer Explorer reporting function, perform "Planning the reporting database" on page 149.

5. 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 281.
      • 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 279.

- 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 Explorer reporting function for a high-performance system, because bpeconfig.jacl will configure the event collector and Business Process Choreographer Explorer reporting function applications on the same deployment target as the Business Process Choreographer configuration. For more information about this option, see "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 180.
  You cannot use bpeconfig.jacl to configure the event collector in interactive mode.

c. Plan the data source:
  - If the Business Process Choreographer Explorer reporting function shares the same physical database as Business Process Choreographer, plan to use a separate data source for the reporting database, and plan its JNDI name.
  - Plan an authentication alias that will be used for the database.
  - 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 reporting 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 depends on the database: For DB2 the default is db2admin, for Oracle the default is system, and for other databases the default is the user ID of the logged on user.
  - 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 Explorer reporting function" on page 284.

6. For each Business Process Choreographer Explorer reporting function that you configure, plan the following:
   - Decide how you will configure this instance:
     – At the same time as creating the Business Process Choreographer Explorer, using the administrative console page for the Business Process Choreographer Explorer. For more information about this option, see "Using the administrative console to configure the Business Process Choreographer Explorer reporting function" on page 283.
     – At the same time as creating the Business Process Choreographer Explorer, using the `clientconfig.jacl` script.
     – At the same time as creating a Business Process Choreographer configuration, using the `bpeconfig.jacl` script.

       Note: Do not use `bpeconfig.jacl` to configure the Business Process Choreographer Explorer reporting function for a high-performance system, because `bpeconfig.jacl` will configure the event collector and Business Process Choreographer Explorer reporting function applications on the same deployment target as the Business Process Choreographer configuration. For more information about this option, see "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 180.
   - The schema name for the reporting database.
   - The JNDI name for the data source that is used by the Business Process Choreographer Explorer to connect to the reporting database.

7. 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 Business Process Choreographer Explorer 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 Business Process Choreographer Explorer reporting function, plan to use one or both of the `bpeconfig.jacl` options `-createEventCollector no` and `-reportFunction no`, which prevent `bpeconfig.jacl` from configuring them.

Results

You have planned the configuration options for the Business Process Choreographer Explorer reporting function and event collector.

Planning for a remote client application

Planning for a remote Business Process Choreographer client application that uses the Business Process Choreographer APIs and runs on a WebSphere Process Server client installation.

About this task

If you want an application to use the Business Process Choreographer APIs, you can use a WebSphere Process Server client installation to run the applications remotely against a full WebSphere Process Server server installation. The client is easier to configure and administer than a full WebSphere Process Server installation.

The WebSphere Process Server client installation does not contain WebSphere Process Server profile templates, but needs to augment the underlying WebSphere Application Server profile with Feature Pack for SCA Version 1.0 with SDO 2.1.1. This means that you can even install the WebSphere Process Server client on top of an existing WebSphere Application Server installation that has federated profiles and those federated WebSphere Application Server profiles can take advantage of the WebSphere Process Server client functionality immediately. This scenario is not possible with the full WebSphere Process Server server because WebSphere Process Server does not support augmentation of already federated profiles.

Procedure

1. Plan to install a WebSphere Process Server client.
   - If you want WebSphere Portal Server to access Business Process Choreographer, you must have a compatible WebSphere Process Server client installed.

Table 26. WebSphere Process Server client versions that WebSphere Portal Server can use to access Business Process Choreographer

WebSphere Portal Server version	WebSphere Process Server client version			
	6.1.0.1	6.1.2	6.2	7.0
6.1.0.1	Yes	Yes	No	No
6.1.0.2	Yes	Yes	Yes	No

   - Any existing profiles, including already federated profiles, can use WebSphere Process Server client immediately, because the client installation does not augment the base profile.
   - If there is no existing WebSphere Application Server installation, a WebSphere Application Server network deployment installation will be created.
2. Decide which type of Business Process Choreographer client application you will use:
   - Custom client application
   - Business Process Choreographer Explorer

     Note: If you use customized JavaServer Pages (JSP), as described in "Developing JSP pages for task and process messages" on page 637, make sure that you know where they are located.
3. If you are going to develop a custom client application that will use Business Process Choreographer, plan which interfaces the application will use. You can handle processes and tasks using one of the following:
   - Web services API, Java Messaging Service (JMS) API or representational state transfer (REST) API – remote client applications that are based on these APIs do not need any WebSphere Process Server installation.

- JavaServer Faces (JSF) components
- Enterprise JavaBeans™ (EJB) API

Note: If you develop a client application, which uses the Business Process Choreographer EJB APIs, it must be packaged in the way that is described in "Accessing the remote interface of the session bean" on page 538.

4. Decide or identify the type of cell where the WebSphere Process Server client will be installed:

   a. In a cell where a managed server or cluster is located, on which Business Process Choreographer is configured, the default configuration of the Remote Artifact Loader (RAL) allows the unsecured transmission of artifacts between the client and the server. This is known as the "single-cell" scenario.

   b. In a cell that does not have a managed server or cluster with Business Process Choreographer configured on it, there are different deployment managers. This is known as the "cross-cell" scenario. If your client application uses the EJB API, you must define a namespace binding so that the client application can locate the server or cluster where Business Process Choreographer is configured.

Results

You have planned for a remote Business Process Choreographer client application.

Business Process Choreographer overview

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 allows you to administer business processes and human tasks. It also includes the optional Business Process Choreographer Explorer reporting function, formerly known as the Business Process Choreographer Observer, which allows you to observe the states of running processes.
- Human workflow widgets as part of Business Space. These widgets allow you to manage work, create tasks for other people, and initiate services and processes.

Related tasks

Planning to configure Business Process Choreographer
Plan your Business Process Choreographer setup and configuration parameters.

Business Process Choreographer Explorer overview

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

It also includes an optional reporting function, which was previously known as the Business Process Choreographer Observer.

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 WebSphere Process Server client installation is only the infrastructure that you need to connect a client to a WebSphere Process Server, it does not contain the Business Process Choreographer Explorer. Use the deployment manager to install the Business Process Choreographer Explorer 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.

Business Process Choreographer Explorer reporting function overview

About Business Process Choreographer Explorer reporting function.

You can use the Business Process Choreographer Explorer reporting function 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.

The Business Process Choreographer Explorer reporting function 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.

The Business Process Choreographer Explorer reporting function is based on two Java EE applications, which are shown in the following figure:
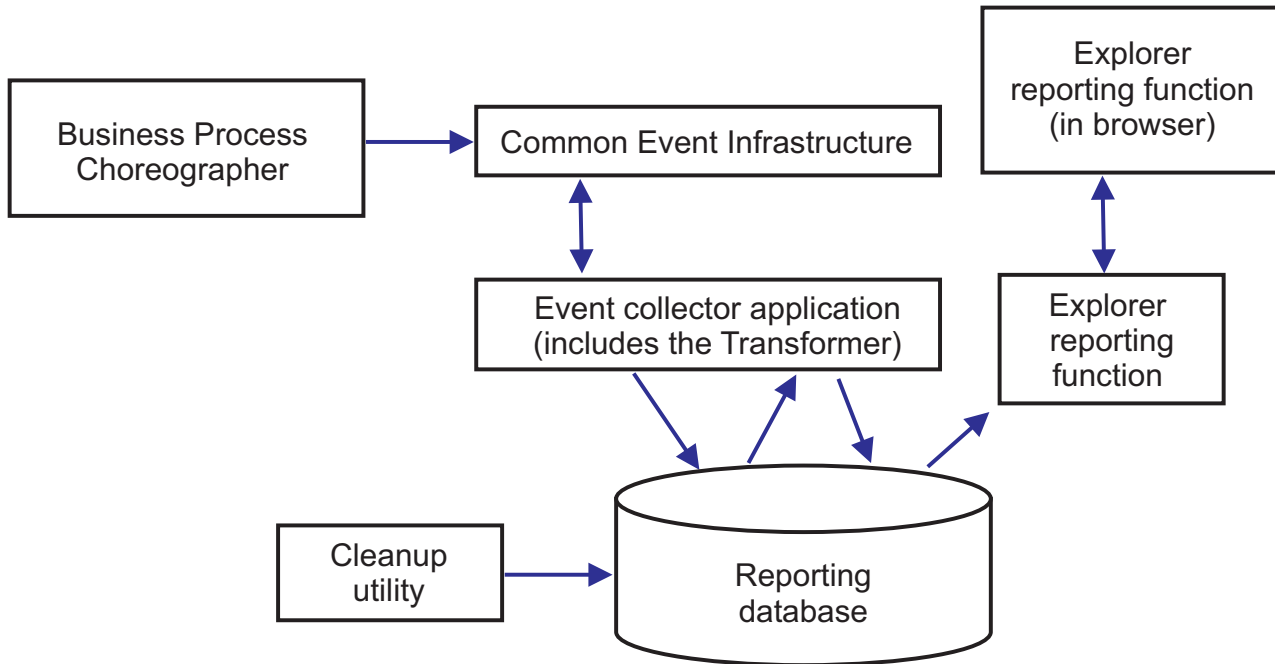

Figure 1. Architecture

- The event collector application reads event information from the CEI bus and stores it in the event collector table in the reporting database.
- The reporting database is a set of database tables that store the event data.
- Periodically the event transformer is triggered, which transforms the raw event data into a format that is suitable for queries from the Business Process Choreographer Explorer reporting function.
- The Business Process Choreographer Explorer reporting function generates the reports and performs other actions that the user can initiate using the 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 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.
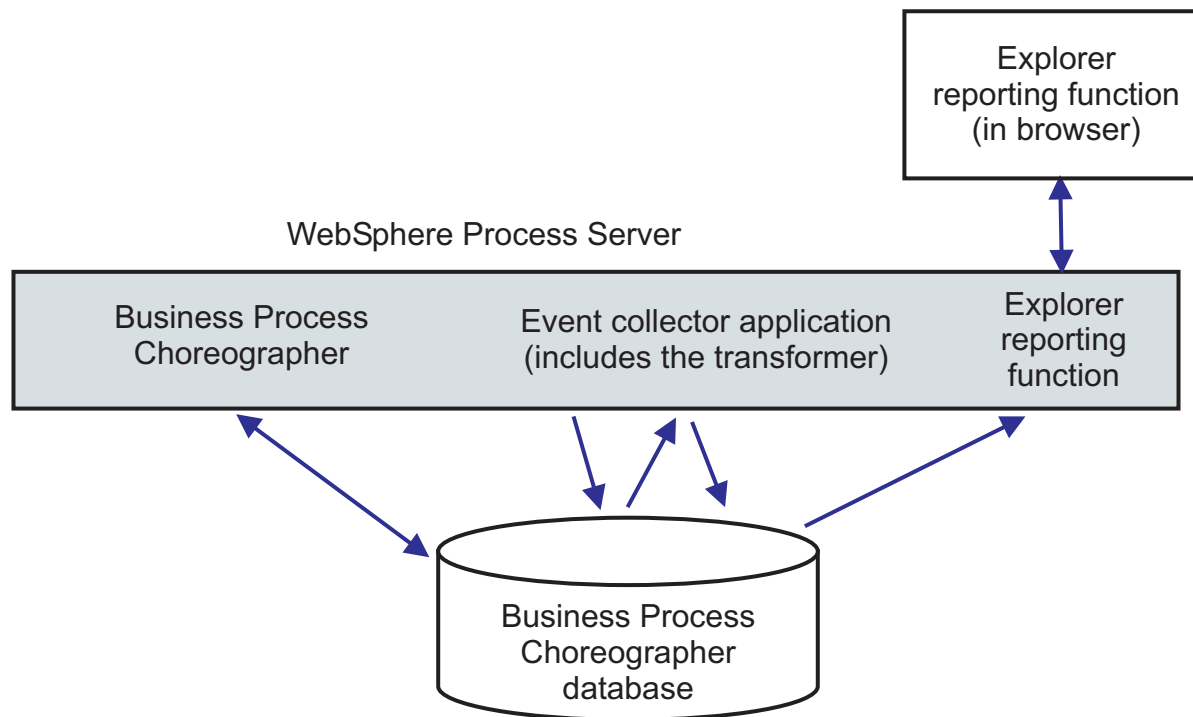
Figure 2. Stand-alone setup

Everything is installed on a single system, and Business Process Choreographer and Business Process Choreographer Explorer reporting function 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 this kind of setup 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 Explorer reporting function architecture. For example, in an ideal configuration for performance, the Business Process Choreographer configuration, CEI event server, and the Business Process Choreographer Explorer (with reporting function) run on separate systems, and Business Process Choreographer and the Business Process Choreographer Explorer reporting function have their own databases.
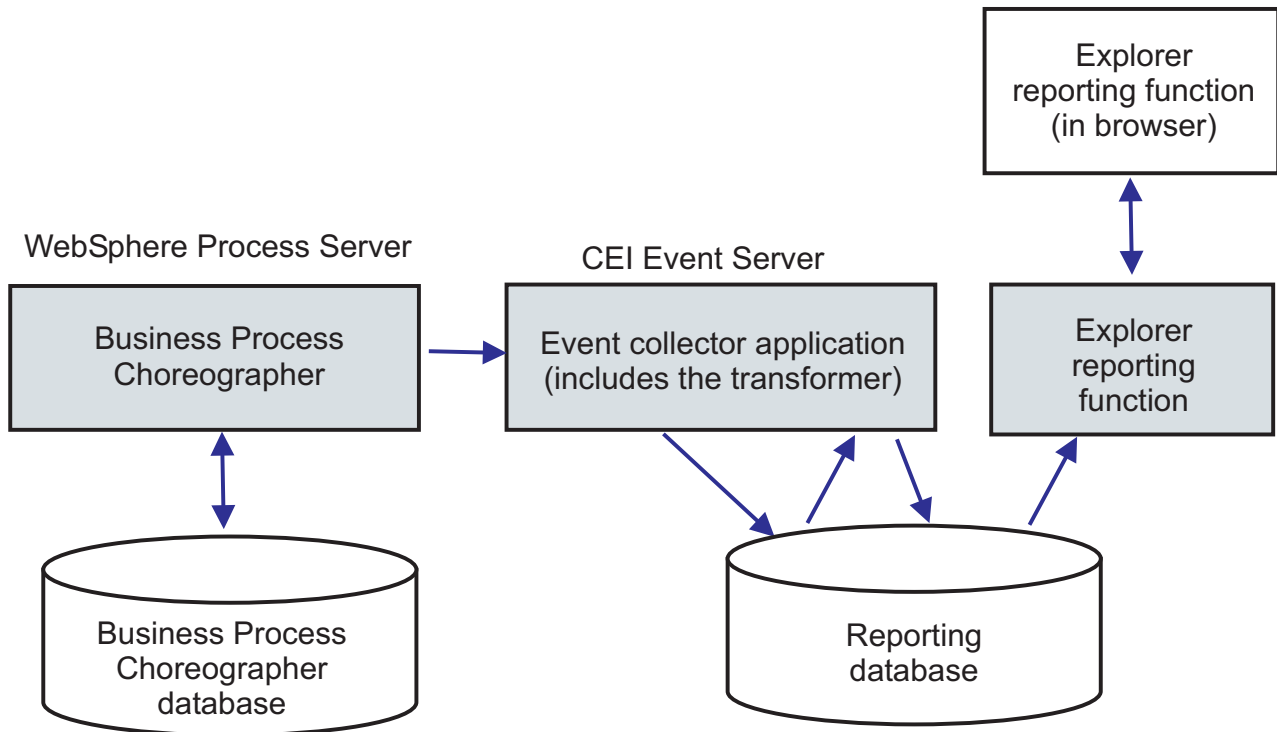
Figure 3. Business Process Choreographer Explorer reporting setup for production performance

If you want to use a separate database for the Business Process Choreographer Explorer reporting function, or to add the Business Process Choreographer Explorer reporting function to an existing Business Process Choreographer configuration in a clustered setup, or to use more sophisticated database options, perform "Configuring the Business Process Choreographer Explorer reporting function and event collector" on page 240.

In a network deployment environment

The following constraints apply if you want to configure Business Process Choreographer Explorer reporting function 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 Explorer reporting function application does not need to be installed on the same system as the event collector.

Configuring Business Process Choreographer

You must configure Business Process Choreographer on a server or cluster before you install any enterprise applications that contain business processes, human tasks, or both.

Before you begin

It is important that you have completed "Planning to configure Business Process Choreographer" on page 121.

About this task

For each server or cluster where you want to configure Business Process Choreographer, and depending on the configuration path that you have chosen, perform one of the following.

Procedure

- 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 172.
- For the "Flexible custom configuration" configuration path, perform the actions described for the configuration tool that you want to use:
  - "Using the administrative console's Business Process Choreographer configuration page" on page 176.
  - "Using the bpeconfig.jacl script to configure Business Process Choreographer" on page 180.

Results

Business Process Choreographer is configured.

What to do next

- To create more Business Process Choreographer configurations on other servers or clusters, repeat this task.
- If you have not configured the Common Event Infrastructure yet, perform Configuring Common Event Infrastructure.

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 127 and you have decided which flavor of non-production system you want, as summarized in Table 14 on page 124.

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 297.

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. If you configured Business Process Choreographer in a network deployment environment:

   a. Map the Web modules for the BPEContainer and TaskContainer applications to a Web server to achieve load balancing and failover. You might need to change the default context roots for the REST API and the JAX Web Services API so that they are unique for each combination of host name and port. To set the context roots perform the following:

      1) To set the context roots for the Business Flow Manager, click **Applications** → **Application Types** → **WebSphere enterprise applications** then **BPEContainer_***suffix* → **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules BFMRESTAPI and BFMJAXWSAPI are correct and unique.

      2) To set the context roots for the Human Task Manager, click **Applications** → **Application Types** → **WebSphere enterprise applications** then **TaskContainer_***suffix* → **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules HTMRESTAPI and HTMJAXWSAPI are correct and unique.

   b. If you changed any of the context roots for the REST API you must also modify the corresponding endpoints:

      1) If you use the Business Process Choreographer Explorer: Change the REST endpoint to match the new context roots by clicking either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**, then set the new value. For example, if the context root for the Business Flow Manager REST API is /rest/bpm/bfm then the full URL might be something like `http://localhost:9080/rest/bpm/bfm`.

      2) If you use the Business Space: Change the REST endpoints to match the new context roots by clicking either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** →

Configuring Business Process Choreographer **171**

WebSphere application servers ˃ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and either **Business Flow Manager** or **Human Task Manager**, then under **Additional Properties** click **REST Service Endpoint**, and set the new value.

5. 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 or the URL prefix for the Business Process Choreographer Explorer, click either **Servers** ˃ **Clusters** ˃ **WebSphere application server clusters** ˃ *cluster_name* or **Servers** ˃ **Server Types** ˃ **WebSphere application servers** ˃ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, and make your changes.
   - If you want to change the e-mail server address, port number, the user ID, or password for the e-mail server, click **Resources** ˃ **Mail** ˃ **Mail sessions**, select **Cell** scope, then click **HTM mail session_*suffix***, where *suffix* is either `node_name_server_name` or `cluster_name`, depending on where Business Process Choreographer is configured. Make your changes.

6. 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 225.
   - If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 224.

7. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 231.

8. Optional: If you want to use group work items, use the administrative console to enable them. Click either **Servers** ˃ **Clusters** ˃ **WebSphere application server clusters** ˃ *cluster_name* or **Servers** ˃ **Server Types** ˃ **WebSphere application servers** ˃ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, then select **Enable group work items**.

9. Optional: If you want to configure a remote Business Process Choreographer client that uses the WebSphere Process Server client, perform, "Configuring a remote client application" on page 292.

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 130.

Procedure

1. Start the deployment environment wizard. In the administrative console click **Servers** → **Server Types** → **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 130:

   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 a generated SQL script to create the database schema for Business Process Choreographer" on page 208. Otherwise, if you did not specify a separate database and you are not using a 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 Explorer reporting function, perform "Preparing the reporting database" on page 240. 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 you want to configure Business Process Choreographer, 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 will be 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 Universal driver, select DB2UNIVERSAL_JDBC_DRIVER_PATH.
      - For DB2 on i5/OS, select OS400_TOOLBOX_JDBC_DRIVER_PATH.
      - For DB2 - IBM DataServer, select DB2_JCC_DRIVER_PATH.
      - For Informix, using IBM DataServer, select INFORMIX_JCC_DRIVER_PATH.
      - For Informix, using IBM DB2 JDBC Universal Driver, select INFORMIX_JCC_DRIVER_PATH.
      - For Microsoft SQL Server using the WebSphere embedded ConnectJDBC driver you do not need to set any environment variable.
      - For Microsoft SQL Server using the DataDirect ConnectJDBC type-4 driver, select CONNECTJDBC_JDBC_DRIVER_PATH.
      - For Microsoft SQL Server using the Microsoft SQL Server JDBC Driver, select MICROSOFT_JDBC_DRIVER_PATHFor Oracle, select ORACLE_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 296.

7. Optional: Verify that the basic Business Process Choreographer configuration works: Perform "Verifying that Business Process Choreographer works" on page 297.

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 or the URL prefix for the Business Process Choreographer Explorer, click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, and make your changes.

   - If you want to change the e-mail server address, port number, the user ID, or password for the e-mail server, click **Resources** → **Mail** → **Mail sessions**, select **Cell** scope, then click **HTM mail session_*suffix***, where *suffix* is either `node_name_server_name` or `cluster_name`, depending on where Business Process Choreographer is configured. Make your changes.

9. If you configured Business Process Choreographer in a network deployment environment:

   a. Map the Web modules for the BPEContainer and TaskContainer applications to a Web server to achieve load balancing and failover. You might need to change the default context roots for the REST API and the JAX Web Services API so that they are unique for each combination of host name and port. To set the context roots perform the following:

1) To set the context roots for the Business Flow Manager, click **Applications** ▸ **Application Types** ▸ **WebSphere enterprise applications** then **BPEContainer_*suffix*** ▸ **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules BFMRESTAPI and BFMJAXWSAPI are correct and unique.

2) To set the context roots for the Human Task Manager, click **Applications** ▸ **Application Types** ▸ **WebSphere enterprise applications** then **TaskContainer_*suffix*** ▸ **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules HTMRESTAPI and HTMJAXWSAPI are correct and unique.

b. If you changed any of the context roots for the REST API you must also modify the corresponding endpoints:

1) If you use the Business Process Choreographer Explorer: Change the REST endpoint to match the new context roots by clicking either **Servers** ▸ **Clusters** ▸ **WebSphere application server clusters** ▸ *cluster_name* or **Servers** ▸ **Server Types** ▸ **WebSphere application servers** ▸ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**, then set the new value. For example, if the context root for the Business Flow Manager REST API is /rest/bpm/bfm then the full URL might be something like `http://localhost:9080/rest/bpm/bfm`.

2) If you use the Business Space: Change the REST endpoints to match the new context roots by clicking either **Servers** ▸ **Clusters** ▸ **WebSphere application server clusters** ▸ *cluster_name* or **Servers** ▸ **Server Types** ▸ **WebSphere application servers** ▸ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and either **Business Flow Manager** or **Human Task Manager**, then under **Additional Properties** click **REST Service Endpoint**, and set the new value.

10. 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 225.
- If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 224.

11. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 231.

12. Optional: If you want to use group work items, use the administrative console to enable them. Click either **Servers** ▸ **Clusters** ▸ **WebSphere application server clusters** ▸ *cluster_name* or **Servers** ▸ **Server Types** ▸ **WebSphere application servers** ▸ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, then select **Enable group work items**.

13. Optional: If you want to configure a remote Business Process Choreographer client that uses the WebSphere Process Server client, perform, "Configuring a remote client application" on page 292.

14. If you have WebSphere application security enabled and you have a long-running process that calls a remote EJB method, make sure that your Common Secure Interoperability Version 2 (CSIv2) inbound authentication configuration has CSIv2 identity assertion enabled. For more information about this, refer to Configuring Common Secure Interoperability Version 2 inbound communications.

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 Explorer reporting function are already configured.

   You can check if they are configured, by looking in the administrative console for enterprise applications with names that start with:
   - BPCECollector
   - BPCExplorer
   - BPEContainer
   - HTM_PredefinedTasksMsg
   - HTM_PredefinedTasks
   - 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 "Removing the Business Process Choreographer configuration" on page 301 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** › **Server Types** › **WebSphere application servers** › *server_name*, 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** → **WebSphere application server clusters** → *cluster_name*, then in the **Business Integration** section, click **Service Component Architecture**.

c. If it is not enabled, select **Support the Service Component Architecture components**, then click **Apply** and **Save**.

3. Create the Business Process Choreographer BPEDB database:

   - 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, perform the following:

     a. If the database does not already exist, create an empty database using the database tool of your choice.

     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. 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 (the next time that the messaging engine is started), perform the following:

     a. If the database does not already exist, create an empty database using the database tool of your choice.

     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.

5. For each node where you want to configure Business Process Choreographer, 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 will be 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 Universal driver, select DB2UNIVERSAL_JDBC_DRIVER_PATH.
      - For DB2 on i5/OS, select OS400_TOOLBOX_JDBC_DRIVER_PATH.
      - For DB2 - IBM DataServer, select DB2_JCC_DRIVER_PATH.
      - For Informix, using IBM DataServer, select INFORMIX_JCC_DRIVER_PATH.
      - For Informix, using IBM DB2 JDBC Universal Driver, select INFORMIX_JCC_DRIVER_PATH.

- For Microsoft SQL Server using the WebSphere embedded ConnectJDBC driver you do not need to set any environment variable.
- For Microsoft SQL Server using the DataDirect ConnectJDBC type-4 driver, select CONNECTJDBC_JDBC_DRIVER_PATH.
- For Microsoft SQL Server using the Microsoft SQL Server JDBC Driver, select MICROSOFT_JDBC_DRIVER_PATHFor Oracle, select ORACLE_JDBC_DRIVER_PATH.

   c. Set the environment variable to point to the location of the JDBC driver's JAR file, or files.

6. In the administrative console, select the server or cluster where you want to configure Business Process Choreographer. Click either **Servers → Clusters → WebSphere application server clusters → *cluster_name*** or **Servers → Server Types → WebSphere application servers → *server_name***.

7. Go to the Business Process Choreographer configuration page: In the **Business Integration** section, expand **Business Process Choreographer** and click **Business Process Choreographer Containers**.

8. Verify that Business Process Choreographer is not configured. There should be a message indicating whether the Business Flow Manager is already installed.

    If the Business Flow Manager and Human Task Manager are already installed, perform "Removing the Business Process Choreographer configuration" on page 301 before continuing with the next step.

9. Enter the values and select the options that you planned for the Business Process Choreographer configuration on this server or cluster.

10. Click **Apply**. Information is displayed reporting the progress deploying and configuring Business Process Choreographer.

11. If the installation succeeded, click **Save Changes**. 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.

12. To create the database schema, 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 208 before you activate Business Process Choreographer in step 13.

    Note: If your database will exist by the time you activate Business Process Choreographer in step 9 on page 186, 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 208, the default schema will be created the first time that Business Process Choreographer accesses the database.

13. Activate Business Process Choreographer: Perform "Activating Business Process Choreographer" on page 296.

14. Optional: Verify that the basic Business Process Choreographer configuration works: Perform "Verifying that Business Process Choreographer works" on page 297.

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 or the URL prefix for the Business Process Choreographer Explorer, click either **Servers → Clusters → WebSphere application server clusters → *cluster_name*** or **Servers → Server Types → WebSphere application servers → *server_name***, then on the

Configuration tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, and make your changes.

- If you want to change the e-mail server address, port number, the user ID, or password for the e-mail server, click **Resources** → **Mail** → **Mail sessions**, select **Cell** scope, then click **HTM mail session_*suffix***, where *suffix* is either *node_name_server_name* or *cluster_name*, depending on where Business Process Choreographer is configured. 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 225.
    - If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 224.

17. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 231.

18. Optional: If you want to use group work items, use the administrative console to enable them. Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, then select **Enable group work items**.

19. If you have WebSphere application security enabled and you have a long-running process that calls a remote EJB method, make sure that your Common Secure Interoperability Version 2 (CSIv2) inbound authentication configuration has CSIv2 identity assertion enabled. For more information about this, refer to Configuring Common Secure Interoperability Version 2 inbound communications.

20. If you configured Business Process Choreographer in a network deployment environment:

    a. Map the Web modules for the BPEContainer and TaskContainer applications to a Web server to achieve load balancing and failover. You might need to change the default context roots for the REST API and the JAX Web Services API so that they are unique for each combination of host name and port. To set the context roots perform the following:

       1) To set the context roots for the Business Flow Manager, click **Applications** → **Application Types** → **WebSphere enterprise applications** then **BPEContainer_*suffix*** → **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules BFMRESTAPI and BFMJAXWSAPI are correct and unique.

       2) To set the context roots for the Human Task Manager, click **Applications** → **Application Types** → **WebSphere enterprise applications** then **TaskContainer_*suffix*** → **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules HTMRESTAPI and HTMJAXWSAPI are correct and unique.

b. If you changed any of the context roots for the REST API you must also
      modify the corresponding endpoints:

      1) If you use the Business Process Choreographer Explorer: Change the
         REST endpoint to match the new context roots by clicking either
         Servers → **Clusters** → **WebSphere application server clusters** →
         cluster_name or **Servers** → **Server Types** → **WebSphere application**
         servers → *server_name*, then on the **Configuration** tab, in the **Business**
         Integration section, expand **Business Process Choreographer**, and click
         Business Process Choreographer Explorer, then set the new value. For
         example, if the context root for the Business Flow Manager REST API
         is /rest/bpm/bfm then the full URL might be something like
         http://localhost:9080/rest/bpm/bfm.

      2) If you use the Business Space: Change the REST endpoints to match
         the new context roots by clicking either **Servers** → **Clusters** →
         WebSphere application server clusters → *cluster_name* or **Servers** →
         Server Types → **WebSphere application servers** → *server_name*, then on
         the **Configuration** tab, in the **Business Integration** section, expand
         Business Process Choreographer, and either **Business Flow Manager**
         or **Human Task Manager**, then under **Additional Properties** click
         REST Service Endpoint, and set the new value.

21. 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 234.

22. Optional: If you want to configure a remote Business Process Choreographer
    client that uses the WebSphere Process Server client, perform, "Configuring a
    remote client application" on page 292.

Results

Business Process Choreographer is configured.

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 "Planning to configure Business Process
   Choreographer" on page 121. If you run the script in batch mode, you must
   include all required parameters. If you run the script interactively, any
   required parameters that are not provided on the command line are prompted
   for. For detailed information about the script, examples, its options and
   parameters, see "bpeconfig.jacl script" on page 188.

Option	Description
If the server (or in a network deployment environment, the deployment manager) is not running	Use the option: -conntype NONE Do not use this option if the server (or deployment manager) is running.

Option	Description
If administrative security is enabled	Include the parameters: `-user  `*`userName`* `-password  `*`userPassword`*
If you are not using the default profile	Include the parameter: `-profileName `*`profileName`*
If you are not configuring Business Process Choreographer on the default server	Include either the parameter: `-cluster `*`clusterName`* or both parameters: `-node `*`nodeName`* `-server `*`serverName`*

Because the script always creates a Business Process Choreographer configuration	Include the parameters required for the Business Flow Manager and Human Task Manager:
	
	`{-adminUsers `*`userList`*` |`
	` -adminGroups `*`groupList`*`}`
	`[-adminJobUser `*`userID`*
	` -adminJobPwd `*`password`*`]`
	`{-monitorUsers `*`userList`*` |`
	` -monitorGroups `*`groupList`*`}`
	`-jmsBFMRunAsUser `*`userID`*
	`-jmsBFMRunAsPwd `*`password`*
	`-jmsHTMRunAsUser `*`userID`*
	`-jmsHTMRunAsPwd `*`password`*
	`-contextRootBFMWS `*`contextRootBFMWS`*
	`-contextRootBFMREST `*`contextRootBFMREST`*
	`-contextRootHTMWS `*`contextRootHTMWS`*
	`-contextRootHTMREST `*`contextRootHTMREST`*
	
	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: `-mailServerName `*`mailServerName`* If the mail server requires authentication, also include the parameters: `-mailUser `*`mailUserID`* `-mailPwd `*`mailPassword`*

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 `-createDB { yes | no }` 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. 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. **Restriction:** The script cannot create the following types of databases: • DB2 for z/OS • Oracle • A remote Microsoft SQL Server • A remote Informix Dynamic Server 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	If you used the database design tool to create a database design file, include the parameter
	
	`-bpcdbDesign databaseDesignFile`
	
	The values specified in the database design file have precedence over any occurrences of the following parameters included on the command line: -dbJava, -dbName, -dbPwd, -dbSchema, -dbServerName, -dbServerPort, -dbTablespaceDir, -dbType, -dbUser, and -driverType.If you do not specify a database design file, include the parameter:
	
	`-dbType databaseType`
	
	Also provide the parameters required for your database type, refer to "bpeconfig.jacl script" on page 188 for details.
	
	`-dbVersion version`
	`-dbHome databaseInstallPath`
	`-dbJava JDBCDriverPath`
	`-dbName databaseName`
	`-dbUser databaseUser`
	`-dbPwd databasePassword`
	`-dbAdmin databaseAdministratorUserID`
	`-driverType JDBCDriverType`
	`-dbTablespaceDir databaseTablespacePath`
	`-dbServerName databaseServerName`
	`-dbServerPort databaseServerPort`
	`-dbStorageGroup DB2zOSStorageGroup`
	`-dbConnectionTarget DB2zOSSubSystem`
	`-dbSchema schemaQualifier`
	`-dbInstance InformixInstance`
	
	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`
	
	Note: If you are using one of the following databases, bpeconfig.jacl can also create the database:
	• A local DB2 for Linux, UNIX, or Windows
	• DB2 on iSeries
	• Derby Embedded
	• Derby Network database and the server is running

Option	Description

Because every Business Process Choreographer configuration uses a JMS provider	Include the parameter:
	
	`-mqType { WPM | MQSeries }`
	
	Also provide the parameters required for your JMS provider (see "bpeconfig.jacl script" on page 188 for details).
	
	`-createQM { yes | no }`
	`-qmNameGet getQueueManagerName`
	`-mqClusterName mqClusterName`
	`-qmNamePut putQueueManagerName`
	`-mqHome MQInstallationDirectory`
	`-mqUser JMSProviderUserID`
	`-mqPwd JMSProviderPassword`
	
	Note: The MQSeries® option is deprecated.

If you are using the -mqType WPM option, and SCA uses a database as its message store, specify the Business Process Choreographer message engine store settings	Include the following parameters:
	
	`-mqCreateTables { true | false }`
	`-mqSchemaName mqSchemaName`
	`-medbUser meDatabaseUser`
	`-medbPwd meDatabasePassword`

Because the script always configures a Business Process Choreographer Explorer	Include any of these parameters:
	
	`-contextRootExplorer explorerContextRoot`
	`-explorerHost explorerURL`
	`-hostName explorerVirtualHostname`
	`-maxListEntries maximum`
	`-remoteCluster clusterName`
	`-remoteNode nodeName`
	`-remoteServer serverName`
	`-restAPIBFM restAPIURL*`
	`-restAPIHTM restAPIURL*`
	
	To configure the Business Process Choreographer Explorer reporting function, and event collector application, use the options:
	
	`-createEventCollector { yes | no }`
	`-reportFunction { yes | no }`
	`-reportAtSnapshotRange number`
	`-reportCreateTables { true | false }`
	`-reportDataSource jndiName`
	`-reportSchemaName schemaName`
	
	For more information about these parameters, including default values, see "bpeconfig.jacl script" on page 188.
	Note: * In a network deployment environment, `-restAPIBFM` and `-restAPIHTM` are required.
	Restriction: The option `-createEventCollector yes` is only supported when running the script in batch mode.

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 Explorer reporting function are already configured.

You can check if they are configured, by looking in the administrative console for enterprise applications with names that start with:

- BPCECollector
- BPCExplorer
- BPEContainer
- HTM_PredefinedTasksMsg
- HTM_PredefinedTasks
- 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 "Removing the Business Process Choreographer configuration" on page 301 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** ˃ **Server Types** ˃ **WebSphere application servers** ˃ *server_name*, 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** ˃ **WebSphere application server clusters** ˃ *cluster_name*, then in the **Business Integration** section, click **Service Component Architecture**.

   c. If it is not enabled, select **Support the Service Component Architecture components**, then click **Apply** and **Save**.

4. If you are using WebSphere Platform Messaging (WPM) as the JMS provider and SCA uses a database other than Derby Embedded as its message store 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 database design tool 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. 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 208 before you activate Business Process Choreographer in step 9.

   Note: If your database is local, and it will exist by the time you activate Business Process Choreographer in step 9, 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 208, the default schema will be created the first time that Business Process Choreographer accesses the database.

7. 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" on page 188.

8. If you are using the WebSphere MQ Java Message Service (JMS) provider, and you used the `-createQM no` option to prevent the script from creating the queue manager and queues, create the queue manager and queues now by performing "Creating the queue manager and queues for Business Process Choreographer" on page 204.

9. Activate Business Process Choreographer: Perform "Activating Business Process Choreographer" on page 296.

10. Optional: Verify that the basic Business Process Choreographer configuration works: Perform "Verifying that Business Process Choreographer works" on page 297.

11. 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.

12. 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 or the URL prefix for the Business Process Choreographer Explorer, click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, and make your changes.

    • If you want to change the e-mail server address, port number, the user ID, or password for the e-mail server, click **Resources** → **Mail** → **Mail sessions**, select **Cell** scope, then click **HTM mail session_suffix**, where *suffix* is either `node_name_server_name` or `cluster_name`, depending on where Business Process Choreographer is configured. Make your changes.

13. 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 225.

- If you are using the Virtual Member Manager (VMM), perform "Configuring the Virtual Member Manager people directory provider" on page 224.

14. Optional: If you configured VMM, and you want to use people substitution, perform "Configuring people substitution" on page 231.

15. Optional: If you want to use group work items, use the administrative console to enable them. Click either **Servers** ⟶ **Clusters** ⟶ **WebSphere application server clusters** ⟶ *cluster_name* or **Servers** ⟶ **Server Types** ⟶ **WebSphere application servers** ⟶ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, then select **Enable group work items**.

16. If you have WebSphere application security enabled and you have a long-running process that calls a remote EJB method, make sure that your Common Secure Interoperability Version 2 (CSIv2) inbound authentication configuration has CSIv2 identity assertion enabled. For more information about this, refer to Configuring Common Secure Interoperability Version 2 inbound communications.

17. 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 234.

18. If you configured Business Process Choreographer in a network deployment environment:

    a. Map the Web modules for the BPEContainer and TaskContainer applications to a Web server to achieve load balancing and failover. You might need to change the default context roots for the REST API and the JAX Web Services API so that they are unique for each combination of host name and port. To set the context roots perform the following:

       1) To set the context roots for the Business Flow Manager, click **Applications** ⟶ **Application Types** ⟶ **WebSphere enterprise applications** then **BPEContainer_***suffix*** ⟶ **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules BFMRESTAPI and BFMJAXWSAPI are correct and unique.

       2) To set the context roots for the Human Task Manager, click **Applications** ⟶ **Application Types** ⟶ **WebSphere enterprise applications** then **TaskContainer_***suffix*** ⟶ **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules HTMRESTAPI and HTMJAXWSAPI are correct and unique.

    b. If you changed any of the context roots for the REST API you must also modify the corresponding endpoints:

       1) If you use the Business Process Choreographer Explorer: Change the REST endpoint to match the new context roots by clicking either **Servers** ⟶ **Clusters** ⟶ **WebSphere application server clusters** ⟶ *cluster_name* or **Servers** ⟶ **Server Types** ⟶ **WebSphere application servers** ⟶ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**, then set the new value. For example, if the context root for the Business Flow Manager REST API is /rest/bpm/bfm then the full URL might be something like `http://localhost:9080/rest/bpm/bfm`.

2) If you use the Business Space: Change the REST endpoints to match the new context roots by clicking either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and either **Business Flow Manager** or **Human Task Manager**, then under **Additional Properties** click **REST Service Endpoint**, and set the new value.

19. Optional: If you want to configure a remote Business Process Choreographer client that uses the WebSphere Process Server client, perform, "Configuring a remote client application" on page 292.

Results

Business Process Choreographer is configured.

bpeconfig.jacl script

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 the Business Process Choreographer Explorer reporting function.

Location

The bpeconfig.jacl script file is located in the Business Process Choreographer `config` directory:

- **Linux** **UNIX** On Linux and UNIX platforms, it is in the directory *install_root*/ProcessChoreographer/config

- **Windows** On Windows platforms, it is 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.

Running the script in a stand-alone server environment

The configuration script is run using the wsadmin command. In a stand-alone server environment:

- Include the `-conntype NONE` option only if the application server is not running.
- If WebSphere administrative security is enabled, and your user ID does not have configurator or administrator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has configurator or administrator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

Running the script in a network deployment environment

The configuration script is run using the wsadmin command. 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, and your user ID does not have configurator or administrator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has configurator or administrator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

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

If you provide the necessary parameters on the command line, you are not prompted for them. To configure Business Process Choreographer, enter one of the following commands:

`Linux`　　`UNIX`　On Linux and UNIX platforms, if your current directory is *install_root,* enter the command:

```
bin/wsadmin.sh -f ProcessChoreographer/config/bpeconfig.jacl parameters
```

`Windows`　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:

```
-adminGroups groupList
-adminUsers userList
-adminJobPwd password
-adminJobUser userID
-bpcdbDesign databaseDesignFile
-cluster clusterName
-conntype NONE
-contextRootBFMWS contextRootBFMWS
-contextRootBFMREST contextRootBFMREST
```

```
-contextRootExplorer explorerContextRoot
-contextRootHTMWS contextRootHTMWS
-contextRootHTMREST contextRootHTMREST
-createDB { yes | no }
-createEventCollector { yes | no }
-createQM { yes | no }
-dbConnectionTarget DB2zOSSubSystem
-dbHome databaseInstallPath
-dbInstance InformixInstance
-dbJava JDBCDriverPath
-dbName databaseName
-dbPwd databasePassword
-dbSchema schemaQualifier
-dbServerName databaseServerName
-dbServerPort databaseServerPort
-dbStorageGroup DB2zOSStorageGroup
-dbTablespaceDir databaseTableSpacePath
-dbType databaseType
-dbUser databaseUser
-dbVersion version
-driverType JDBCDriverType
-explorerHost explorerURL
-hostName VirtualHostname
-jmsBFMRunAsPwd password
-jmsBFMRunAsUser userID
-jmsHTMRunAsPwd password
-jmsHTMRunAsUser userID
-mailPwd mailPassword
-mailServerName mailServerName
-mailUser mailUserID
-maxListEntries max
-medbPwd meDatabasePassword
-medbUser meDatabaseUser
-monitorGroups groupList
-monitorUsers userList
-mqClusterName mqClusterName
-mqCreateTables  { true | false }
-mqHome MQInstallationDirectory
-mqPwd JMSProviderPassword
-mqSchemaName mqSchemaName
-mqType JMSProviderType
-mqUser JMSProviderUserID
-node nodeName
-precompileJSPs { yes | no }
-qmNameGet getQueueManagerName
-qmNamePut putQueueManagerName
-remoteCluster clusterName
-remoteNode nodeName
-remoteServer serverName
-reportAtSnapshotRange number
-reportCreateTables { true | false }
-reportDataSource jndiName
-reportFunction { yes | no }
-reportSchemaName schemaName
-restAPIBFM restAPIURL
-restAPIHTM restAPIURL
-server serverName
```

Note: Some of these 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. If the same
parameter is specified more than once, the last value specified is used.

Parameters

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

-adminGroups *groupList*
: Where *groupList* is the list of names of groups, from the user registry, to which to map the BPESystemAdministrator and TaskSystemAdministrator Java EE 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 adminUsers or adminGroups options must be set.

-adminUsers *userList*
: Where *userList* is the list of names of users, from the user registry, to which to map the BPESystemAdministrator and TaskSystemAdministrator Java EE roles. 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 adminUsers or adminGroups options must be set.

-bpcdbDesign *databaseDesignFile*
: If you used the database design tool to create a database design file, use this option to specify the database design file, *databaseDesignFile*.

: The values specified in the database design file have precedence over any occurrences of the following parameters included on the command line: -dbJava, -dbName, -dbPwd, -dbSchema, -dbServerName, -dbServerPort, -dbTablespaceDir, -dbType, -dbUser, and -driverType.

-adminJobPwd *password*
: Where *password* is the password for the administration job user ID.

-adminJobUser *user ID*
: This user ID is used to run administrative jobs such as the Business Flow Manager and Human Task Manager cleanup services and the process instance migration tool. If you will not use these services, you do not need to specify the user ID.This user ID is optional. If you specify this user ID, it must be a member of both the BPESystemAdminstrator and TaskSystemAdministrator Java EE roles.

-cluster *clusterName*
: Where *clusterName* is the name of the cluster where Business Process Choreographer 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.

-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. This is a wsadmin parameter, if you do not specify it, you will not be prompted for it.

-contextRootBFMREST *contextRootBFMREST*
: Where *contextRootBFMREST* is the context root for the REST API endpoint URL. For the Business Flow Manager (BFM) the default context root on a server or a cluster is `/rest/bpm/bfm`.

-contextRootBFMWS *contextRootBFMWS*
: Where *contextRootBFMWS* 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`.

-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 for each combination of host name and port.

-contextRootHTMREST *contextRootHTMREST*

Where *contextRootHTMREST* is the context root for the REST API endpoint URL. For the Human Task Manager (HTM) the default context root on a server or a cluster is /rest/bpm/htm.

-contextRootHTMWS *contextRootHTMWS*

Where *contextRootHTMWS* 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*.

-createDB { yes | no }

Possible values are yes or no. If set to yes, the script creates 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.

-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 the Business Process Choreographer Explorer reporting function. When -createEventCollector has the value yes, you can use the -report* parameters to specify options for the Business Process Choreographer Explorer reporting function (previously known as the Business Process Choreographer Observer), for example, -reportDataSource to specify a separate reporting database rather than sharing the Business Process Choreographer BPEDB database. If you do not want the Business Process Choreographer event collector application to be installed, set the value of this parameter to no.

-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.

-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.

-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:

- Windows On Windows platforms, the default is *current_drive*\Program Files\IBM\SQLLIB where *current_drive* is the current drive letter and *dbHome* is the installation database home directory.
- Solaris 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:

- Windows On Windows platforms, the default is *current_drive*\Program Files\Informix where *current_drive* is the current drive letter and *dbHome* is the database home directory.
- Solaris   HP-UX On Solaris and HP-UX platforms, the default is /opt/informix.
- Linux   AIX On Linux and AIX platforms, the default is /usr/informix.

The file *dbHome*/jdbc/lib/ifxjdbc.jar must exist.

-dbInstance *InformixInstance*
 Where *InformixInstance* is the instance name for a Business Process Choreographer Informix database. The default value is ids1.

-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 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 either the Microsoft or the DataDirect driver type. There is no default value.
- Oracle. There is no default value.

Where *databaseInstallPath* is the installation directory for the database system.

When running the script in batch mode to configure 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, *databaseName* is the TNS.

- For Derby Network (not Derby Embedded) *databaseName* must be an absolute path name.
- For DB2 on i5/OS, it is the database name or IASP hardware device name, the default is *SYSBAS.

-dbPwd *databasePassword*
> Where *databasePassword* is the password for the user ID *databaseUser*.

-dbSchema *schemaQualifier*

> For DB2 on i5/OS, *schemaQualifier* is the collection name, the default value is BPEDB. *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.

-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 is configured for DB2, otherwise a type 4 JDBC provider is 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.

-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.

-dbType *databaseType*
> Where *databaseType* is the database type. This parameter 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

-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

-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 8 or 9.
- For MSSQL, *version* must have one of the following values:
    - If the database has no Unicode support, either 2500 or 2800.
    - If the database has Unicode support, either 2500U or 2800U.

-driverType *JDBCDriverType*

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

- For DB2 for Linux, UNIX, Windows, and z/OS platforms possible values are `Universal` or `DataServer`.
- For Derby: possible values are `Embedded`, `Embedded40`, `Network`, or `Network40`.
- For Oracle, possible values are `oci` or `thin`.
- For MSSQL, possible values are `Microsoft` or `DataDirect`.

-explorerHost *explorerURL*

Where *explorerURL* is the URL of the Business Process Choreographer Explorer. If this parameter is not specified for a non-cluster environment, 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.

-hostName *VirtualHostname*

Where *VirtualHostname* is the virtual host where the Business Process Choreographer Explorer, the Web service bindings of the Business Flow Manager and Human Task Manager APIs, and the REST bindings of the Business Flow Manager and Human Task Manager APIs will run. The default value is `default_host`.

-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.

-jmsBFMRunAsUser *user ID*

Where *user ID* is the run-as user ID from the user registry for the Java EE role JMSAPIUser. This property is required to configure the business process 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.

-jmsHTMRunAsUser *user ID*

Where *user ID* is the run-as user ID from the user registry for the Java EE role EscalationUser. This property is required to configure the human task container. This parameter has no default value. It must be set.

-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.

-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.

-maxListEntries *maximum*

Where *maximum* is the maximum number of results that the Business Process Choreographer Explorer returns for a query. The default is 10000.

-medbPwd *MEDBPassword*

Where *MEDBPassword* is the password for the user ID that is provided for the medbUser parameter. This parameter has no default value.

-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 SCA is using a database, and the messaging engine database is not accessed through the Derby Embedded JDBC provider.

-monitorGroups *groupList*

Where *groupList* is the list of names of groups, from the user registry, to which to map the BPESystemMonitor and TaskSystemMonitor Java EE roles. 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 monitorUsers or monitorGroups must be set.

-monitorUsers *userList*

Where *userList* is the list of names of users, from the user registry, to which to map the BPESystemMonitor and TaskSystemMonitor Java EE roles. 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 monitorUsers or monitorGroups must be set.

-mqType *JMSProviderType*

Where *JMSProviderType* is the type of Java Message Service (JMS) provider to

use for Business Process Choreographer. This parameter 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.

-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.

-mqCreateTables *{ true | false}*
This Boolean parameter only has an effect when the mqType option is set to `WPM` and the Service Component Architecture (SCA) is using a database for its message store rather than using a FILESTORE. This parameter controls whether the default JMS provider automatically creates its tables in the message engine database upon the first connection. The default is inherited from the SCA setting, you can use this parameter to override the default.

-mqHome *MQInstallationDirectory*
Where *MQInstallationDirectory* is the installation directory of WebSphere MQ. This parameter 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 **Windows platforms:**
current_drive\Program Files\IBM\WebSphere MQ

AIX **AIX:**
/usr/mqm

Solaris HP-UX Linux **Solaris, HP-UX, and Linux:**
/opt/mqm

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

-mqSchemaName *mqSchemaName*
Where *mqSchemaName* is the name of the database schema for the default JMS provider's messaging engine. This parameter only has an effect when SCA uses a database as its message store, rather than using a FILESTORE. Business Process Choreographer will use the same database as SCA, but uses a different schema. You can use this parameter to override the default schema name. The default value for Oracle databases is the user ID specified for *medbUser*, for other databases, the default is a generated value, for example WPRBM00.

-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 Service Integration (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`.

-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.

-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.

-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.

-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`.

-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.

-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.

-reportAtSnapshotRange *number*
> A snapshot report is built by evaluating all events that are older than the qualifying snapshot date and time. This optional parameter defines the number of days for which events can be included in a snapshot report. Only events that have been emitted within this period are evaluated by the snapshot report. The default is 60 days. This optional parameter only has an effect if the reporting function is enabled using the option `-reportFunction yes`.
>
> 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.

-reportCreateTables { true | false }

This optional parameter indicates if the Business Process Choreographer Explorer reporting function schema is created when Business Process Choreographer Explorer connects to the database the first time. The default is true. This optional parameter only has an effect if the reporting function is enabled using the option `-reportFunction yes`.

-reportDataSource *jndiName*

Where *jndiName* is the JNDI name of the data source JNDI that is used to connect to the database. Mandatory parameter when `-reportFunction yes` is specified. The data source is not created automatically.

-reportFunction { yes | no }

This optional parameter controls whether the Business Process Choreographer Explorer reporting function is enabled. In interactive mode, the default is no. In batch mode, for compatibility with earlier versions, the default is yes.

-reportSchemaName *schemaName*

This optional parameter identifies the database schema that is used as a prefix for all reporting database objects. If you specify no schema name, a unique schema name is generated. This optional parameter only has an effect if the reporting function is enabled using the option `-reportFunction yes`. The default value is WPRBC00.

-restAPIBFM *restAPIURL*

Where *restAPIURL* is the URL for the Business Flow Manager REST API, which is needed to support the graphical process widget for the Business Process Choreographer Explorer. On a standalone server, the default is computed, for example, `http://localhost:9080/rest/bpm/bfm`. In a network deployment environment there is no default value.

-restAPIHTM *restAPIURL*

Where *restAPIURL* is the URL for the Human Task Manager REST API, which is needed to support the graphical process widget for the Business Process Choreographer Explorer. On a standalone server, the default is computed, for example, `http://localhost:9080/rest/bpm/htm`. In a network deployment environment there is no default value.

-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.

Example: Configuring a stand-alone server non-interactively

To configure Business Process Choreographer on a stand-alone server on a Windows platform, using a DB2 database, the batch mode command might be like the following:

```
wsadmin -conntype none -f bpeconfig.jacl
     -adminGroups bpcadmins -monitorGroups bpcmonitors
     -createDB no
     -createEventCollector no
     -dbSchema WPRBE00 -dbUser db2user -dbPwd secret
     -dbServerName db2host.acme.com -dbJava d:\\programs\\IBM\\SQLLIB\\java
     -dbTablespaceDir d:\\DB2\\tablespacedir -mqType WPM
     -dbType DB2 -dbName BPEDB
     -driverType Universal
     -explorerHost http://wpshost.acme.com:80/bpc
     -jmsBFMRunAsUser jmsuser -jmsBFMRunAsPwd secret
     -jmsHTMRunAsUser escalationuser -jmsHTMRunAsPwd secret
     -mailServerName smtphost.acme.com -mailUser {}
     -mqCreateTables true
     -mqSchemaName WPRBM00
```

```
      -mqUser sibuser -mqPwd secret
      -reportFunction no
      -restAPIBFM http://wpshost.acme.com:80/rest/bpm/bfm
      -restAPIHTM http://wpshost.acme.com:80/rest/bpm/htm
```

For other platforms, only the file system paths would be different.

Example: Configuring on a cluster non-interactively

To configure Business Process Choreographer on a cluster "cluster1", consisting of two nodes; a Windows node "Node01" and a UNIX node "Node02", using a DB2 database, the batch mode command might be like the following:

```
wsadmin -conntype none -profileName Dmgr01 -f bpeconfig.jacl
      -adminUsers bpcadmins
      -cluster cluster1
      -contextRootBFMREST /rest/bpm/bfm
      -contextRootBFMWS /BFMIF_cluster1
      -contextRootExplorer /bpc
      -contextRootHTMWS /HTMIF_cluster1
      -contextRootHTMREST /rest/bpm/htm
      -createEventCollector no
      -dbJava.acmeNode01 "c:\\Progam Files\\IBM\\SQLLIB\\java"
      -dbJava.acmeNode02 /home/db2inst1/sqllib
      -dbName BPEDB62
      -dbSchema WPRBE00
      -dbType DB2
      -dbUser db2user -dbPwd secret
      -createDB no
      -explorerHost http://wps.acme.com/bpc
      -jmsBFMRunAsUser jmsuser -jmsBFMRunAsPwd secret
      -jmsHTMRunAsUser escalationuser -jmsHTMRunAsPwd secret
      -mailServerName smtphost.acme.com -mailUser {}
      -maxListEntries 5000
      -medbUser db2user
      -monitorUsers bpcmonitors
      -mqCreateTables true
      -mqSchemaName WPRBM00
      -mqType WPM -hostName default_host
      -mqUser sibuser -mqPwd secret
      -reportFunction no
      -restAPIBFM http://wps.acme.com/rest/bpm/bfm
      -restAPIHTM http://wps.acme.com/rest/bpm/htm
```

For other platforms, only the file system paths would be different.

Example: Configuring interactively

This example, illustrates running the bpeconfig.jacl script to install and configure Business Process Choreographer using an existing DB2 database, a human task container, and a Business Process Choreographer Explorer. This example assumes that you are using the default profile and that either administrative security is not enabled, or that your user ID has operator or administrator authority.

Restriction: When run interactively, this script cannot configure the Business Process Choreographer Explorer reporting function, nor the necessary event collector application. If you want to use the Business Process Choreographer Explorer reporting function, perform "Configuring the Business Process Choreographer Explorer reporting function and event collector" on page 240.

1. On the server, or for network deployment, on the deployment manager, start the script:

   - **Linux** **UNIX** On Linux and UNIX platforms, enter the command:

     install_root/bin/wsadmin.sh -f *install_root*/ProcessChoreographer/config/bpeconfig.jacl

   - **Windows** On Windows platforms, enter the command:

     install_root\bin\wsadmin.bat -f *install_root*\ProcessChoreographer\config\bpeconfig.jacl

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 `Run-as UserId for administrative jobs (may be empty; if set, must be a BPESystemAdminstrator):`, you can enter the run-as user ID that will be used for administrative jobs such as cleanup service jobs and process instance migration. If you will not use these services, you can leave it empty. If you specify a user ID, it must be a member of the BPESystemAdminstrator and TaskSystemAdministrator Java EE roles.

j. If you specified a user ID for administrative jobs, enter the password for the user ID.

k. For the question `Use WebSphere default messaging or WebSphere MQ [WPM/MQSeries]?`, select the JMS provider that you want to use.

l. 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*.

3) For the question `Context root for the REST API [/rest/bpm/bfm]:`, press **Enter** to accept the default value /rest/bpm/bfm.

m. For the question `Create the DataSource for the Process Choreographer database?`, enter **Yes**.

n. 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.

o. Enter the database name.

p. At the `Database schema name (may be empty)` prompt, hit **Enter** to use the implicit schema qualifier.

q. For the question `DB2 User ID`, enter the user ID to access the database.

r. Enter the password for the database user ID.

s. 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.

t. For the question `Database server port`, enter the database server port, for example, 50000.

u. At the `JDBC driver directory on [yourHost]` prompt, enter the directory where the DB2 JDBC driver JAR files are located.

v. 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' executable 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)` enter **Yes** if the BPEDB database already exists, otherwise enter **No**.

w. 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.

x. Enter the password for the service integration bus authentication user ID.

y. For the question `Message store type to use [DATASTORE/FILESTORE]: DATASTORE`, select which type of store to use for messages.

z. For the question `Messaging engine database schema qualifier [WPRBM00]: WPRBM00`, press **Enter** to accept the default schema qualifier for the messaging engine database, or enter a different one.

aa. For the question `Automatically create the database tables when the messaging engine connects for the first time [True/false]? true`, press **Enter** to accept the default that the database tables for the messaging engine the first time that it connects to the database, or enter **false** if you will create them manually.

ab. For the question `Install the task container?`, enter **Yes**.

ac. 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.

ad. 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.

ae. 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.

af. For the question `Group(s) to add to role TaskSystemMonitor`, enter the groups from the domain user registry that are mapped onto the role of task monitor.

ag. For the question `Run-as UserID for role EscalationUser`, enter the run-as user ID for the role of escalation user, for example `db2admin`.

ah. 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 201.

ai. For the question `Context root for the SCA Web Service [/HTMIF_PNODE_server1]:`, press **Enter** to accept the default value `HTMIF_nodeName_serverName`.

aj. For the question `Context root for the REST API [/rest/bpm/htm]:`, press **Enter** to accept the default value `/rest/bpm/htm`.

ak. 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.

al. 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.

am. 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.

an. For the question `Maximum number of list entries for the Process Choreographer Explorer`, press **Enter** to use the default value 10000.

ao. The following reminder is displayed:
```
**************************************************************************
* NOTE: The Process Choreographer REST API URLs are needed by the
* Process Choreographer Explorer's graphical process widget.
**************************************************************************
```

ap. For the question `URL for the Business Flow Manager REST API`, press **Enter** to use the default value http://*host_name*:9080/rest/bpm/bfm.

aq. For the question `URL for the Human Task Manager REST API`, press **Enter** to use the default value http://*host_name*:9080/rest/bpm/htm.

ar. For the question `Enable the reporting function (formerly known as 'Observer') [No/yes]?` no, press **Enter** to accept the default that the Business Process Choreographer Explorer reporting function will not be enabled . Otherwise, enter **Yes** to enable it.

as. 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.
* To clear the default value, enter a space character.
**************************************************************************
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.

at. A reminder is displayed about where to find the script files that you can use to configure the Business Process Choreographer Explorer reporting function.

To interactively configure the EventCollector, please use the script setupEventCollector 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 – Unless you used the wsadmin `-tracefile` parameter to specify a different file name.

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

- `  Linux  ` `  UNIX  ` On Linux and UNIX platforms: In the directory `profile_root`/logs

- `  Windows  ` 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 180
Describes how to use the bpeconfig.jacl script to configure Business Process Choreographer and all the necessary resources on a given server or cluster.

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:

      `  Linux  ` `  UNIX  ` On Linux and UNIX platforms, enter:

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

      `  Windows  ` On Windows platforms, enter:

      ```
      cd install_root\ProcessChoreographer\config
      createQueues.bat 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 to the server that hosts the central queue manager:

      - `Linux` `UNIX` If your central queue manager is on a Linux or UNIX server, copy the file: `createQueues.sh`

      - `Windows` If your central queue manager is on a Windows workstation, copy the file: `createQueues.bat`

   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:

      `Linux` `UNIX` On Linux and UNIX platforms, enter:

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

      `Windows` On Windows platforms, enter:

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

      Where *queueManager* is the name to give the new queue manager.

   d. Add a listener for the new queue manager:

      `Linux` `UNIX` On Linux and UNIX platforms, enter:

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

      `Windows` On Windows platforms, enter:

      ```
      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

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.

Using a generated SQL script to create the database schema for Business Process Choreographer

When you configure Business Process Choreographer, an SQL script is generated that creates the database objects for Business Process Choreographer.

Before you begin

You have used the administrative console or the bpeconfig.jacl script to configure Business Process Choreographer. If you used the bpeconfig.jacl script to configure Business Process Choreographer, 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 `createSchema.sql` SQL script.

   - If you configured Business Process Choreographer in a network deployment environment using the administrative console or by running the `bpeconfig.jacl` script in connected mode, the `createSchema.sql` script file will be generated on the node of the deployment manager.

   - If you configured Business Process Choreographer on a standalone server using the administrative console or by running the `bpeconfig.jacl` script in connected mode, the `createSchema.sql` script file will be generated on the node where you invoked wsadmin.

   - If you configured Business Process Choreographer by running the `bpeconfig.jacl` script in disconnected mode, the `createSchema.sql` script file will be generated on the node of the standalone server.

   - `Linux` `UNIX` For Linux and UNIX platforms:

      - If you specified a schema qualifier, the generated script is:
        profile_root/dbscripts/ProcessChoreographer/*database_type*/ *database_name*/*database_schema*/createSchema.sql.

      - If you did **not** specify a schema qualifier, the generated script is:
        profile_root/dbscripts/ProcessChoreographer/*database_type*/ *database_name*/createSchema.sql.

   - `Windows`

   - For Windows platforms:

      - If you specified a schema qualifier, the generated script is:
        profile_root\dbscripts\ProcessChoreographer*database_type*\ *database_name**database_schema*\createSchema.sql

      - If you did **not** specify a schema qualifier, the generated script is:
        profile_root\dbscripts\ProcessChoreographer*database_type*\ *database_name*\createSchema.sql

   Note: For SQL Server, there is also a version named `createSchemaUnicode.sql`, which you should use if your database is configured for Unicode.

   Where:

   database_type
   > is one of the following strings, which identify the database systems that are supported by the generated scripts:
   > - DB2
   > - DB2zOSV8
   > - DB2zOSV9
   > - 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 the database does not yet exist, get your database administrator to create the database and user IDs according to the values you planned in "Planning the BPEDB database" on page 144 and "Planning security, user IDs, and authorizations" on page 136.

   Note: This step is not necessary if your database is one of the following, because the generated script will create the database:
   - Derby Embedded
   - Derby Network and the database server is running
   - DB2 on iSeries
   - DB2 for Linux, UNIX, or Windows local database

3. If the database is remote, copy the generated script to the database host. 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 can customize the SQL script:
   a. If you used the administrative console to configure Business Process Choreographer, substitute actual values for the following placeholders:
      - For DB2 on z/OS: Replace @STOGRP@ with the storage group name, the default value is SYSDEFLT.
      - Replace @location@ (or for Oracle, replace &1) with the table space directory.
   b. For a high-performance system specify the allocation of disks and table spaces that you planned in step 5 on page 146 of "Planning the BPEDB database" on page 144.

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:
	`db2 -tf createSchema.sql`
	
	For the EBCDIC version:
	`db2 -tf createSchema.ddl`

For a Derby database	`java -Dij.protocol=jdbc:derby:`
	`      -Dij.database=database_name`
	`      org.apache.derby.tools.ij`
	`      createSchema.sql`

For an Informix database	`dbaccess database_name createSchema.sql`
For an Oracle database	`sqlplus userID/password` `      @database_name@createSchema.sql`

For an SQL Server database	For an ASCII database:
	`sqlcmd -U userID`
	`        -P password`
	`        -d database_name`
	`        -i createSchema.sql`
	
	For a Unicode database:
	`sqlcmd -U userID`
	`        -P password`
	`        -d database_name`
	`        -i createSchemaUnicode.sql`

6. For all existing Business Process Choreographer configurations, configure Java Database Connectivity (JDBC) to access the database remotely: Perform the following steps:

   - On each node that hosts a member of the cluster where you configured Business Process Choreographer.
   - On any server that runs Business Process Choreographer.

   a. If the database server is different to the Business Process Choreographer server, 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. The database must be catalogued and accessible through an alias name.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.

Using SQL scripts to create the database for Business Process Choreographer

You might choose to create the database for Business Process Choreographer manually before you configure Business Process Choreographer, or even before you have installed the product.

Before you begin

You have performed "Planning the BPEDB database" on page 144.

About this task

Your organization might require that databases be created by a separate database administrator. If you use the administrative console or the bpeconfig.jacl script to configure Business Process Choreographer, customized SQL scripts are generated that you can give to your DBA to create the BPEDB database. However, if you want to create the database before configuring Business Process Choreographer, or even before product installation, your DBA must use the non-customized SQL scripts. This topic describes how to use the non-customized SQL scripts, which are available on the product media.

Procedure

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."
- "Creating a DB2 for i5/OS database for Business Process Choreographer" on page 213.
- "Creating a DB2 for Linux, UNIX, and Windows database for Business Process Choreographer" on page 214.
- "Creating a DB2 for z/OS database for Business Process Choreographer" on page 216.
- "Creating an Informix Dynamic Server database for Business Process Choreographer" on page 219.
- "Creating a Microsoft SQL Server database for Business Process Choreographer" on page 220.
- "Creating an Oracle database for Business Process Choreographer" on page 221.

Results

The Business Process Choreographer database exists.

Creating a Derby database for Business Process Choreographer

Only use this task if you want to create a Derby database for Business Process Choreographer before you configure Business Process Choreographer, or before you have installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144. The Derby database is installed with the WebSphere Process Server. However, if you want to create the

database before installing the product, you must already have a Derby installation on your database server.

About this task

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

Procedure

1. If you want to create a Derby Network Server database, rather than a Derby Embedded database, make sure that the for Derby Network Server is running, and that you have planned to use the Derby Network Server JDBC provider.
2. Create the parent directory for the database 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.
     - ▆▆Linux▆▆ ▆▆UNIX▆▆ On Linux and UNIX platforms create *profile_root*/databases.
     - ▆▆Windows▆▆ On Windows 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.
3. Copy the database creation script to the directory that you created in step 2. The script is located in the following directories:
   - ▆▆Linux▆▆ ▆▆UNIX▆▆ On Linux and UNIX platforms:
     - Location on the product media: *media_root* or *extract_directory*/ dbscripts/ProcessChoreographer/Derby/createDatabase.sql
     - Location after installation: *install_root*/dbscripts/ ProcessChoreographer/Derby/createDatabase.sql
   - ▆▆Windows▆▆ On Windows platforms:
     - Location on the product media: *media_root* or *extract_directory*\ dbscripts\ProcessChoreographer\Derby\createDatabase.sql
     - Location after installation: *install_root*\dbscripts\ ProcessChoreographer\Derby\createDatabase.sql
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

Only use this task if you want to create the DB2 for i5/OS database schema for Business Process Choreographer before you configure Business Process Choreographer, or before you have installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144.

Procedure

1. On the system 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 system that hosts the database. The script is located in the following directories:

   - `Linux` `UNIX` On Linux and UNIX platforms:
     - Location on the product media: *media_root* or *extract_directory*/ `dbscripts/ProcessChoreographer/DB2iSeries/createSchema.sql`
     - Location after installation: *install_root*/dbscripts/ `ProcessChoreographer/DB2iSeries/createSchema.sql`

   - `Windows` On Windows platforms:
     - Location on the product media: *media_root* or *extract_directory*\ `dbscripts\ProcessChoreographer\DB2iSeries\createSchema.sql`
     - Location after installation: *install_root*\dbscripts\ `ProcessChoreographer\DB2iSeries\createSchema.sql`

3. Customize a copy of the SQL file `createSchema.sql` according to the values you planned in "Planning the BPEDB database" on page 144, "Planning security, user IDs, and authorizations" on page 136, and the instructions in the header of the file.
4. Create the database objects. In a qshell environment, run your customized script. For example, if the script is in your current directory, enter the command:

   `db2 -tf createSchema.sql`

5. Plan to use the Toolbox JDBC driver. Copy the JAR file `/QIBM/ProdData/HTTP/ Public/jt400/lib/jt400.jar` from the database host to the WebSphere Process Server.

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

Only use this task if you want to create a DB2 database for Business Process Choreographer before you configure Business Process Choreographer, or before you have installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144.

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. Copy all the Business Process Choreographer database SQL script files to the server that hosts the database. The scripts are located in the following directories:

- **Linux** **UNIX** On Linux, and UNIX platforms:
  - Location on the product media: *media_root* or *extract_directory*/ `dbscripts/ProcessChoreographer/DB2`
  - Location after installation: *install_root*/dbscripts/ `ProcessChoreographer/DB2`

- **Windows** On Windows platforms:
  - Location on the product media: *media_root* or *extract_directory*\ `dbscripts\ProcessChoreographer\DB2`
  - Location after installation: *install_root*\dbscripts\ `ProcessChoreographer\DB2`

4. Change to the directory where you copied the SQL scripts.

5. If you want to use an existing database, skip to step 10.

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:

- **AIX** On AIX systems, enter the command:

  `/usr/opt/db2_08_01/adm/db2licm -l`

- **Solaris** **HP-UX** **Linux** 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. 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.

9. Optional: If you only want to create a non-production database, named BPEDB, using default settings, for stand-alone development, evaluation, or demonstration purposes:

   a. Enter the following command:

      `db2 -tf createDatabase.sql`

   b. Skip to step 11 on page 216.

10. If the database is for a production system, create the table space and schema:

    a. Make sure that you use a user ID that has administrator rights for the database system.

    b. Customize a copy of the `createTablespace.sql` table space creation script according to the instructions in the file's header, using the values that you planned in "Planning the BPEDB database" on page 144.

    c. 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 Business Process Choreographer.

    d. Make sure that you are attached to the correct instance. Check the DB2INSTANCE environment variable.

e. To connect to a database named *database_name*, in the DB2 command-line processor, enter the command:

```
db2 connect to database_name
```

f. 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.

g. Customize a copy of the `createSchema.sql` schema creation script according to the instructions in the file's header, using that values that you planned in "Planning the BPEDB database" on page 144.

h. 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.

11. On each application server that remotely accesses the database:

a. Catalog the database by entering the command:

```
db2 catalog database database_name as database_alias at node node_name
```

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 database_alias user user_ID using password
db2 connect reset
```

Results

The database for Business Process Choreographer exists.

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

Only use this task if you want to create a DB2 for z/OS database for use by WebSphere Process Server Business Process Choreographer that is running on a Linux, UNIX, or Windows platform and have not yet configured Business Process Choreographer, or you have not yet installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144.

About this task

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

Procedure

1. Optional: You have already installed WebSphere Process Server on a UNIX, Linux, or Windows server.

2. Copy all the Business Process Choreographer database script files to the z/OS server that hosts the database. The scripts are located in the following directories:

   - `Linux` `UNIX` On Linux and UNIX platforms:
     - Location on the product media: *media_root* or *extract_directory*/ dbscripts/ProcessChoreographer/*database_type*
     - Location after installation: *install_root*/dbscripts/ ProcessChoreographer/*database_type*

   - `Windows` On Windows platforms:
     - Location on the product media: *media_root* or *extract_directory*\ dbscripts\ProcessChoreographer*database_type*
     - Location after installation: *install_root*\dbscripts\ ProcessChoreographer*database_type*

   Where *database_type* is one of the following:
   - DB2zOSV8
   - DB2zOSV9

3. 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, using the values that you planned in "Planning the BPEDB database" on page 144, then run the script. To run the script, enter the command:

        ```
        db2 -tf createDatabase.sql
        ```

   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. This 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.

   h. Customize a copy of the `createTablespace.sql` table space creation script according to what you planned in "Planning the BPEDB database" on page 144 and the instructions in the header. Replace @STOGRP@ with the storage group name and replace @DBNAME@ with the database name (not the subsystem name).

   i. Run your customized version of the table space creation script. For example, to run the script, enter the command:

      ```
      db2 -tf createTablespace.sql
      ```

If you want to drop the table space, customize and run the `dropTablespace.sql` script.

   j. Edit the `createSchema.sql` create schema script according to what you planned in "Planning the BPEDB database" on page 144 and the instructions in the header.

      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.

  k. Run your customized version of the create schema script. For example, to run the script, enter the command:

```
db2 -tf createSchema.sql
```

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.

4. Optional: On any server that will host a WebSphere Process Server Business Process Choreographer configuration:

  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'
```

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 system.

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
```

  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. Verify that you can establish a connection to the remote subsystem by entering the following command:

```
db2 connect to database_alias user user_ID using password
db2 connect reset
```

Results

The database for Business Process Choreographer exists.

Creating an Informix Dynamic Server database for Business Process Choreographer

Only use this task if you want to create an Informix Dynamic Server database for Business Process Choreographer before you configure Business Process Choreographer, or before you have installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144.

Procedure

1. Install the Informix server on the computer that hosts the database.
2. Create an Informix server instance. Make sure that the following Informix environment variables are set correctly:
   - *INFORMIXSERVER* must point to the new instance
   - *ONCONFIG* must point to the configuration file for the instance.
   - The environment variables relating to Global Language Support (GLS) must be set to Unicode (UTF-8) support, which is required to store all the characters that can be handled in Java code.

   For more details about the different environment variables, refer to the Informix Dynamic Server documentation.
3. Copy and configure the Java Database Connectivity (JDBC) driver on all remote application servers that use the database server.
4. Copy all the Business Process Choreographer database script files to the server that hosts the database.

   - `Linux` `UNIX` On Linux, and UNIX platforms, copy all the SQL and SH files:
     - Location on the product media: *media_root* or *extract_directory*/ dbscripts/ProcessChoreographer/Informix
     - Location after installation: *install_root*/dbscripts/ ProcessChoreographer/Informix

   - `Windows` On Windows platforms, copy all the SQL and BAT files:
     - Location on the product media: *media_root* or *extract_directory*\ dbscripts\ProcessChoreographer\Informix
     - Location after installation: *install_root*\dbscripts\ ProcessChoreographer\Informix

5. Change to the directory where you copied the files.
6. If you want to create a non-production database using default settings that is suitable for standalone development, evaluation, or demonstration 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.

      `Linux` `UNIX` On Linux and UNIX systems, read the instructions in the `createDbspace.sh` file. Adjust the value parameters in the script to values appropriate for your environment, then run the file.

      `Windows` On Windows systems, read the instructions in the `createDbspace.bat` file. Adjust the value parameters in the script to values appropriate for your environment, then run the file.
   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

Only use this task if you want to create a Microsoft SQL Server database for Business Process Choreographer before you configure Business Process Choreographer, or before you have installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144.

Procedure

1. Install a Microsoft SQL Server, on the server that hosts the database. Make sure that the following requirements are met:
   - The server must support Unicode.
   - The database server must be configured for distributed transactions.
   - The instance must be 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.

   For more information about these configuration options, refer to the documentation for Microsoft SQL Server.
2. Make sure that the database server and the Distributed Transaction Coordinator (DTC) are running.
3. Copy all the Business Process Choreographer database SQL script files to the server that hosts the database. The scripts are located in the following directories:
   - `Windows` For installations on Windows platforms:
     – Location on the product media: *media_root* or *extract_directory*\ dbscripts\ProcessChoreographer\SQLServer

– Location after installation: *install_root*\dbscripts\
    ProcessChoreographer\SQLServer
- `Linux` `UNIX` For installations on Linux, and UNIX platforms, which
    will access a remote Microsoft SQL Server database:
    – Location on the product media: *media_root* or *extract_directory*/
        dbscripts/ProcessChoreographer/SQLServer
    – Location after installation: *install_root*/dbscripts/
        ProcessChoreographer/SQLServer
4. 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, customize a copy of one of the following scripts, as
            described in the header of the file, using the values you planned, then
            run it.
            – `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

Only use this task if you want to create an Oracle database for Business Process
Choreographer before you configure Business Process Choreographer, or before
you have installed the product.

Before you begin

You completed "Planning the BPEDB database" on page 144.

Procedure

1. Install the Oracle server on the computer that hosts the database. Make sure that you are using the 32-bit Oracle libraries that are located in the `lib32` subdirectory.

2. `Linux` `UNIX` On Linux and UNIX systems:,

   a. Make sure that the environment variables *ORACLE_BASE* and *ORACLE_HOME* are set for the root user.

   b. If you are using Oracle 10g, create a soft link to the Oracle library `libclnt.so.10.1` in the `/usr/lib` directory.

3. Check the class path to be sure that your JDBC driver is using the correct JAR file:

   - For Oracle 10g, use the ojdbc5.jar file.
   - For Oracle 11g, use the ojdbc5.jar file.

4. 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.

5. Start the Oracle listener by entering the command:

   `lsnrctl start`

6. 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.

7. Copy all the Business Process Choreographer database SQL script files to the server that hosts the database. The scripts are located in the following directories:

   - `Linux` `UNIX` On Linux, and UNIX platforms:
     - Location on the product media: *media_root* or *extract_directory*/`dbscripts/ProcessChoreographer/Oracle`
     - Location after installation:*install_root*/`dbscripts/ProcessChoreographer/Oracle`

   - `Windows` On Windows platforms:
     - Location on the product media:*media_root* or *extract_directory*\`dbscripts\ProcessChoreographer\Oracle`
     - Location after installation:*install_root*\`dbscripts\ProcessChoreographer\Oracle`

8. Change to the directory where you copied the SQL scripts.

9. 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 4.

10. 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.

11. Customize a copy of the `createTablespace.sql` table space creation script according to what you planned in "Planning the BPEDB database" on page 144 and the instructions in the header of the script file.

12. 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.

13. 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.

14. 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`.

15. 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

Business Process Choreographer can use one of four people directory providers to determine who can start processes or claim activities or tasks. Two providers can be used in their default configurations (Local operating system user registry, WebSphere Application Server user registry). The Virtual Member Manager and LDAP people directory providers can usually be used in its default configuration, but in some cases, it must be configured.

About this task

Each type of supported people directory service requires a corresponding people directory provider plug-in. Table 27 lists the supported people directory providers and corresponding plug-ins, which are installed by default.

Table 27. Supported people directory providers and their plug-ins

People directory provider	People directory provider plug-in name
Virtual Member Manager (VMM)	VMM people directory provider
Lightweight Directory Access Protocol (LDAP) directory	LDAP people directory provider
Local operating system user registry	System people directory provider
WebSphere Application Server user registry	User Registry people directory provider

• To use VMM and LDAP, you will probably need to customize the configuration before using it, as described in the following topics.

• You can use the user registry and system plug-ins with the default configurations. Because you can use these people directory providers without further customization, they are often ideal for development and test environments.

Configuring the Virtual Member Manager people directory provider

You configure the Virtual Member Manager (VMM) people directory provider so that Business Process Choreographer can perform people assignment, which determines who can start processes or claim activities or tasks. The default configuration of the VMM people directory provider is ready to use, and only needs to be configured if you introduce custom people assignment criteria.

Before you begin

To configure the VMM people directory provider, you must have a configured federated repository.

Procedure

1. Make a copy of the standard transformation file for VMM, and give it another name, for example, `myVMMTransformation.xsl`.

   - **`Linux`** **`UNIX`** On Linux and UNIX platforms standard XSL transformation for VMM is located in *`install_root`*`/ProcessChoreographer/Staff/VMMTransformation.xsl`

   - **`Windows`** On Windows platforms, standard XSL transformation for VMM is located in *`install_root`*`\ProcessChoreographer\Staff\VMMTransformation.xsl`

2. Adapt the copy of the transformation file to suit the schema for your organization, as described in "Adapting the LDAP transformation file" on page 227.

   CAUTION:
   Do not modify the original version of the transformation file because it can be overwritten without warning when you apply a service pack or fix pack.

3. If Business Process Choreographer is configured on a cluster, place the copy of the transformation file in the `ProcessChoreographer/Staff` directory to make it 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 from the following list:

Option	Description
For a standalone profile	Only one node is displayed.
In a network deployment environment, where Business Process Choreographer is configured on one 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 (perform step 6) 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 (step 6) for all of the other nodes that host members of the cluster.

6. 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 the copy of the Extensible Stylesheet Language (XSL) transformation file that you adapted in step 2 on page 224. If the node agent is running, you can browse the file system of remote nodes to select the file.

d. Click **Next** to copy the file to the `ProcessChoreographer/Staff` directory on the selected node.

e. Enter an administrative name for the new people directory configuration, and optionally, a description

f. Enter a unique Java Naming and Directory Interface (JNDI) name to identify this configuration to the system, for example, `bpe/staff/ myvmmconfiguration`.

   Note: There are no other configuration parameters

g. Click **OK**, then click **Save**.

7. To activate the provider configuration, stop and start the server or servers where you configured the provider.

8.

Results

The VMM people directory provider is configured. If you have problems with this procedure, refer to the *Troubleshooting WebSphere Process Server* PDF.

Configuring the LDAP people directory provider

You configure the Lightweight Directory Access Protocol (LDAP) people directory provider so that Business Process Choreographer can perform people assignment, which determines who can start processes or claim activities or tasks.

Before you begin

To configure LDAP, you must have planning for it, as described in "Planning for the people directory provider" on page 156.

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 actual 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 another name, for example, `myLDAPTransformation.xsl`.

   - `Linux` `UNIX` On Linux and UNIX platforms the file is located in `install_root/ProcessChoreographer/Staff/LDAPTransformation.xsl`.

   - `Windows` On Windows platforms, the file is located in `install_root\ ProcessChoreographer\Staff\LDAPTransformation.xsl`.

2. Adapt the copy of the transformation file to suit the schema for your organization, as described in "Adapting the LDAP transformation file" on page 227.

> **CAUTION:**
> **Do not modify the original version of the transformation file because it can be overwritten without warning when you apply a service pack or fix pack.**

3. If Business Process Choreographer is configured on a cluster, place the copy of the transformation file in the `ProcessChoreographer/Staff` directory to make it 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 from the following list:

Option	Description
For a standalone profile	Only one node is displayed.
In a network deployment environment, where Business Process Choreographer is configured on one 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 (perform step 6) 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 (step 6) for all of the other nodes that host members of the cluster.

6. 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 the copy of the Extensible Stylesheet Language (XSL) transformation file that you adapted in step 2 on page 225. If the node agent is running, you can browse the file system of remote nodes to select the file.

   d. Click **Next** to copy the file to the `ProcessChoreographer\Staff` directory on the selected node.

   e. Enter an administrative name for the new people directory configuration, and optionally, a description

   f. Enter a unique Java Naming and Directory Interface (JNDI) name for human tasks to use to reference this provider. For example, `bpe/staff/ldapserver1`

   g. Click **Apply**, then click **Custom Properties**.

   h. For each of the required properties and for any optional properties that you planned in 2 on page 157, click the name of the property, enter a value, and click **OK**. For the optional additional properties, you can set properties that are defined for JNDI, for example to enable LDAP referrals, create an additional property named `java.naming.referral` with the value `follow`.

      For **providerURL**, you can specify a URL that starts with `ldap://` or `ldaps://`. If you have multiple LDAP servers that contain mirrored data for high availability, enter the URLs for the LDAP servers, using the space character to separate them.

   i. To apply the changes, click **Save**.

7. To activate the provider configuration, stop and start the server or servers where you configured the provider.

8. 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. This schema assumes the following:

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

If your LDAP schema uses name for object class and attribute names that are different from those listed above, you perform the following steps.

1. Make a copy of the standard transformation file for LDAP, and give it another name, for example, `myLDAPTransformation.xsl`.

   - **Linux** **UNIX** On Linux and UNIX platforms the file is located in `install_root`/ProcessChoreographer/Staff/LDAPTransformation.xsl.

   - **Windows** On Windows platforms, the file is located in `install_root\ProcessChoreographer\Staff\LDAPTransformation.xsl`.

2. In the copy of the file, change the names of the object classes and attributes to match the names used by your LDAP schema. For most situations, you can 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>
   ```

   CAUTION:
   Do not modify the original version of the standard transformation file because it might be overwritten without warning when you apply a service pack or fix pack.

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

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:resultObject>
<xsl:attribute name="objectclass">groupOfUniqueNames</xsl:attribute>
<xsl:attribute name="usage">recursive</xsl:attribute>
<sldap:resultAttribute>
<xsl:attribute name="name">uniqueMember</xsl:attribute>
<xsl:attribute name="destination">intermediate</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

<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">cn</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

</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">
(&
...
<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:resultObject>
<xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">myuid</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

<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: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">managerentry</xsl:attribute>
<xsl:attribute name="destination">intermediate</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>
</sldap:user>
</sldap:intermediateResult>

<sldap:user>
...
<xsl:attribute name="name">name</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">name</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
```

```
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

</sldap:user>
</sldap:StaffQueries>
```

Example: PersonSearch

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

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

<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:resultObject>
<xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">myuid</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>
</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:resultObject>
<xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">myuid</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
```

```
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

</sldap:user>
```

Configuring people substitution

This topic describes how to configure people substitution for Business Process Choreographer.

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 224. You know whether you will use a file registry, property extension registry, or an existing Lightweight Directory Access Protocol (LDAP) schema to store the property extensions.

About this task

To use people substitution in a production environment, you must use the Virtual Member Manager (VMM) property extension repository as described in this topic. If however, you only want to use people substitution in a single-server test environment, you can use the file registry that is associated by default with federated repositories, without having to configure VMM.

Procedure

1. Add the attributes, "isAbsent", "substitutes", "substitutionStartDate", and "substitutionEndDate" to the VMM definition for PersonAccount:

   a. Locate the `wimxmlextension.xml` file:

      - `Linux` `UNIX` On Linux and UNIX platforms, it is located in *profile_root*/config/cells/*cell_name*/wim/model

      - `Windows` 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>
<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
    dataType="STRING" multiValued="false" propertyName="substitutionStartDate">
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
</wim:propertySchema>

<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
```

```
        dataType="STRING" multiValued="false" propertyName="substitutionEndDate">
     <wim:applicableEntityTypeNames>PersonAccount
     </wim:applicableEntityTypeNames>
   </wim:propertySchema>
```

If you are using a file registry, `fileRegistry.xml`, skip to step 4 on page 234.

2. Set up the property extension repository. 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 class path. Click **Environment** → **WebSphere variables** to check. If necessary, add the JDBC driver to the class path. For DB2, add `db2jcc.jar`,`db2jcc_license_cu.jar` and `db2jcc_license_cisuz.jar` to the server's class path, 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.

      - Linux   UNIX   On Linux and UNIX platforms, it is located in *install_root*/etc/wim/setup

      - Windows   On Windows platforms, it is located in *install_root*\etc\wim\setup

   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>
<wimprop:property wimPropertyName="substitutionStartDate" dataType="String"
     valueLength="128" multiValued="false">
   <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
   </wimprop:applicableEntityName>
</wimprop:property>

<wimprop:property wimPropertyName="substitutionEndDate" dataType="String"
     valueLength="128" multiValued="false">
   <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 on a Windows platform:

```
$AdminTask setupIdMgrPropertyExtensionRepositoryTables {
-reportSqlError true
-schemaLocation install_root\etc\wim\setup
-laPropXML  install_root\etc\wim\setup\wimlaproperties.xml
-databaseType db2
-dbURL jdbc:DB2://host:port/name
-dbDriver com.ibm.db2.jcc.DB2Driver
-dbAdminId userID
-dbAdminPassword password }
```

Where *host* is the host name of the database server, *port* is the service port for the DB2 instance, and *name* is the name of the database.

i. If you are using a Lightweight Directory Access Protocol (LDAP) user repository, locate the `wimconfig.xml` file.

- **Linux** **UNIX** On Linux and UNIX platforms, the path is *profile_root*/config/cells/*cellName*/wim/config/wimconfig.xml

- **Windows** On Windows platforms, the path is *profile_root*\config\ cells*cellName*\wim\config\wimconfig.xml

Edit the file 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:propertiesNotSupported name="substitutionEndDate"/>
         <config:propertiesNotSupported name="substitutionStartDate"/>
      </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://host:port/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://host:port/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", "substitutes", "substitutionStartDate", and "substitutionEndDate" (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", "substitutes", "substitutionStartDate", and "substitutionEndDate", you must define your attribute names in the administrative console by clicking either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, on the **Configuration** tab select **Custom properties** then set the desired names for the custom properties `Substitution.SubstitutesAttribute`, `Substitution.AbsenceAttribute`, `Substitution.StartDateAttribute`, and `Substitution.EndDateAttribute`.

4. Restart the server.

5. Enable substitution in the Human Task Manager:

a. either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**, and click **Human Task Manager**, then either **Runtime** or **Configuration**.

b. To enable substitution, select **Enable substitution**.

c. If non-administrators are allowed to perform substitution for other users, clear the **Restrict substitute management to administrators** option.

   Note: This settings does not affect the ability of users to change substitution for themselves.

d. Click **Apply**.

e. If you selected **Configuration** in step 5a, restart the server to activate the substitution settings.

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.

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.
- You want to configure the optional Business Process Choreographer Explorer reporting function, which was previously available as the Business Process Choreographer Observer.

To configure Business Process Choreographer Explorer, perform one of the following:

Procedure

1. If you want to use a script, perform "Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer" on page 236.
2. 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.

Using the administrative console to configure the Business Process Choreographer Explorer

You can use the administrative console to configure Business Process Choreographer Explorer and the optional Business Process Choreographer Explorer reporting function.

Procedure

1. Click either **Servers → Clusters → WebSphere application server clusters → cluster_name** or **Servers → Server Types → WebSphere application servers → server_name**, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**.
2. To configure a new instance, click **Add**.
3. Enter values for the following fields:
   - If you want the new instance to start automatically at server start, select **Enable autostart**.
   - The **Context root** must be unique on the deployment target server or cluster.
   - **Explorer search result limit**.
   - **Managed Business Process Choreographer container**.
   - **Business Flow Manager REST API URL** , for standalone servers a default value is provided that points to the server's Web container.
   - **Human Task Manager REST API URL** , for standalone servers a default value is provided that points to the server's Web container.
4. Optional: If you want to configure the Business Process Choreographer Explorer reporting function perform the following.

a. Ensure that a Business Process Choreographer event collector is installed and configured.

b. Select **Enable reporting function**.

c. Select which Business Process Choreographer event collector will be visualized. If the list is empty, you must first install and configure a Business Process Choreographer event collector, as described in "Configuring the Business Process Choreographer Explorer reporting function and event collector" on page 240.

d. For **Report at snapshot range** specify how many days of data will be visualized.

5. Click **Apply**. Messages are displayed indicating the progress.

6. Optional: If any problems are reported, check the `SystemOut.log` file.

7. 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.

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. You can also use it to change configuration settings for an existing instance, including changing the maxListEntries and configuring the Business Process Choreographer Explorer reporting function.

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:

- Linux UNIX On Linux and UNIX platforms, it is in the directory `install_root/ProcessChoreographer/config`

- Windows On Windows platforms, it is in the directory `install_root\ProcessChoreographer\config`

Running the script in a stand-alone server environment

The configuration script is run using the wsadmin command. In a stand-alone server environment:

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

- If WebSphere administrative security is enabled, and your user ID does not have configurator or administrator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has configurator or administrator authority.

- If you are not working with the default profile, use the wsadmin -profileName
  profile option to specify the profile.

Running the script in a network deployment environment

The configuration script is run using the wsadmin command. 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, and your user ID does not have
  configurator or administrator authority, include the wsadmin -user and
  -password options to specify a user ID that has configurator or administrator
  authority.
- If you are not working with the default profile, use the wsadmin -profileName
  profile option to specify the profile.

Configuring the Business Process Choreographer Explorer non-interactively

Change your current directory to *install_root* and perform the following:

`Linux` `UNIX` On Linux and UNIX platforms, enter the command:

```
bin/wsadmin.sh -f ProcessChoreographer/config/clientconfig.jacl parameters
```

`Windows` On Windows platforms, enter the command:

```
bin\wsadmin.bat -f ProcessChoreographer/config/clientconfig.jacl parameters
```

Where *parameters* are:

```
  ( [-node nodeName][-server serverName] )
    [-cluster clusterName]
    [-contextRootExplorer explorerContextRoot]
    [-explorerHost explorerURL]
    [-hostName explorerVirtualHostname]
    [-precompileJSPs { yes | no }]
( ( [-remoteNode nodeName][-remoteServer serverName] )
      | [-remoteCluster clusterName] )
    [-maxListEntries maximum]
    [-reportAtSnapshotRange number]
    [-reportCreateTables { true | false }]
    -reportDataSource jndiName
    [-reportFunction { yes | no }]
    [-reportSchemaName schemaName]        -restAPIBFM restAPIURL
  -restAPIHTM restAPIURL
```

Note: If you run the script in batch mode, you must include all required
parameters. If you run the script interactively, any required parameters that are not
provided on the command line are prompted for.

Parameters

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

-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.

-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. In batch mode 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.

-hostName *VirtualHostname*

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

-maxListEntries *maximum*

Where *maximum* is the maximum number of results that the Business Process Choreographer Explorer returns for a query. The default is 10000.

-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.

-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.

-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.

-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.

-reportAtSnapshotRange *number*

A snapshot report is built by evaluating all events that are older than the qualifying snapshot date and time. This optional parameter defines the number of days for which events can be included in a snapshot report. Only events that have been emitted within this period are evaluated by the snapshot report. The default is 60 days. This optional parameter only has an effect if the reporting function is enabled using the option -reportFunction yes.

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.

-reportCreateTables { true | false }
This optional parameter indicates if the Business Process Choreographer Explorer reporting function schema is created when Business Process Choreographer Explorer connects to the database the first time. The default is true. This optional parameter only has an effect if the reporting function is enabled using the option `-reportFunction yes`.

-reportDataSource *jndiName*
Where *jndiName* is the JNDI name of the data source JNDI that is used to connect to the database. Mandatory parameter when `-reportFunction yes` is specified. The data source is not created automatically.

-reportFunction { yes | no }
This optional parameter controls whether the Business Process Choreographer Explorer reporting function is enabled. In interactive mode, the default is no. In batch mode, for compatibility with earlier versions, the default is yes.

-reportSchemaName *schemaName*
This optional parameter identifies the database schema that is used as a prefix for all reporting database objects. If you specify no schema name, a unique schema name is generated. This optional parameter only has an effect if the reporting function is enabled using the option `-reportFunction yes`. The default value is WPRBC00.

-restAPIBFM *restAPIURL*
Where *restAPIURL* is the URL for the Business Flow Manager REST API, which is needed to support the graphical process widget for the Business Process Choreographer Explorer. On a standalone server, the default is computed, for example, `http://localhost:9080/rest/bpm/bfm`. In a network deployment environment there is no default value.

-restAPIHTM *restAPIURL*
Where *restAPIURL* is the URL for the Human Task Manager REST API, which is needed to support the graphical process widget for the Business Process Choreographer Explorer. On a standalone server, the default is computed, for example, `http://localhost:9080/rest/bpm/htm`. In a network deployment environment there is no default value.

-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.

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:

* `Linux` `UNIX` On Linux and UNIX platforms: In the directory `profile_root`/logs

* `Windows` 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 Explorer reporting function and event collector

Using the Business Process Choreographer Explorer reporting function is optional, however, before you can use it, you must setup the database and install the applications.

Before you begin

You have performed "Planning for the Business Process Choreographer Explorer reporting function" on page 159, and "Configuring Business Process Choreographer Explorer" on page 234, but you did not configure the reporting function.

About this task

You want to configure the Business Process Choreographer Explorer reporting function, with its own database.

Procedure

1. If the database for the Business Process Choreographer does not already exist, perform "Preparing the reporting database."
2. Perform "Configuring the Business Process Choreographer event collector application" on page 278.
3. If you did not already enable the reporting function when you configured the Business Process Choreographer Explorer, perform "Enabling the Business Process Choreographer Explorer reporting function" on page 318.
4. Perform "Changing configuration parameters for the Business Process Choreographer Explorer reporting function" on page 284.
5. Perform "Enabling logging for Business Process Choreographer" on page 317.
6. Perform "Verifying the Business Process Choreographer Explorer reporting function" on page 291.

Results

The Business Process Choreographer Explorer reporting function is configured and working.

What to do next

You can use Business Process Choreographer Explorer reporting function to generate reports, as described in "Reporting on business processes and activities" on page 431.

Preparing the reporting database
Perform the actions for your database.

Using SQL scripts to create the reporting database:

You might choose to create the database for the Business Process Choreographer Explorer reporting function manually before you configure Business Process Choreographer, or even before you have installed the product.

Before you begin

You have performed "Planning the reporting database" on page 149.

About this task

Your organization might require that databases be created by a separate database administrator. If you use the administrative console or the bpeconfig.jacl script to configure Business Process Choreographer, customized SQL scripts are generated that you can give to your DBA to create the BPEDB database. However, if you want to create the database before configuring Business Process Choreographer, or even before product installation, your DBA must use the non-customized SQL scripts. This topic describes how to use the non-customized SQL scripts, which are available on the product media.

Procedure

On the server that hosts the database, create the database according to the description for your database system.

- "Using an SQL script to prepare a DB2 for iSeries database for the Business Process Choreographer Explorer reporting function" on page 247.
- "Using SQL scripts to prepare a DB2 Universal database for the Business Process Choreographer Explorer reporting function."
- "Using an SQL script to prepare a Derby database for the Business Process Choreographer Explorer reporting function" on page 260.
- "Using SQL scripts to prepare an Oracle database for the Business Process Choreographer Explorer reporting function" on page 264.

Results

The database for the Business Process Choreographer Explorer reporting function exists.

Preparing a DB2 Universal database for the Business Process Choreographer Explorer reporting function:

You can either use scripts or an interactive tool to prepare the reporting database.

Using SQL scripts to prepare a DB2 Universal database for the Business Process Choreographer Explorer reporting function:

This describes how to use scripts to prepare a DB2 Universal database on Linux, UNIX, and Windows platforms.

About this task

Your reporting database must already exist. You can either use an existing database, or use a new one. To perform this task, you must have administration rights for the destination database.

Procedure
1. If you configured Business Process Choreographer using the bpeconfig.jacl script in batch mode or using the administrative console, use the generated SQL script to create the database for the Business Process Choreographer Explorer reporting function.

a. Locate the generated `createSchema_Observer.sql` SQL script.

- If you configured Business Process Choreographer in a network deployment environment using the administrative console or by running the `bpeconfig.jacl` script in connected mode, the `createSchema_Observer.sql` script file is generated on the node of the deployment manager.

- If you configured Business Process Choreographer on a standalone server using the administrative console or by running the `bpeconfig.jacl` script in connected mode, the `createSchema_Observer.sql` script file is generated on the node where you invoked wsadmin.

- If you configured Business Process Choreographer by running the `bpeconfig.jacl` script in disconnected mode, the `createSchema_Observer.sql` script file is generated on the node of the standalone server.

- Linux   UNIX On Linux and UNIX platforms:
  - If you specified a schema qualifier, the generated script is:
    profile_root/dbscripts/ProcessChoreographer/DB2/*database_name*/*database_schema*/createSchema_Observer.sql.
  - If you did **not** specify a schema qualifier, the generated script is:
    profile_root/dbscripts/ProcessChoreographer/DB2/*database_name*/createSchema_Observer.sql.

- Windows For Windows platforms:
  - If you specified a schema qualifier, the generated script is:
    profile_root\dbscripts\ProcessChoreographer\DB2*database_name**database_schema*\createSchema_Observer.sql
  - If you did **not** specify a schema qualifier, the generated script is:
    profile_root\dbscripts\ProcessChoreographer\DB2*database_name*\createSchema_Observer.sql

Where:

database_name
    is the name of your database.

database_schema
    is the name of the schema, if you are using one.

b. If the database is remote, copy the generated script to the database host. If you are not authorized to perform this, give your database administrator a copy of the script and discuss your requirements with her.

c. Optional: You or your database administrator can customize the SQL script. For example, to specify the allocation of disks and table spaces that you planned in "Planning the reporting database" on page 149.

d. Run the SQL script on the database host by entering the following command:

```
db2 -tf createSchema_Observer.sql
```

2. If you configured Business Process Choreographer using the bpeconfig.jacl script in interactive mode, or if you have not configured Business Process Choreographer yet, there are no generated SQL scripts; you must customize a copy of the standard SQL script.

a. Change to the Business Process Choreographer subdirectory where the configuration scripts for your database are located.

- Linux   UNIX On Linux, and UNIX platforms:

- – Location on the product media: *media_root* or *extract_directory*/ `dbscripts/ProcessChoreographer/DB2`
      - – Location after installation:*install_root*`/dbscripts/ ProcessChoreographer/DB2`
    - • Windows On Windows platforms:
      - – Location on the product media:*media_root* or *extract_directory*\ `dbscripts\ProcessChoreographer\DB2`
      - – Location after installation:*install_root*`\dbscripts\ ProcessChoreographer\DB2`
  b. Copy all `*Observer.sql` script files to your database server.
  c. On the database server, change to the directory where you copied the script files.
  d. Create the table space:
    1) Edit the `createTablespace_Observer.sql` script file according to the instruction at the top of the file.
    2) Run the table space creation script file, enter the command:

       `db2 -tf createTablespace_Observer.sql`
    3) 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.
  e. Create the schema (tables, indexes, and views).
    1) Edit the `createSchema_Observer.sql` script file according to the instructions at the top of the file.
    2) In the DB2 command-line processor, enter the command:

       `db2 -tf createSchema_Observer.sql`
    3) Make sure that the script output contains no errors. If you want to drop the schema, use the `dropSchema_Observer.sql` script file.
3. If you want to use the Java implementation instead of the SQL implementation of the Business Process Choreographer Explorer task history UDFs perform "Selecting between Java and SQL user-defined functions" on page 269.
4. 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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Using the setupEventCollector tool to prepare a DB2 Universal database for the Business Process Choreographer Explorer reporting function:

This describes how use an interactive menu-driven tool and the `createTablespace_Observer.sql` script to prepare the reporting database .

Before you begin

Your database must already exist.

Procedure

1. If you use a type 2 JDBC connection:
   a. Prepare the command line environment:
      - `Linux` `UNIX` On Linux and UNIX platforms, run the `db2profile` for your DB2 instance.
      - `Windows` 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.
      - `Linux` `UNIX` On Linux and UNIX platforms, change to the directory *install_root*/dbscripts/ProcessChoreographer/DB2.
      - `Windows` 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. Connect to the database using a user ID that has SYSCTRL or SYSADM authority.
   e. Run the table space creation script file, enter the command:
      ```
      db2 -tf createTablespace_Observer.sql
      ```
   f. 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 Business Process Choreographer Explorer reporting function 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.

   Linux **UNIX** On Linux and UNIX platforms, enter:

   `cd install_root/ProcessChoreographer/config`

   Windows 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 290.

6. Prepare the database:

   a. When you see:

   ```
   1) Prepare a database for the Event Collector and reporting function
   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 reporting function
   6) Administer reporting function related user-defined functions

   0) Exit Menu
   ```

   Select option 1 to prepare a database for the event collector and Business Process Choreographer Explorer reporting function.

   b. When you see:

   ```
   Prepare a database for the Event Collector and reporting function

   Select the type of your database provider:

       c) Derby
       d) DB2 Universal
       i) DB2 iSeries
       8) DB2 V8/V9 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.
   - 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. If you use a different database, it must exist before you continue.

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 reporting function user-defined functions.

  Note: The Java UDFs are more precise, but they require a jar file
  installed to the database.
  Visit the reporting function 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.

You see something similar to the following:

```
Checking for required tablespace(es) ['OBSVRTS']
All required tablespaces were found.
Loading the jar file 'install_root\lib\bpcodbutil.jar' into the database.
The jar file 'install_root\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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Preparing a DB2 for iSeries database for the Business Process Choreographer Explorer reporting function:

You can either use scripts or an interactive tool to prepare the reporting database.

Using an SQL script to prepare a DB2 for iSeries database for the Business Process Choreographer Explorer reporting function:

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. You must have administrative authority (*ALLOBJ) for the database.

Procedure

1. If you configured Business Process Choreographer using the bpeconfig.jacl script in batch mode or using the administrative console, use the generated SQL script to create the database for the Business Process Choreographer Explorer reporting function.

   a. Locate the generated `createSchema_Observer.sql` SQL script.

      - If you configured Business Process Choreographer in a network deployment environment using the administrative console or by running the `bpeconfig.jacl` script in connected mode, the `createSchema_Observer.sql` script file is generated on the node of the deployment manager.

      - If you configured Business Process Choreographer on a standalone server using the administrative console or by running the `bpeconfig.jacl` script in connected mode, the `createSchema_Observer.sql` script file is generated on the node where you invoked wsadmin.

      - If you configured Business Process Choreographer by running the `bpeconfig.jacl` script in disconnected mode, the `createSchema_Observer.sql` script file is generated on the node of the standalone server.

      The generated script is: *profile_root*/dbscripts/ProcessChoreographer/ DB2iSeries/*collection_name*/createSchema_Observer.sql. Where:

      collection_name
          is the name of the collection.

   b. Copy the generated script to the database host. If you are not authorized to perform this, give your database administrator a copy of the script and discuss your requirements with her.

   c. Optional: You or your database administrator can customize the SQL script. For example, to specify the allocation of disks and table spaces that you planned in "Planning the reporting database" on page 149.

   d. Make sure that you are either in the DB2 command-line processor, or in a qshell.

   e. Run the SQL script on the database host by entering the following command:

      ```
      db2 -tf createSchema_Observer.sql
      ```

2. If you configured Business Process Choreographer using the bpeconfig.jacl script in interactive mode, or if you have not configured Business Process Choreographer yet, there are no generated SQL scripts; you must customize a copy of the standard SQL script.

   a. Locate the Business Process Choreographer subdirectory where the configuration scripts for your database are located.

      - Location on the product media: *media_root* or *extract_directory*/ dbscripts/ProcessChoreographer/DB2iSeries/createSchema.sql

      - Location after installation:*install_root*/dbscripts/ ProcessChoreographer/DB2iSeries/createSchema.sql

   b. Copy all `*Observer.sql` script files to your i5/OS database server.

   c. On the database server, change to the directory where you copied the script files.

   d. Create the schema (tables, indexes, and views).

      1) Edit the `createSchema_Observer.sql` script file according to the instruction at the top of the file.

2) Either in the DB2 command-line processor, or in a qshell, enter the command:

```
db2 -tf createSchema_Observer.sql
```

3) Make sure that the script output contains no errors. If you want to drop the schema, use the `dropSchema_Observer.sql` script file.

3. 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 269.

4. 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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Using the setupEventCollector tool to prepare a DB2 for iSeries database from a remote system:

This describes how use an interactive menu-driven tool to prepare a DB2 for iSeries database for the Business Process Choreographer Explorer reporting function 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 system. 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.

-    Linux   UNIX   On Linux and UNIX platforms, enter:

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

-    Windows   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 290.

5. Prepare the database:

   a. When you see:

```
1) Prepare a database for the Event Collector and reporting function
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 reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

      Select option 1 to prepare a database for the event collector and Business Process Choreographer Explorer reporting function.

   b. When you see:

```
Prepare a database for the Event Collector and reporting function

Select the type of your database provider:

    c) Derby
    d) DB2 Universal
    i) DB2 iSeries
    8) DB2 V8/V9 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 reporting function 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 Explorer reporting function has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Preparing a DB2 for z/OS database for the Business Process Choreographer Explorer reporting function:

You can prepare the reporting database remotely or within the UNIX System Services.

Creating a DB2 for z/OS database for the Business Process Choreographer Explorer reporting function in USS:

This describes how use an interactive menu-driven tool, and the `createTablespace_Observer.sql` script in UNIX System Services (USS) on a z/OS system, 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.

   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) Enable the DB2-supplied stored procedures and define the tables used by the DB2 Universal JDBC Driver, as described in the DB2 documentation.
>
> 2) Set up the environment for interpreted Java routines, as described in the DB2 documentation.
>
> 3) 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 Explorer reporting function database scripts for your database system are located. Depending on your DB2 version, enter one of the following commands:

      ```
      cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
      cd install_root/dbscripts/ProcessChoreographer/DB2zOSV9
      ```

   b. Edit the ASCII `createTablespace_Observer.sql` script. Replace `@STOGRP@` with the storage group name and replace `@DBNAME@` with the database name (not the subsystem name).

   c. Make sure that you are connected to your database, and run your customized version of the script.

      ```
      db2 -tf createTablespace_Observer.sql
      ```

4. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 290.

5. Prepare the database: When you see:

   ```
   1) Prepare a database for the Event Collector and reporting function
   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 reporting function
   6) Administer reporting function related user-defined functions

   0) Exit Menu
   ```

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

   b. When you see:

      ```
      c) Derby
      d) DB2 Universal
      i) DB2 iSeries8) DB2 V8/V9 on z/OSo) Oracle
      ```

      Enter 8 to select DB2 on z/OS.

   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.

Configuring Business Process Choreographer **253**

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 252.

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 252.

f. If you see:

```
Specify the hostname of the z/OS DB2 database server: [localhost]
```

Enter the host name 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 252.

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 252.

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.

k. If you see:

```
Note: The Java UDFs are more precise, but they require a jar file
installed to the database.
Visit the reporting function 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 252, 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 252.

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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Creating a DB2 for z/OS database for the Business Process Choreographer Explorer reporting function from a remote system:

This describes how use an interactive menu-driven tool, and the createTablespace_Observer.sql script on a Linux, UNIX, or Windows system to prepare the schema for the reporting 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.

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) Enable the DB2-supplied stored procedures and define the tables used by the DB2 Universal JDBC Driver, as described in the DB2 documentation.

2) Set up the environment for interpreted Java routines, as described in the DB2 documentation.

3) 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 system.

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 Explorer reporting function scripts for your database system are located:

   - Windows On Windows platforms, depending on your DB2 version, enter one of the following commands:

     ```
     cd install_root\dbscripts\ProcessChoreographer\DB2zOSV8
     cd install_root\dbscripts\ProcessChoreographer\DB2zOSV9
     ```

   - Linux UNIX On Linux and UNIX platforms, depending on your DB2 version, enter one of the following commands:

     ```
     cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
     cd install_root/dbscripts/ProcessChoreographer/DB2zOSV9
     ```

d. Edit the ASCII `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 script.

```
db2 -tf createTablespace_Observer.sql
```

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.

   Linux UNIX On Linux and UNIX platforms, enter:

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

   Windows 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 290.

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

i. When you see:

```
c) Derby
d) DB2 Universal
i) DB2 iSeries8) DB2 V8/V9 on z/OSo) Oracle
```

Enter 8 to select DB2 on z/OS.

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 256.

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 256.

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 reporting function 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 255.

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 256. 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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Preparing a Derby database for the Business Process Choreographer Explorer reporting function:

You can either use scripts or an interactive tool to prepare the reporting database.

Using an SQL script to prepare a Derby database for the Business Process Choreographer Explorer reporting function:

This describes how to use the `createSchema_Observer.sql` script to prepare the schema for the reporting database.

About this task

You must create the schema for the reporting database. You can either create it in an existing database, or have the script file create a new database for you.

Procedure
1. If you configured Business Process Choreographer using the bpeconfig.jacl script in batch mode or using the administrative console, use the generated SQL script to create the database for the Business Process Choreographer Explorer reporting function.
   a. Locate the generated SQL file.
      - `Linux` `UNIX` On Linux and UNIX platforms:
        - If you specified a schema qualifier, the generated script is located in the directory: *profile_root*/dbscripts/ProcessChoreographer/Derby/*database_name*/*database_schema*.
        - If you did **not** specify a schema qualifier, the generated script is located in the directory: *profile_root*/dbscripts/ProcessChoreographer/Derby/*database_name*.
      - `Windows` On Windows platforms:
        - If you specified a schema qualifier, the generated script is located in the directory: *profile_root*\dbscripts\ProcessChoreographer\Derby*database_name**database_schema*\
        - If you did **not** specify a schema qualifier, the generated script is located in the directory: *profile_root*\dbscripts\ProcessChoreographer\Derby*database_name*\

      Where:

      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.
   b. In a derby network server environment, copy the SQL script to the network server.
   c. Copy the JAR file `bpcodbutil.jar`, from the `lib` subdirectory of the *install_root* directory to the same directory on your database server.
   d. If you connect to an existing database using the embedded Derby driver, stop the server and any other applications that use the database.
   e. Run the script to create the schema. For example:
      - To create a database named OBSVRDB, from the directory where the database will be created, enter the command:

```
java -Dij.protocol=jdbc:derby:
    -Dij.database=OBSVRDB;create=true
    org.apache.derby.tools.ij
    createSchema_Observer.sql
```

- For a database named OBSVRDB, which already exists, from the directory where the database was created, enter the command:

```
java -Dij.protocol=jdbc:derby:
    -Dij.database=OBSVRDB
    org.apache.derby.tools.ij
    createSchema_Observer.sql
```

2. If you configured Business Process Choreographer using the bpeconfig.jacl script in interactive mode, or if you have not configured Business Process Choreographer yet, there are no generated SQL scripts; you must customize a copy of the standard SQL script.

   a. Change to the Business Process Choreographer subdirectory where the standard configuration scripts for your database are located.

      - **Linux** **UNIX** On Linux and UNIX platforms:
        - Location on the product media: *media_root* or *extract_directory*/`dbscripts/ProcessChoreographer/Derby`
        - Location after installation:*install_root*/`dbscripts/ProcessChoreographer/Derby`

      - **Windows** On Windows platforms:
        - Location on the product media:*media_root* or *extract_directory*\`dbscripts\ProcessChoreographer\Derby`
        - Location after installation:*install_root*\`dbscripts\ProcessChoreographer\Derby`

   b. In a derby network server environment, copy the `*Observer.sql` scripts to the network server.

   c. Copy the JAR file `bpcodbutil.jar`, from the `lib` subdirectory of the *install_root* directory to the same directory on your database server.

   d. 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.

   e. If you connect to an existing database using the embedded Derby driver, stop the server and any other applications that use the database.

   f. 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.

   g. 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 reporting database has been prepared.

Using the setupEventCollector tool to prepare a Derby database for the Business Process Choreographer Explorer reporting function:

This describes how use an interactive menu-driven tool, setupEventcollector, to prepare a Derby database for the reporting database on any supported platform.

Procedure

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

   Linux **UNIX** On Linux and UNIX platforms, enter:

   cd *install_root*/ProcessChoreographer/config

   Windows 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 290.

4. When you see:

   ```
   1) Prepare a database for the Event Collector and reporting function
   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 reporting function
   6) Administer reporting function 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 Event Collector and reporting function

   Select the type of your database provider:

       c) Derby
       d) DB2 Universal
       i) DB2 iSeries
       8) DB2 V8/V9 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 or embedded 40 JDBC driver
  2) Connect using the network or network 40 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 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 262.

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 262.

    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 reporting database has been prepared.

Preparing an Oracle database for the Business Process Choreographer Explorer reporting function:

You can either use scripts or an interactive tool to prepare the reporting database.

Using SQL scripts to prepare an Oracle database for the Business Process Choreographer Explorer reporting function:

This describes how to use scripts to prepare the reporting database.

About this task

Your reporting database must already exist. You can either use an existing database, or create a new one according to the database documentation.

Procedure

1. If you configured Business Process Choreographer using the bpeconfig.jacl script in batch mode or using the administrative console, use the generated SQL script to create the database for the Business Process Choreographer Explorer reporting function.

   a. Locate the generated SQL file.

      - ▨ Linux ▨ ▨ UNIX ▨ On Linux and UNIX platforms:
        - If you specified a schema qualifier, the generated script is: *profile_root*/dbscripts/ProcessChoreographer/Oracle/ *database_name*/*database_schema*/createSchema_Observer.sql.
        - If you did **not** specify a schema qualifier, the generated script is: *profile_root*/dbscripts/ProcessChoreographer/Oracle/ *database_name*/createSchema_Observer.sql.

      - ▨ Windows ▨ On Windows platforms:
        - If you specified a schema qualifier, the generated script is: *profile_root*\dbscripts\ProcessChoreographer\Oracle\ *database_name**database_schema*\createSchema_Observer.sql
        - If you did **not** specify a schema qualifier, the generated script is: *profile_root*\dbscripts\ProcessChoreographer\Oracle\ *database_name*\createSchema_Observer.sql

   b. Copy the generated script createSchema_Observer.sql to your database server.

   c. Run the createSchema_Observer.sql script file, by entering the following command:

   ```
   sqlplus userID/password
      @database_name@createSchema_Observer.sql
   ```

2. If you configured Business Process Choreographer using the bpeconfig.jacl script in interactive mode, or if you have not configured Business Process Choreographer yet, there are no generated SQL scripts; you must customize a copy of the standard SQL script.

   a. Change to the Business Process Choreographer subdirectory where the configuration scripts for your database are located.

      - ▨ Linux ▨ ▨ UNIX ▨ On Linux and UNIX platforms:
        - Location on the product media: *media_root* or *extract_directory*/ dbscripts/ProcessChoreographer/Oracle
        - Location after installation:*install_root*/dbscripts/ ProcessChoreographer/Oracle

      - ▨ Windows ▨ On Windows platforms:
        - Location on the product media:*media_root* or *extract_directory*\ dbscripts\ProcessChoreographer\Oracle
        - Location after installation:*install_root*\dbscripts\ ProcessChoreographer\Oracle

   b. Copy all *Observer.sql script files to your database server.

   c. Create the table space

      1) Edit your copy of the createTablespace_Observer.sql script file according to the instruction at the top of the file.
      2) Run your copy of the createTablespace_Observer.sql script file, by entering the following command:

```
sqlplus userID/password
   @database_name@createTablespace_Observer.sql
```

      3) 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.

  d. Create the schema (tables, indexes, and views).

      1) Edit your copy of the `createSchema_Observer.sql` script file according to the instruction at the top of the file.

      2) Run your copy of the `createSchema_Observer.sql` script file, by entering the following command:

```
sqlplus userID/password
   @database_name@createSchema_Observer.sql
```

      3) Make sure that the script output contains no errors. If you want to drop the schema, use the `dropSchema_Observer.sql` script file.

3. If you want to use the Java-based Business Process Choreographer user-defined functions:

  a. 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.

      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 system as the application server, change to the `lib` subdirectory of the *install_root* directory.

        • If your database is not on the same server 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 event collector database 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. 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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Using the setupEventCollector tool to prepare an Oracle database for the Business Process Choreographer Explorer reporting function:

This describes how use an interactive menu-driven tool, and the `createTablespace_Observer.sql` script to prepare the reporting 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. 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 standard configuration scripts for your database are located.

      - `Linux` `UNIX` On Linux and UNIX platforms, change to the directory *install_root*`/dbscripts/ProcessChoreographer/Oracle`.

      - `Windows` On Windows platforms, change to the directory *install_root*`\dbscripts\ProcessChoreographer\Oracle`.

   b. Edit a copy of 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.

   `Linux` `UNIX` On Linux and UNIX platforms, enter:

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

   `Windows` 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 290.
5. Prepare the database: When you see:

   ```
   1) Prepare a database for the Event Collector and reporting function
   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 reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

Perform the following:

a. Select option 1 to prepare a database for the event collector and Business Process Choreographer Explorer reporting function. The following menu is displayed:

```
Prepare a database for the Event Collector and reporting function

Select the type of your database provider:

     c) Derby
     d) DB2 Universal
     i) DB2 iSeries
     8) DB2 V8/V9 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 host name 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 reporting function user-defined functions.

  Note: The Java UDFs are more precise, but they require a jar file
  installed to the database.
  Visit the reporting function 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.

You see something similar to the following:

```
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 'install_root\lib\bpcodbutil.jar' into the database.
The jar file 'install_root\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 reporting database has been prepared.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Related tasks

"Selecting between Java and SQL user-defined functions"
Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Selecting between Java and SQL user-defined functions

Use the setupEventCollector tool or run scripts to switch between the Java-based and SQL-based user-defined functions (UDFs) in the reporting database.

Related concepts

"User-defined functions for Business Process Choreographer Explorer reporting function"
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

User-defined functions for Business Process Choreographer Explorer reporting function:

With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

UDFs can be installed in either of the following implementations:

The SQL implementation
Use the SQL implementation for UDFs that are implemented in plain SQL, using the built-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.

The 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 269.

Related tasks

"Selecting between Java and SQL user-defined functions" on page 269
Use the setupEventCollector tool or run scripts to switch between the Java-based
and SQL-based user-defined functions (UDFs) in the reporting database.

"Using an SQL script to prepare a DB2 for iSeries database for the Business Process
Choreographer Explorer reporting function" on page 247
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 from a
remote system" on page 249
This describes how use an interactive menu-driven tool to prepare a DB2 for
iSeries database for the Business Process Choreographer Explorer reporting
function from a remote Linux, Windows, or UNIX system.

"Creating a DB2 for z/OS database for the Business Process Choreographer
Explorer reporting function in USS" on page 252
This describes how use an interactive menu-driven tool, and the
`createTablespace_Observer.sql` script in UNIX System Services (USS) on a z/OS
system, to create a DB2 for z/OS database.

"Creating a DB2 for z/OS database for the Business Process Choreographer
Explorer reporting function from a remote system" on page 255
This describes how use an interactive menu-driven tool, and the
`createTablespace_Observer.sql` script on a Linux, UNIX, or Windows system to
prepare the schema for the reporting database.

"Using SQL scripts to prepare an Oracle database for the Business Process
Choreographer Explorer reporting function" on page 264
This describes how to use scripts to prepare the reporting database.

"Using the setupEventCollector tool to prepare an Oracle database for the Business
Process Choreographer Explorer reporting function" on page 267
This describes how use an interactive menu-driven tool, and the
`createTablespace_Observer.sql` script to prepare the reporting database.

"Using SQL scripts to prepare a DB2 Universal database for the Business Process
Choreographer Explorer reporting function" on page 241
This describes how to use 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 Explorer reporting function" on page 244
This describes how use an interactive menu-driven tool and the
`createTablespace_Observer.sql` script to prepare the reporting database .

Related reference

"setupEventCollector tool" on page 290
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. You
must configure an event collector if you want to use the Business Process
Choreographer Explorer reporting function.

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 Business Process Choreographer reporting
database.

Using scripts to select Java user-defined functions:

This describes how to use scripts to switch to the Java-based user-defined functions (UDFs) in the Business Process Choreographer reporting database.

About this task

You want to use the Java implementation of the UDFs instead of the SQL implementation. The Derby database can only use the Java implementation.

Procedure
1. Copy the jar file `bpcodbutil.jar`, to the same directory on your database server.
   - `Linux` `UNIX` On Linux and UNIX platforms: *install_root*/lib
   - `Windows` On Windows platforms: *install_root*\lib
2. Connect to the database.

Option	Description
DB2	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 `*`database_name`*
Oracle	Start the SQLPLUS command processor and connect to the database by entering the following command: `sqlplus `*`user@database_name`*

3. Install the jar file `bpcodbutil.jar`.

Option	Description

DB2	Enter the 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:
	
	• `Linux` `UNIX` 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')`
	
	• `Windows` 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')`

Option	Description

Oracle	1. Change to the directory where you copied the JAR file to.
	
	2. Enter the following command:
	
	`loadjava -user user_id/password`
	`        @localhost:port:database_name`
	`        -resolve bpcodbutil.jar -thin`
	`        -schema schema`
	
	Where *user_id* is a user ID with permission to install a JAR file, *password* is the password for the user ID, *port* is the port number of the SID listener, *database_name* is the name of the database, and *schema* is the name of the schema of the reporting tables, where the JAR file will be installed.

4. Edit the `dropFunctions_Observer.sql` script file according to the instruction at the top of the file. The script is located in the following directories:

   - **Linux** **UNIX** On Linux and UNIX platforms: *install_root*/`dbscripts/ProcessChoreographer/`*database_type*

   - **Windows** On Windows platforms: *install_root*\`dbscripts\`
`ProcessChoreographer\`*database_type*

5. Drop the SQL implementation of the UDFs.

Option	Description
DB2	In a DB2 command-line processor, enter the command: `db2 -tf dropFunctions_Observer.sql`
Oracle	In the SQLPLUS command processor, enter the command: `@ path/dropFunctions_Observer.sql` Where *path* is the fully qualified path.

6. Edit the `createFunctionsJava_Observer.sql` script file according to the instruction at the top of the file. The script is located in the following directories:

   - **Linux** **UNIX** On Linux and UNIX platforms: *install_root*/`dbscripts/ProcessChoreographer/`*database_type*

   - **Windows** On Windows platforms: *install_root*\`dbscripts\`
`ProcessChoreographer\`*database_type*

7. Run the script to create the Java implementation of the UDFs.

Option	Description
DB2	In a DB2 command-line processor, enter the command: `db2 -tf createFunctionsJava_Observer.sql`

Option	Description
Oracle	In the SQLPLUS command processor, enter the command: `@ path/createFunctionsJava_Observer.sql` Where *path* is the fully qualified path.

Results

The UDF implementation used has been switched to use Java.

Using scripts to select SQL user-defined functions:

This describes how to use scripts to switch to the SQL-based user-defined functions (UDFs) in the Business Process Choreographer reporting database.

About this task

You want to use the SQL implementation of the UDFs instead of the Java implementation. The Derby database can only use the Java implementation.

Procedure

1. Edit the `dropFunctions_Observer.sql` script file according to the instruction at the top of the file. The script is located in the following directories:

   - Linux   UNIX   On Linux and UNIX platforms: *install_root*/ `dbscripts/ProcessChoreographer/`*database_type*

   - Windows   On Windows platforms: *install_root*\dbscripts\ `ProcessChoreographer\`*database_type*

2. Connect to the database.

Option	Description
DB2	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 `*database_name*
Oracle	Start the SQLPLUS command processor and connect to the database by entering the following command: `sqlplus user@`*database_name*

3. Run the script to drop the Java implementation of the UDFs.

Option	Description
DB2	In a DB2 command-line processor, enter the command: `db2 -tf dropFunctions_Observer.sql`
Oracle	In the SQLPLUS command processor, enter the command: `@ path/dropFunctions_Observer.sql` Where *path* is the fully qualified path.

4. Edit the `createFunctionsSql_Observer.sql` script file according to the instruction at the top of the file.
5. Run the script to create the SQL implementation of the UDFs.

Option	Description
DB2	`db2 -tf createFunctionsSql_Observer.sql`
Oracle	`@ path/createFunctionsSql_Observer.sql` Where *path* is the fully qualified path.

6. Optional: Remove the JAR file from the database. Enter the command for your database.

Option	Description

DB2	`db2 call sqlj.remove_jar(`
	`    'schema.BPCODBUTIL')`
	
	Where *schema* is the name of the schema under which the JAR file was installed.

Oracle	`dropjava -user user_id/password`
	`          @localhost:port:database_name`
	`          -resolve bpcodbutil.jar -thin`
	`          -schema schema`
	
	Where *user_id* is a user ID with permission to install a JAR file, *password* is the password for the user ID, *port* is the port number of the SID listener, *database_name* is the name of the database, and *schema* is the name of the schema under which the JAR file was installed.

Results

The UDF implementation used has been switched to use SQL.

Using the setupEventCollector tool to select between Java and SQL user-defined functions:

This describes how use an interactive menu-driven tool to switch between the Java-based and SQL-based user-defined functions (UDFs) in the Business Process Choreographer reporting database.

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 290. You see the following menu:

```
1) Prepare a database for the Event Collector and reporting function
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 reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

2. Select option 6 to administer the user-defined functions for the Business Process Choreographer Explorer reporting function. You will see the following menu:

```
c) Derby
d) DB2 Universal
i) DB2 iSeries
8) DB2 V8/V9 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, 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 DB2 for iSeries, select option i.

   a. Enter the following connection information:

   Database name

   Database server host name
           Only prompted for if you are running the tool remotely.

   Directory of your JDBC driver

   Database user ID

   Password

5. 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

6. If a connection can be established to the database, you will see the menu for administering the UDFs for the database:

```
6) Administer reporting function 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 reporting function applications. Continue anyway?
y) yes
n) no
Your selection:
```

If an important report is running, either enter n to not continue the switch or wait until it has completed. Enter y 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 reporting function 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 reporting function functions
     into the database
  2) Remove the jar file containing the reporting function 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 275.

Results

The reporting database will use the UDFs that you selected.

Configuring the Business Process Choreographer event collector application

The Business Process Choreographer event collector is a prerequisite for using the Business Process Choreographer Explorer reporting function. You can 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 use an interactive menu-driven tool to install and configure the event collector application on a server or cluster.

Procedure

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

   - `Linux` `UNIX` On Linux and UNIX platforms, enter the command:

     cd *install_root*/ProcessChoreographer/config

   - `Windows` On Windows platforms, enter the command:

     cd *install_root*\ProcessChoreographer\config

2. Start the tool to set up the event collector, as described in "setupEventCollector tool" on page 290. For example, to start the tool to work with a server named server1 enter one of the following commands:

   `Linux` `UNIX` **On Linux and UNIX platforms:**
   ```
   setupEventCollector.sh -server server1
   ```

   `Windows` **On Windows platforms:**
   ```
   setupEventCollector.bat -server server1
   ```

   You see the Commands Menu:
   ```
   Commands Menu

     1) Prepare a database for the Event Collector and reporting function
     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 reporting function
     6) Administer reporting function 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 in a network deployment environment, you will see something like:

Configuring Business Process Choreographer   **279**

```
          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 Explorer reporting function.

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 reporting database.

Procedure
1. In the administrative console, navigate to the Business Process Choreographer event collector configuration page: Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, 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 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**
         - **Existing event group name**
         - **Event group name**
   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** → **Application Types** → **WebSphere 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.

Enabling the Business Process Choreographer Explorer reporting function after migration

Existing Business Process Choreographer Observer and Explorer configurations are migrated and can still be used, but the new Business Process Choreographer Explorer reporting function, which can replace the Business Process Choreographer Observer is disabled.

Before you begin

You have migrated from a previous release, your Business Process Choreographer Explorer configurations are migrated, and any existing Business Process Choreographer Observer configurations are migrated.

About this task

Your old Business Process Choreographer Observer is not modified, it remains at the code level that you migrated from. The old URL (the default is *host***:***port*/bpcobserver) will continue to work, until you manually remove the Business Process Choreographer Observer application, and switch to using the new Business Process Choreographer Explorer reporting function, which you can do whenever it is convenient.

Procedure

1. If the Business Process Choreographer Observer and Business Process Choreographer Explorer were configured in the migration source release, then a template JACL script is generated during migration, which you must edit and run to enable the Business Process Choreographer Explorer reporting function.

   a. Locate the script file:
      - On a stand-alone server:
        - [Linux] [UNIX] On Linux and UNIX platforms, it is generated in *profile_root*/ProcessChoreographer/migrate_BPCObserver_*scope*.jacl.
        - [Windows] On Windows platforms, it is generated in *profile_root*\ProcessChoreographer\migrate_BPCObserver_*scope*.jacl.
      - In a network deployment environment, it is generated in the deployment manager profile. Do not run the script before all the profiles where Business Process Choreographer Explorer instances run have been migrated.

      The script file name has the postfix *scope*, which has the value of *nodeName_serverName* or *clusterName*.

   b. Edit the generated script file according to the instructions in the script file.

   c. To enable the reporting function on the selected Business Process Choreographer Explorer instance, run the customized script according to the instructions in the script file.

d. In a network deployment environment, make sure that there is a data
      source pointing to the reporting database, and that it is visible in the scope
      of your Business Process Choreographer Explorer reporting function. If your
      previous Business Process Choreographer Observer and Business Process
      Choreographer Explorer configurations were on different deployment
      targets or if you want to enable the reporting function on multiple instances
      of Business Process Choreographer Explorer that are installed in different
      clusters, then you might need to create a new data source.
2. Make sure that the server or servers for the Business Process Choreographer
   Explorer are running, and that the Business Process Choreographer Explorer
   application is started.
3. Enable existing clients. Either notify all users to use the new URL (default:
   host:*port*/bpc) instead of the old URL (default: *host*:*port*/bpcobserver), or
   configure an automatic redirection on your Web server.
4. Test that clients can access the Business Process Choreographer Explorer
   reporting function and that it is working correctly.
5. To uninstall the old Business Process Choreographer Observer enterprise
   application, perform the following:
   a. In the administrative console, click **Applications** → **Application Types** →
      WebSphere enterprise applications.
   b. Locate the Business Process Choreographer Observer instances. Their 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*.
   c. To uninstall the Business Process Choreographer Observer application, select
      the application instance that you want to delete, then click **Uninstall** → **OK** →
      Save.

Results

The Business Process Choreographer Explorer reporting function is enabled,
existing users can access it, and the old Business Process Choreographer Observer
application has been removed.

Using the administrative console to configure the Business Process Choreographer Explorer reporting function

This describes how to use the administrative console to configure an instance of
the Business Process Choreographer Explorer reporting function to connect to the
data source for a particular event collector.

Before you begin

You have configured the Business Process Choreographer event collector and the
Business Process Choreographer Explorer, but you did not select the option to
configure the Business Process Choreographer Explorer reporting function.

Procedure

1. In the administrative console, navigate to the Business Process Choreographer
   Explorer configuration page: Click either **Servers** → **Clusters** → **WebSphere**

application server clusters → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**

2. Select the Business Process Choreographer Explorer instance that you want to enable the reporting function for.

3. If the option for **Enable reporting function** is disabled, then it is already configured.

4. If the option for the Business Process Choreographer Explorer reporting function is enabled, you can configure it by performing the following.

   a. Ensure that a Business Process Choreographer event collector is installed and configured.

   b. Select **Enable reporting function**.

   c. Select which Business Process Choreographer event collector will be visualized. If the list is empty, you must first install and configure a Business Process Choreographer event collector, as described in "Configuring the Business Process Choreographer Explorer reporting function and event collector" on page 240.

   d. For **Report at snapshot range** specify how many days of data will be visualized.

5. Click **Apply**. Messages are displayed indicating the progress.

6. Optional: If any problems are reported, check the SystemOut.log file.

Results

The Business Process Choreographer Explorer reporting function is configured and ready to use.

What to do next

You can configure more instances of the Business Process Choreographer Explorer reporting function, on the same or different deployment targets, however, each instance must connect to a different event collector data source.

Changing configuration parameters for the Business Process Choreographer Explorer reporting function

Tuning the configuration parameters for the Business Process Choreographer Explorer reporting function 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 Explorer reporting function.

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. You might need to change this parameter if did not specify a schema when you created the setup, or if you change the runtime user ID or the database provider.

Configuration parameter	Data type / Units	Default value	Description
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 Business Process Choreographer Explorer reporting function. When 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 Business Process Choreographer Explorer reporting function. 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 Business Process Choreographer Explorer reporting function. 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 Business Process Choreographer Explorer reporting function.
ObserverCreateTables	Boolean		This parameter indicates if the Business Process Choreographer Explorer reporting function schema should be created when the EJB connects to the database the first time. Valid values are 'true' and 'false'. You might want to enable or disable this, for example, if you want to reuse an existing setup, but want to use a new datasource.

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 Business Process Choreographer Explorer reporting function.

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 reporting 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 290. You see the following menu:

```
1) Prepare a database for the Event Collector and reporting function
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 reporting function
6) Administer reporting function 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 Business Process Choreographer Explorer reporting function

The value for the ReportAtSnapshotRange parameter can have a big impact on the performance of snapshot reports.

Configuration parameter and clientconfig.jacl parameter	Data type / Units	Default value	Description
ObserverSchemaName `-reportSchemaName` *schemaName*	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. You might need to change this parameter if did not specify a schema when you created the setup, or if you change the runtime user ID or the database provider. This must match the value for the event collector.
ReportAtSnapshotRange `-reportAtSnapshotRange` *number*	Integer / Days	60	A snapshot report is built by evaluating all events that are older than the qualifying snapshot date and time. This 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. 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
 	
`-reportCreateTables { true`
`| false }`
	Boolean	For DB2 on z/OS: `false`.
		
		For all other database types: `true`.
			This parameter indicates if the Business Process Choreographer Explorer reporting function schema is created when the EJB connects to the database the first time. Valid values are 'true' and 'false'.

Changing configuration parameters for the Business Process Choreographer Explorer reporting function

To change Business Process Choreographer Explorer reporting function parameters, you can run the clientconfig.jacl script using one of the following parameters: -reportSchemaName *schemaName,* -reportAtSnapshotRange *number,* and -reportCreateTables { true | false }. For more information about these parameters, see "Using the clientconfig.jacl script file to configure the Business Process Choreographer Explorer" on page 236.

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. You must configure an event collector if you want to use the Business Process Choreographer Explorer reporting function.

Location

This tool is located in the Business Process Choreographer subdirectory for configuration scripts:

 Linux   UNIX On Linux and UNIX platforms: *install_root*/ ProcessChoreographer/config.

 Windows 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.

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 administrative 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.

`Linux` `UNIX` **On Linux and UNIX platforms:**

```
setupEventCollector.sh -server server1
```

`Windows` **On Windows platforms:**

```
setupEventCollector.bat -server server1
```

You will see the Commands menu:

```
1) Prepare a database for the Event Collector and reporting function
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 reporting function
6) Administer reporting function 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 Explorer reporting function" on page 270
With Business Process Choreographer Explorer reporting function you can run reports based on timeslices or time intervals that result in SQL queries. To perform these reports, Business Process Choreographer Explorer reporting function requires some specific user-defined functions (UDFs) to be installed in the reporting database.

Verifying the Business Process Choreographer Explorer reporting function

After configuring the Business Process Choreographer Explorer reporting function, verify that it works correctly.

Before you begin

Initially, the Business Process Choreographer Explorer reporting function database is empty.

Procedure

1. Generate some business events.
   a. In a browser, start the Business Process Choreographer Explorer 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 `bpc`.
   b. Perform some actions that will generate business events, for example, start a process instance.
2. Click **Reports**. 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 Explorer reporting function" on page 284.
3. Verify that the events you expect to be available are displayed.
4. In case of problems, see "Troubleshooting Business Process Choreographer Explorer reports" on page 772.

Results

The Business Process Choreographer Explorer reporting function is working.

Configuring a remote client application

Configuring a remote Business Process Choreographer client application that runs on a WebSphere Process Server client installation.

Before you begin

You have performed "Planning for a remote client application" on page 162, and know whether you are creating the "single-cell" scenario or the "cross-cell" scenario.

Procedure

1. For the "single-cell" scenario, where the WebSphere Process Server client installation is in the same cell as the Business Process Choreographer server or cluster that the client connects to, perform the following:
   a. Install and configure the WebSphere Process Server client:
      1) Install the WebSphere Process Server client without installing the server: Make sure that the **WebSphere Process Server - Client** option is selected and the **WebSphere Process Server** option is not selected.

Note: If you want to use the WebSphere Process Server client in a cluster, then you must install the WebSphere Process Server client on all WebSphere Application Server installations that host cluster members.

2) If the profile already exists, make sure that it is augmented with Feature Pack for SCA Version 1.0 with SDO 2.1.1.

3) If the profiles do not yet exist, perform the following:

   a) Either start the profile management tool and select **Custom profile with Feature Pack for SCA Version 1.0 with SDO 2.1.1**, or if you are using the manageprofiles command line tool, use the following profile template:

-       **Linux**   **UNIX**   On Linux and UNIX platforms: *install_root*/profileTemplates/SCA/managed.sdo

-       **Windows**   On Windows platforms: *install_root*\profileTemplates\SCA\managed.sdo profile template.

   b) Federate the profile into the WebSphere Process Server cell. You can also perform this action later, using the addNode command.

   c) Using the administrative console, create an application server using the WebSphere "default" server template on the WebSphere Process Server client node.

b. Optional: Configure the Business Process Choreographer Explorer on the application server in the WebSphere Process Server client using either the administrative console or the clientconfig.jacl script. For the Business Process Choreographer container target, make sure that you select the WebSphere Process Server server or cluster that hosts the Business Flow Manager and Human Task Manager.

c. Optional: Install and configure a custom client application.

1) Install the custom client application on the application server in the WebSphere Process Server client installation.

2) Edit the EJB bindings for the custom client applications:

   a) Using the administrative console, click **Applications** › **Application Types** › **WebSphere enterprise applications**.

   b) Click on your custom client application.

   c) Under **References**, select **EJB references**. You will see the resource references specified by your client application.

   d) Locate the references to the Business Process Choreographer API EJBs. You will see the default resource reference names and the following JNDI names for the target resources:

```
ejb/BusinessFlowManagerHome   com/ibm/bpe/api/BusinessFlowManagerHome
ejb/HumanTaskManagerHome      com/ibm/task/api/HumanTaskManagerHome
```

   e) Change the target resource JNDI names to values where the Business Flow Manager and Human Task Manager API are located in your cell:

- If Business Process Choreographer is configured on another server in the same cell, the setting has the following structure:

```
cell/nodes/nodename/servers/servername/com/ibm/bpe/api/BusinessFlowManagerHome
cell/nodes/nodename/servers/servername/com/ibm/task/api/HumanTaskManagerHome
```

- If Business Process Choreographer is configured on a cluster in the same cell, the setting looks like:

```
cell/clusters/clustername/com/ibm/bpe/api/BusinessFlowManagerHome
cell/clusters/clustername/com/ibm/task/api/HumanTaskManagerHome
```

3) Save and synchronize your changes.

4) Restart your client application.

d. The default configuration of the Remote Artifact Loader (RAL) allows the unsecured transmission of artifacts between the client and the server. To enable security for this connection, refer to Remote artifact loader.

2. If you want a " cross-cell" scenario, where the WebSphere Process Server client is not located in the cell that has the managed server or cluster with Business Process Choreographer configured on it, you can install the WebSphere Process Server client on any WebSphere Application Server installation that hosts standalone profiles or managed profiles for a different network deployment cell. As a minimum, this network deployment cell only requires a WebSphere Application Server deployment manager. To set up your WebSphere Process Server client installation in this kind of environment and configure it to access the cell with the Business Process Choreographer configuration, perform the following:

a. Install and configure the WebSphere Process Server client:

1) Install the WebSphere Process Server client without installing the server: Make sure that the **WebSphere Process Server - Client** option is selected and the **WebSphere Process Server** option is not selected.

   Note: If you want to use the WebSphere Process Server client in a cluster, then you must install the WebSphere Process Server client on all WebSphere Application Server installations that host cluster members.

2) If the profile already exists, make sure that it is augmented with Feature Pack for SCA Version 1.0 with SDO 2.1.1.

3) If the profiles do not yet exist, and you are using a **standalone** profile, either start the profile management tool and select **Application server with Feature Pack for SCA Version 1.0 with SDO 2.1.1**, or if you are using the manageprofiles command line tool, use the following profile template:

   - `Linux` `UNIX` On Linux and UNIX platforms: *install_root*/profileTemplates/SCA/default.sdo

   - `Windows` On Windows platforms: *install_root*\profileTemplates\ SCA\default.sdo

   For example, to augment a standalone profile named "wp_profile" on a Linux or UNIX platform, using the manageprofiles tool, enter the following:

   ```
   install_root/bin/manageprofiles.sh -augment -templatePath
   install_root/profileTemplates/SCA/default.sdo -profileName wp_profile
   ```

   After several minutes, when the augmentation has succeeded, you should see the following message:

   ```
   INSTCONFSUCCESS: Profile augmentation succeeded.
   ```

4) If the profiles do not yet exist, and you are using a **managed** profile, perform the following:

   a) Either start the profile management tool and select **Custom profile with Feature Pack for SCA Version 1.0 with SDO 2.1.1**, or if you are using the manageprofiles command line tool, use the following profile template:

      - `Linux` `UNIX` On Linux and UNIX platforms: *install_root*/profileTemplates/SCA/managed.sdo

      - `Windows` On Windows platforms: *install_root*\ profileTemplates\SCA\managed.sdo

b) Federate the profile into the WebSphere Process Server cell. You can also perform this action later, using the addNode command.

c) Using the administrative console, create an application server using the WebSphere "default" server template on the WebSphere Process Server client node.

b. Optional: Install and configure a custom client application.

1) Make sure the custom client application uses the Business Process Choreographer EJB APIs.

2) Install the custom client application on the application server or cluster in the WebSphere Process Server client installation.

c. Define a new indirect namespace binding (or bindings) to connect to the cluster or server where Business Process Choreographer is configured:

1) Using the administrative console on the client cell, click **Environment** → **Naming** → **Name Space Bindings**.

2) For **Scope**, select the cell.

3) Depending on whether the client application uses one or both of the Business Flow Manager EJB API and the Human Task Manager EJB API, perform the following steps once or twice to create a new binding for one or both of the EJB APIs:

a) Click **New**.

b) For the **Binding Type**, select **Indirect**. In the next screen, specify the following properties:

i. A unique binding identifier name. Although you are free to choose a unique name, for consistency with the service component architecture (SCA), you can derive a valid name from the namespace by replacing the forward slashes in the namespace with underscore characters. For example, the namespace

`bpc/`*`remoteCellName_remoteNode_remoteServer`*`/com/ibm/bpe/api/BusinessFlowManagerHome`

becomes the binding ID name

`bpc_`*`remoteCellName_remoteNode_remoteServer`*`_com_ibm_bpe_api_BusinessFlowManagerHome`

ii. The namespace of the client to be used for the binding. For consistency, consider using the following convention:

- If the remote Business Process Choreographer configuration is on a server: `bpc/`*`remoteCellName_remoteNode_remoteServer`*`/com/ibm/bpe/api/BusinessFlowManagerHome` or `bpc/`*`remoteCellName_remoteNode_remoteServer`*`/com/ibm/task/api/HumanTaskManagerHome`

- If the remote Business Process Choreographer configuration is on a cluster: `bpc/`*`remoteCellName_remoteCluster`*`/com/ibm/bpe/api/BusinessFlowManagerHome` or `bpc/`*`remoteCellName_remoteCluster`*`/com/ibm/task/api/HumanTaskManagerHome`

iii. The provider URL property for the naming server that is used by the server or cluster, where the Business Process Choreographer configuration exists that the client will connect to. For example, `corbaloc:iiop:myremotehostname:2809`. Make sure that the bootstrap port matches the BOOTSTRAP_ADDRESS of the server (or one of the members of the cluster) where Business Process Choreographer is hosted.

c) Specify the target resource JNDI name where the Business Flow Manager API or Human Task Manager API is located.

Configuring Business Process Choreographer   **295**

- If Business Process Choreographer is configured on a standalone server, the setting has the following structure:

  ```
  com/ibm/bpe/api/BusinessFlowManagerHome
  com/ibm/task/api/HumanTaskManagerHome
  ```

- If Business Process Choreographer is configured on in a network deployment environment, the setting has the following structure:

  ```
  cell/nodes/nodename/servers/servername/com/ibm/bpe/api/BusinessFlowManagerHome
  cell/nodes/nodename/servers/servername/com/ibm/task/api/HumanTaskManagerHome
  ```

- If Business Process Choreographer is configured on a cluster, the setting looks like:

  ```
  cell/clusters/clustername/com/ibm/bpe/api/BusinessFlowFlowManagerHome
  cell/clusters/clustername/com/ibm/task/api/HumanTaskManagerHome
  ```

4) Using the administrative console on the client system:

   a) Click **Applications** → **Enterprise Applications** → *client_application_name*, then on the name of the client application, then in the **References** section, select **EJB references**.

   b) There will be one **Target Resource JNDI Name** field for each namespace that you defined. Enter the JNDI name or names that you specified in step 2c3bii on page 295 for the Business Flow Manager, Human Task Manager, or both.

   c) Save and synchronize your changes.

   d) Restart your client application.

Results

You have configured a remote Business Process Choreographer client application that uses a WebSphere Process Server client installation.

Related concepts

"Comparison of the programming interfaces for interacting with business processes and human tasks" on page 461
Enterprise JavaBeans (EJB), Web service, and Java Message Service (JMS), and Representational State Transfer Services (REST) generic programming interfaces are available for building client applications that interact with business processes and human tasks. Each of these interfaces has different characteristics.

Related tasks

"Configuring Business Process Choreographer Explorer" on page 234
You can either run a script or use the administrative console to configure Business Process Choreographer Explorer.

"Developing client applications for business processes and tasks" on page 461
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.

"Accessing the remote interface of the session bean" on page 538
An EJB client application for business processes or human tasks accesses the remote interface of the session bean through the remote home interface of the bean.

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, click **Applications** → **Application Types** → **WebSphere 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.

   Restriction: In a network deployment environment, you can only install one instance of the Business Process Choreographer installation verification application. For example, if you have two Business Process Choreographer clusters in the same network deployment cell, you can install the `bpcivt.ear` application only on one of the clusters. Later, if you want to install it on the second cluster, you must first uninstall it from the first cluster.
   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 Explorer, Business Process Choreographer Explorer reporting function, 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 information center. 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.

Removing the Business Process Choreographer configuration

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

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. Uninstall all enterprise applications that contain human tasks or business processes.
3. Perform one of the following actions:
   - To remove the Business Process Choreographer configuration, Business Process Choreographer Explorer, 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 304.

Results

The Business Process Choreographer configuration has been removed.

Using a script to remove the Business Process Choreographer configuration

Remove the Business Flow Manager, Human Task Manager, Business Process Choreographer Explorer, and the associated resources from a server or cluster.

Before you begin

Before you can remove the Business Process Choreographer configuration, the following conditions must be met:

- You must delete any business process and human task instances, then uninstall all enterprise applications that contain business processes or human tasks.
- If WebSphere administrative security is enabled, and your user ID does not have operator or administrator authority, include the wsadmin -user and -password options to specify a user ID that has operator or administrator authority.
- If you are not working with the default profile, use the wsadmin -profileName *profile* option to specify the profile.
- For stand-alone servers, stop the application server and use the -conntype NONE option. This step ensures that any 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.

Procedure

1. Change to the Business Process Choreographer `config` directory:

   `Linux` `UNIX` On Linux and UNIX platforms, enter the command:

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

   `Windows` On Windows platforms, enter the command:

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

2. Run the script `bpeunconfig.jacl`.

   - `Linux` `UNIX` For a single server on a Linux or UNIX platform, enter
     the command:

     ```
     install_root/bin/wsadmin.sh
         -f bpeunconfig.jacl
         -server Server -node Node
         [-deleteDB deleteDatabase]
         [-forcePredefTasks yes|no]
     ```

   - `Linux` `UNIX` For a cluster on a Linux or UNIX platform, enter the
     command:

     ```
     install_root/bin/wsadmin.sh
         -f bpeunconfig.jacl
         -cluster Cluster
         [-forcePredefTasks yes|no]
     ```

   - `Windows` For a single server on a Windows platform, enter the command:

     ```
     install_root\bin\wsadmin.bat
         -f bpeunconfig.jacl
         -server Server -node Node
         [-deleteDB deleteDatabase]
         [-forcePredefTasks yes|no]
     ```

   - `Windows` For a cluster on the Windows platform, enter the command:

     ```
     install_root\bin\wsadmin.bat
         -f bpeunconfig.jacl
         -cluster Cluster
         [-forcePredefTasks yes|no]
     ```

   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.

forcePredefTasks yes|no

This specifies whether to force the removal of the predefined Human Tasks enterprise application:

yes

no

If you select **yes**, the predefined Human Tasks application is removed from the WebSphere configuration repository, but the corresponding entries remain in the Business Process Choreographer database.

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 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 reporting database, start the tool to set up the event collector, as described in "setupEventCollector tool" on page 290, and select the option **Drop the database schema of the Event Collector and reporting function**.

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:

   - Installing the BusinessCalendar system application
   - 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 a tool to remove the Business Process Choreographer event collector

Remove the Business Process Choreographer event collector application, and the
associated resources from a server or cluster.

Procedure

1. To remove the Business Process Choreographer event collector application, run
   the "setupEventCollector tool" on page 290, either specify the -remove option
   on the command line, or select the **Remove the Event Collector application
   and related objects** option on the initial menu.
2. Optional: If you installed the Java user-defined functions, remove them using
   the menu option **Administer reporting function related user-defined
   functions**.
3. Optional: Drop the database schema for the event collector.

Results

The Business Process Choreographer event collector application and its related
objects 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, and the
associated resources.

Before you begin

Before you can remove the Business Process Choreographer configuration, you
must 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** → **Application Types** →
      WebSphere 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.
      - HTM_PredefinedTasks_V*nnn_scope* and
        HTM_PredefinedTaskMsg_V*nnn_scope* are for the Business Process
        Choreographer Business Space.

Where *nnn* is the version number, and 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 predefined tasks, Business Flow Manager, and Human Task Manager applications.

1) Select `HTM_PredefinedTasks_Vnnn_scope` and `HTM_PredefinedTaskMsg_Vnnn_scope`, then click **Uninstall** → **OK** → **Save**.

2) Select `BPEContainer_scope`, and `TaskContainer_scope`, then click **Uninstall** → **OK** → **Save**.

d. Optional: If you configured the Business Process Choreographer Explorer, uninstall all instances that you configured.

- 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**.

e. If you configured any Business Process Choreographer event collectors, perform "Using the administrative console to remove the Business Process Choreographer event collector" on page 309 for each event collector application instance.

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 the associated authentication data alias (if any) and Java Naming and Directory Interface (JNDI) name, and then remove 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 Embedded 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 connection factories and queues for the Business Flow Manager, and Human Task Manager. Here are the normal JNDI names:

Connection factories for the Business Flow Manager:
        `jms/BPECF`
        `jms/BPECFC`
        `jms/BFMJMSReplyCF`

Queues for the Business Flow Manager:
        `jms/BPEIntQueue`

```
jms/BPERetQueue
jms/BPEHldQueue
jms/BFMJMSAPIQueue
jms/BFMJMSCallbackQueue
jms/BFMJMSReplyQueue
```

Connection factory for the Human Task Manager:
```
jms/HTMCF
```

Queues for the Human Task Manager:
```
jms/HTMIntQueue
jms/HTMHldQueue
```

How you delete the connection factories and queues depends on the JMS messaging provider that you use.

- 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**.

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** ⮞ **Server Types** ⮞ **WebSphere application servers** ⮞ *server_name*.

   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** → **Global security** then in the **Authentication** section, expand **Java Authentication and Authorization Service** , click **JCA authentication data**.

   2) If the data source identified in step 2 on page 305 had an authentication data alias, remove that alias. If you did not migrate your Business Process Choreographer configuration from Version 6.0.2.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 305 have an authentication data alias, remove the alias with great care:

      • If you did **not** create your Business Process Choreographer configuration on 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 created your Business Process Choreographer configuration on 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** → **Server Types** → **WebSphere application servers** → *server_name*, then in the **Container Settings** section, expand **Business Process Services**, and click **Work area partition service**.

   2) On the Application servers pane, select and delete the work area partition `BPECompensation`.

Removing the Business Process Choreographer configuration   **307**

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 304

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**, then in 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. Click **Service integration** ▸ **Buses** ▸ **BPC.***cellName***.Bus**, then in 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. Click **Save** to save all your deletions in the master configuration.
6. Restart the application server or cluster.
7. Optional: Delete the Business Process Choreographer database.
8. Optional: If you used the Business Process Choreographer Explorer reporting function with a dedicated reporting database, delete the database.
9. Optional: If you are using WebSphere MQ, delete the queue manager used by Business Process Choreographer.
10. 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 308 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 308 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 308. Usually, the Derby database location is the following:

       - `Linux`   `UNIX`   On Linux and UNIX platforms:

         `profile_root`/databases/com.ibm.ws.sib/
              `nodeName`.`serverName`-BPC.`cellName`.Bus

       - `Windows`   On Windows platforms:

         `profile_root`\databases\com.ibm.ws.sib\
              `nodeName`.`serverName`-BPC.`cellName`.Bus

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 Explorer reporting function.

Procedure

1. Display the enterprise applications.

   In the administrative console, select **Applications** ⇒ **Application Types** ⇒ **WebSphere 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** ⇒ **CEI.***cellName***.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 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**.

5. 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**.

6. 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**.

7. 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**.

8. If your configuration was created on version 6.0.2, delete the authentication data alias:
   a. Click **Security** → **Global security** then in the **Authentication** section, expand **Java Authentication and Authorization Service** , click **JCA authentication data**.
   b. select BPCEventCollectorJMSAuthenticationAlias_*scope*.
   c. Click **Delete**.

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

10. Drop the schema and table space used by the Business Process Choreographer Explorer reporting function by running the following scripts:
    - dropSchema_Observer.sql
    - dropTablespace_Observer.sql (not available for Derby, which has no table spaces)
    - `Linux` `UNIX` On Linux and UNIX platforms they are located in *install_root*/dbscripts/ProcessChoreographer/*database_type*
    - `Windows` On Windows platforms they are located in *install_root*\dbscripts\ProcessChoreographer*database_type*

Results

The Business Process Choreographer event collector configuration has been
removed.

Part 3. Administering

Administering Business Process Choreographer

You can administer Business Process Choreographer using the administrative console or using scripts.

Cleanup procedures for Business Process Choreographer

An overview of the runtime objects that can be deleted from the database after they are no longer needed, and the tools available.

Types of tools available for deleting objects

Depending on which types of objects you want to delete, you will be able to use one or more of the following tools:

- The cleanup service.
- The administrative console.
- Administrative scripts.
- The modelling tool.
- Failed Event Manager
- Business Process Choreographer Explorer
- Business Process Choreographer APIs

Objects that can be deleted and tools to use

The following Business Process Choreographer database objects can be deleted when they are no longer needed.

API-accessible objects
> You can write your own cleanup tool that uses the Business Process Choreographer APIs to delete process instances, task instances, and task templates that were created at runtime using the APIs. Templates that are part of an enterprise application cannot be deleted using the APIs. For general information about using the APIs, refer to "Developing client applications for business processes and tasks" on page 461.

Process and task templates
> Templates can be deleted in the following ways:
> - Uninstall the applications:
>   - "Uninstalling business process and human task applications, using the administrative console" on page 650.
>   - "Uninstalling business process and human task applications, using an administrative command" on page 651.
> - Run a script to delete templates:
>   - "Deleting process templates that are no longer valid" on page 348.
>   - "Deleting human task templates that are no longer valid" on page 350.

Process and task instances
> Instances can be deleted in the following ways:

- Using the administrative console to configure the cleanup service to schedule jobs that periodically delete eligible instances. This is described in "Configuring the cleanup service and cleanup jobs" on page 325.
- Run a script to delete completed instances:
  - "Deleting completed process instances" on page 339.
  - "Deleting completed task instances" on page 342.
- Set the appropriate properties in the business model, using WebSphere Integration Developer:

  For business processes:
  > The property Automatically delete the process after completion can have the value Yes, No, or On successful completion. If this property has the value No or On successful completion, it makes sense to configure a cleanup job to delete the process instances.

  For human tasks:
  > The property Auto deletion mode can have the value On completion, or On successful completion (which is the default). Deletion will only take place, and you can only change the value for Auto deletion mode, if the property Duration until task is deleted either has the value Immediate or a defined interval. If the property Duration until task is deleted has the value Never, automatic deletion is disabled, the Auto deletion mode property cannot be changed, and it makes sense to configure a cleanup job to delete the human tasks. Otherwise, if Duration until task is deleted does not have the value Never, and Auto deletion mode has the value On successful completion, then it makes sense to define a cleanup job to delete the human tasks that do not complete successfully.

- Uninstall the template and using the **-force** option to also delete all instances. This option is described in "Uninstalling business process and human task applications, using an administrative command" on page 651.
- To delete a small number of instances, it can be convenient to use the Business Process Choreographer Explorer, which allows you to check details about the instances before you delete them.

Note: You can use more than one of the above techniques for deleting instances. In which case, an instance will be deleted by the first attempt to delete it.

Audit log entries
> You can delete audit log entries by running the deleteAuditLog.py script, which is described in "Deleting audit log entries, using administrative scripts" on page 337.

Reporting events
> You can delete reporting events by running the observerDeleteProcessInstanceData.py script, which is described in "Deleting data from the reporting database" on page 345.

People queries
> You can delete unused people queries by running the cleanupUnusedStaffQueryInstances.py script, which is describe in "Removing unused people query results, using administrative scripts" on page 352.

Hold queue

Messages that cannot be processed are placed on the hold queue, this includes messages for instances that have been deleted. You can empty the hold queue by replaying the messages in the queue, which will cause any messages for deleted instances to be discarded.

- "Querying and replaying failed messages, using the administrative console" on page 321 describes how to replay messages using the Business Process Choreographer pages, and using the failed event manager page.
- "Querying and replaying failed messages, using administrative scripts" on page 333

Related tasks

"Configuring the cleanup service and cleanup jobs" on page 325
Use the administrative console to configure and schedule cleanup jobs that periodically delete instances of business processes and human tasks that are in particular states.

"Deleting Business Process Choreographer objects" on page 337
Various database objects accumulate in a running system, for example, audit log entries, task and process instances, task and process templates, reporting data, and people queries. Regularly running administrative scripts to delete objects that are no longer needed from the Business Process Choreographer databases can help prevent wasting storage space.

"Querying and replaying failed messages, using the administrative console" on page 321
This describes how to check for and replay any messages for business processes or human tasks that could not be processed.

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 Explorer reporting function, 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 Explorer reporting function, 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 Business Process Choreographer" on page 329.

To enable CEI logging for Business Process Choreographer, using the administrative console, perform "Enabling Common Base Events, the audit trail, and the task history using the administrative console" on page 319.

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 the administrative console to administer Business Process Choreographer

Describes the administrative actions that can be performed using the administrative console.

Related tasks

"Using scripts to administer Business Process Choreographer" on page 329
Describes the administrative actions that can be performed using scripts.

Enabling the Business Process Choreographer Explorer reporting function

This describes how to use the administrative console to enable the Business Process Choreographer Explorer reporting function to connect to the data source for a particular event collector.

Before you begin

You have configured the Business Process Choreographer event collector and the Business Process Choreographer Explorer, but you did not select the option to configure the Business Process Choreographer Explorer reporting function.

About this task

This task uses the administrative console, but you can also use the clientconfig.jacl script file to enable the reporting function.

Procedure

1. In the administrative console, navigate to the Business Process Choreographer Explorer configuration page: Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**.
2. A list of configured Business Process Choreographer Explorer instances is displayed. Select which one you want to enable the reporting function on. If the option for the Business Process Choreographer Explorer reporting function is already selected, then it is already configured.
3. If the option for the Business Process Choreographer Explorer reporting function is not selected, you can configure it by performing the following.

a. Ensure that a Business Process Choreographer event collector is installed and configured.

b. Select **Enable reporting function**.

c. Select which Business Process Choreographer event collector will be visualized. If the list is empty, you must first install and configure a Business Process Choreographer event collector, as described in "Configuring the Business Process Choreographer Explorer reporting function and event collector" on page 240.

d. For **Report at snapshot range** specify how many days of data will be visualized.

4. Click **Apply**. Messages are displayed indicating the progress.

5. Optional: If any problems are reported, check the `SystemOut.log` file.

Results

The Business Process Choreographer Explorer reporting function is configured and ready to use.

Enabling Common Base Events, the audit trail, and the task history 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. You can also use this task to exploit task history data using either Business Space or the Task Instance History Representational State Transfer (REST) interface.

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 task history, 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 either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**.

b. 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:

   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.

   Enable task history
   This option is only available for the Human Task Manager. Select this check box to display task history data in Business Space, or to retrieve task history data using the Task Instance History Representational State Transfer (REST) interface.

3. Accept the change.
   a. Click **OK**.
   b. In the Messages box, click **Save**.

4. To enable WebSphere Business Monitor to monitor Service Component Architecture (SCA) events, you must set a custom property.
   a. In the administrative console, click **Servers → Clusters → WebSphere application server clusters → *cluster_name*** or **Servers → Server Types → WebSphere application servers → *server_name***, then under **Business Integration**, expand **Business Process Choreographer**, and click **Business Flow Manager → Custom Properties**.
   b. Click **New** to add a new custom property.
   c. Enter the name `Compat.SCAMonitoringForBFMAPI` and the value `true`.
   d. Save the changes. The setting will be activated the next time that you restart the server.

Results

The state observers are set, as you required.

What to do next

Restart the server, to activate 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 task history, audit log, or common event infrastructure logging for the session.

About this task

Choices made on the **Runtime** tab are effective immediately.

Procedure

1. Display the Business Flow Manager or Human Task Manager pane.
   a. Click either **Servers → Clusters → WebSphere application server clusters → *cluster_name*** or **Servers → Server Types → WebSphere application servers →**

server_name, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**.

   b. 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:

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.

Enable task history
This option is only available for the Human Task Manager. Select this check box to display task history data in Business Space, or to retrieve task history data using the Task Instance History Representational State Transfer (REST) interface.

3. To enable WebSphere Business Monitor to monitor Service Component Architecture (SCA) events, you must set a custom property.

   a. In the administrative console, click **Servers → Clusters → WebSphere application server clusters → *cluster_name*** or **Servers → Server Types → WebSphere application servers → *server_name***, then under **Business Integration**, expand **Business Process Choreographer**, and click **Business Flow Manager → Custom Properties**.

   b. Click **New** to add a new custom property.

   c. Enter the name `Compat.SCAMonitoringForBFMAPI` and the value `true`.

4. If you want any changes made on the **Runtime** tab to remain in effect after the next server restart, select **Save runtime changes to configuration**.

5. Click **OK** to accept the change.

Results

The state observers are set, as you required.

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. For the Business Flow Manager, the most flexible way to check and reply and messages on the hold queue, is to use the administrative console page for the failed event manager.

a. Click **Integration Applications** → **Failed Event Manager** → **Search failed events**, for **Event type**, select **BFM hold** then click **OK**.

b. If the search results contains any messages, you can select any of them then either click **Resubmit** to replay the messages, or **Delete** to delete them from the hold queue without replaying them.

2. To check how many messages are in the hold and retention queues, and replay them using the Business Process Choreographer administrative console pages:

a. Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**.

b. 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**.

c. 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 administrator or operator authority.

Results

Business Process Choreographer tries to service all replayed messages again.

Related concepts

"Cleanup procedures for Business Process Choreographer" on page 315
An overview of the runtime objects that can be deleted from the database after they are no longer needed, and the tools available.

Related tasks

"Querying and replaying failed messages, using the administrative console" on page 321
This describes how to check for and replay any messages for business processes or human tasks that could not be processed.

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 Business Process Choreographer administration page for the application server or cluster.

   Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**.

2. Refresh the message counts.
   a. 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.

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 either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.

2. 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 administrator or operator authority.
Refreshing the people query results in this way can cause a high load on the

application and database. Consider using an administrative script instead.

Results

Related tasks

"Refreshing people query results, using administrative scripts" on page 335
The results of a people query are static. Use the administrative scripts to refresh people queries.

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 either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.
   b. Choose one of the following options:
      • To change settings without having to restart the server, select the **Runtime** tab.
      • To make changes that will only have an effect after the server is restarted, select the **Configuration** tab.
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. If you want any changes made on the **Runtime** tab to remain in effect after the next server restart, select **Save runtime changes to configuration**.
5. Click **OK**.
6. Save the changes. To make your changes that you made 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 tasks

"Refreshing people query results, using administrative scripts" on page 335
The results of a people query are static. Use the administrative scripts to refresh
people queries.

Configuring the cleanup service and cleanup jobs

Use the administrative console to configure and schedule cleanup jobs that
periodically delete instances of business processes and human tasks that are in
particular states.

Before you begin

Identify times of the day and days of the week, when it would be best to schedule
the cleanup service, for example, when there is the lowest load on the database.
For each business process and human task that you want the cleanup service to
delete, decide which states make an instance a candidate for deletion, and decide
how long an instance must be in one of those states before the next scheduled
cleanup deletes them.

About this task

You want completed instances to be deleted automatically after keeping them for a
while. There is a separate cleanup service for the Business Flow Manager and for
the Human Task Manager. For each of them, you must first enable the service and
define the service parameters, such as the schedule, maximum duration of the
cleanup, and the database transaction size. Then you can define cleanup jobs for
sets of templates and define the end states and the duration that an instance must
be in to qualify for deletion. The Human Task Manager cleanup service only
deletes stand-alone human tasks, but when the Business Flow Manager cleanup
service deletes a business process, it also deletes all of the child processes and
inline human tasks that are contained in the process. When security is enabled, the
cleanup user ID specified for the Business Process Choreographer configuration
must be in the business administrator role.

Procedure

1. Configure the cleanup service for the **Business Flow Manager**.

   a. To configure the cleanup service in a cluster, in the administrative console,
   click **Servers** → **Clusters** → **WebSphere application server clusters** →
   cluster_name, then on the **Configuration** tab, in the **Business Integration**
   section, expand **Business Process Choreographer**, and click **Business Flow
   Manager**.

   b. To configure the cleanup service on a stand-alone server, in the
   administrative console, click **Servers** → **Server Types** → **WebSphere
   application servers** → *server_name*, then on the **Configuration** tab, in the
   Business Integration section, expand **Business Process Choreographer**, and
   click **Business Flow Manager**.

   c. Choose one of the following options:

   - To change settings without having to restart the server, select the
     Runtime tab.

   - To make changes that will only have an effect after the server is restarted,
     select the **Configuration** tab.

   d. In the **Additional properties** section, click **Cleanup Service Settings**.

e. If the cleanup service is not enabled, select **Enable cleanup service**. For a cluster configuration, the cleanup service will be scheduled to run on one of the cluster members of the cluster it is configured on.

f. For **Frequency**, specify the time and frequency when the Business Flow Manager cleanup service will run. Enter a WebSphere crontab format string, which defines the start of a low load time slot. For example, to run the cleanup service every night at eleven o'clock, use the default value of 0 0 23 * * ? .

g. For **Maximum duration**, enter the maximum time that the cleanup is allowed to run. The default is 120 minutes. Make sure that the maximum duration is shorter than the time interval specified by the frequency.

h. For **Transaction slice**, enter the number of business process instances that will be deleted in each database transaction. The default value is 10. Because the value affects the performance of the cleanup service, it is worth trying different values. Depending on the size of the human tasks being deleted, you might be able to increase the slice size to increase the performance. However, if you get transaction timeouts, you should reduce the value.

i. Save your changes.

2. Add a new cleanup job for the **Business Flow Manager**.

   a. In the administrative console, on the **Business Flow Manager** page, click **Cleanup Service Jobs**.

   b. To create a new cleanup job, click **Add**.

   c. If this is not the only cleanup job, for **Order Number**, you can select a sequence number that determines the order that the jobs will be run, starting with number zero.

   d. For **Cleanup Job**, enter a name for the job.

   e. For **Templates**, either enter the name of one or more business process templates (one per line) whose instances (including any inline human tasks) will be deleted, or enter an asterisk ('*') to specify all business process templates.

   f. For **Restrict cleanup to instances in the following states**, select one or more of the following states:

      • **FINISHED**

      • **TERMINATED**

      • **FAILED**

   g. For **Duration until deletion**, specify how long an instance must be in one of the specified states before it becomes eligible for deletion by the cleanup job. Enter integers in the following fields: **Minutes**, **Hours**, **Days**, **Months**, and **Years**. The default is two hours.

   h. Click **Apply** or **OK**.

   i. Save your changes.

   j. If necessary, repeat this step to define more cleanup jobs for business process instances.

3. Configure the cleanup service for the **Human Task Manager**.

   a. To configure the cleanup service in a cluster, in the administrative console, click **Servers → Clusters → WebSphere application server clusters → cluster_name**, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.

b. To configure the cleanup service on a stand-alone server, in the administrative console, click **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.

c. If the cleanup service is not enabled, select **Enable cleanup service**. For a cluster configuration, the cleanup service will be scheduled to run on one of the cluster members of the cluster it is configured on.

d. For **Frequency**, specify the time and frequency when the Human Task Manager cleanup service will run. Enter a WebSphere crontab format string, which defines a low load time slot.

   Tip: If the cleanup service for the Business Flow Manager is also enabled, specify a schedule that does not overlap with the time window defined by the values specified in steps 1f on page 326 and 1g on page 326. For example, if the Business Flow Manager cleanup service starts every night at one o'clock, and can run for up to two hours, you can specify that the cleanup service for the Human Task Manager runs every night at three o'clock by entering the value 0 0 3 * * ?.

e. For **Maximum duration**, enter the maximum time that the cleanup is allowed to run. The default is 120 minutes. Make sure that the maximum duration is shorter than the time interval specified by the frequency.

f. For **Transaction slice**, enter the number of human task instances that will be deleted in each database transaction. The default value is 10. Because the value affects the performance of the cleanup service, it is worth trying different values. Depending on the size of the human tasks being deleted, you might be able to increase the slice size to increase the performance. However, if you get transaction timeouts, you should reduce the value.

g. Save your changes.

4. Add a new cleanup job for the **Human Task Manager**.

a. In the administrative console, on the **Human Task Manager** page, click **Cleanup jobs**.

b. To create a new cleanup job, click **Add**.

c. If this is not the only cleanup job, for **Order Number**, you can select a sequence number that determines the order that the jobs will be run, starting with number zero.

d. For **Cleanup Job**, enter a name for the job.

e. For **Templates**, either enter the name of one or more stand-alone human task templates (one per line) whose instances will be deleted, or enter an asterisk ('*') to specify all standalone human task templates. To specify a namespace for a task template, append it in brackets, for example, myTaskTemplate (http://bpc/samples/task/).

   Note: The Human Task Manager cleanup service can also delete inline invocation tasks that are started using the Human Task Manager API.

f. For **Restrict cleanup to instances in the following states**, select one or more of the following states:
   - **FINISHED**
   - **TERMINATED**
   - **FAILED**
   - **INACTIVE**
   - **EXPIRED**

g. For **Duration until deletion**, specify how long an instance must be in one of the specified states before it becomes eligible for deletion by the cleanup job. Enter integers in the following fields: **Minutes**, **Hours**, **Days**, **Months**, and **Years**. The default is two hours.

h. Click **Apply** or **OK**.

i. Save your changes.

j. If necessary, repeat this step to define more cleanup jobs for standalone human task instances.

5. If you made the changes on the **Configuration** tab, restart the server to activate the changes.

Results

You have activated the cleanup services and defined cleanup jobs to delete completed instances. When the cleanup service starts and finishes, the messages CWWBF0118I and CWWBF0119I are written to the SystemOut.log file. When one cleanup job starts and finishes, the messages CWWBF0116I and CWWBF0117I are written to the SystemOut.log file. Progress updates of the cleanup processing are written with message CWWBF0120I to the SystemOut.log.

Related concepts

"Cleanup procedures for Business Process Choreographer" on page 315
An overview of the runtime objects that can be deleted from the database after they are no longer needed, and the tools available.

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. In a cluster, you must perform this server-level setup consistently for each cluster member. 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.

Note: In a high availability (HA) environment, each server in a cluster must have a unique compensation log and transaction log directory, so that multiple servers do not attempt to access the same log file. Also, each server in the cluster must be able to access the transaction and compensation log directories of the other servers in the cluster. To change the directory in which compensation logs are written, type the full path name of the directory in the **Recovery log directory** field.

You can use the administrative console to view and change properties of the compensation service for application servers.

Procedure

1. In the administrative console, click **Servers** ⟶ **Server Types** ⟶ **WebSphere application servers** ⟶ *server_name*.

2. On the **Configuration** tab, under **Container Settings**, click **Container Services** → **Compensation service**. This action displays a panel with the compensation service properties.

3. 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.

Using scripts to administer Business Process Choreographer

Describes the administrative actions that can be performed using scripts.

About this task

When using administrative scripts that trigger long-running work on the server, a script can fail if the connection timeout is not long enough to complete the action. If the wsadmin scripting client terminates because of a connection timeout, check the `SystemOut.log` file on the server to see whether you need to restart the script. If it happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the `soap.client.props` file. Some scripts have parameters that you can specify, which influence how much work is performed.

There is no cross-cell support for the Business Process Choreographer administrative scripts. This means that you can connect the scripting client only to a server or the deployment manager of the cell to which the node of the profile where the script runs belongs.

Related tasks

"Using the administrative console to administer Business Process Choreographer" on page 318
Describes the administrative actions that can be performed using the administrative console.

Using a script to enable logging for Business Process Choreographer

This describes how to use the `setStateObserver.py` script to enable or disable Common Event Infrastructure (CEI), audit events for Business Process Choreographer, or the task history logging for the Human Task Manager.

Location

The `setStateObserver.py` script is located in the Business Process Choreographer `config` directory.

- ▌Linux▐ ▌UNIX▐ On Linux and UNIX platforms, it is in the directory *install_root*/ProcessChoreographer/config

- ▌Windows▐ On Windows platforms, it is in the directory *install_root*\ProcessChoreographer\config

Running the script

To run the `setStateObserver` script:

 Linux UNIX On Linux and UNIX platforms, enter:

```
install_root/bin/wsadmin.sh
      -f install_root/ProcessChoreographer/config/setStateObserver.py
```

 Windows On Windows platforms, enter:

```
install_root\bin\wsadmin.bat
      -f install_root\ProcessChoreographer\config\setStateObserver.py
```

Parameters

The script file can take the following parameters:

-bfm
> Optionally specifies that the enabling or disabling is to apply to the Business Process Choreographer's Business Flow Manager, which runs business processes.

-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.

-conntype *NONE*
> Only include this option if the application server (for stand-alone) or deployment manager is not running.

-enable { CEI | AuditLog | TaskHistory}
> Optionally specifies whether to enable CEI logging, audit logging, or the Human Task Manager task history. To specify more than one, use a semi-colon as a separator, for example, to enable CEI and audit logging, use `-enable CEI;AuditLog`. The value `TaskHistory` is not valid if `-bfm` is specified.

-disable { CEI | AuditLog | TaskHistory}
> Optionally specifies whether to disable CEI logging, audit logging, or the Human Task Manager task history. To specify more than one, use a semi-colon as a separator, for example, to enable CEI and audit logging, use `-enable CEI;AuditLog`. The value `TaskHistory` is not valid if `-bfm` is specified.

-htm
> Optionally specifies that the enabling or disabling is to apply to the Business Process Choreographer's Human Task Manager, which runs human tasks.

-node *nodeName*
> Where *nodeName* is the name of the node. Do not specify this option if you specify a cluster.

-profileName *profileName*
> Where *profileName* is the name of the profile to use.

-server *serverName*
> Where *serverName* is the name of the server. Do not specify this option if you specify a cluster.

Example

To enable CEI logging for business process events on `server1`:

- `Linux` `UNIX` On Linux and UNIX platforms:

  `wsadmin.sh -f setStateObserver.py -server server1 -enable CEI -bfm`

- `Windows` On Windows platforms:

  `wsadmin.bat -f setStateObserver.py -server server1 -enable CEI -bfm`

How time zones are handled in Business Process Choreographer

When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Depending on the client that you are using, times are displayed in your browser in the local time of the client or the server.

For administrative scripts, time parameters end with the postfix `Local` or `UTC`, which indicates whether the times are interpreted as being in the scripting client's local time or in Coordinated Universal Time (UTC). By using the `Local` version of the time parameters you can avoid having to perform any calculations to adjust for time zones and daylight saving time.

Table 28. Time zone use in Business Process Choreographer interfaces

Client or interface	Time zone used or displayed
Administrative console	Server local time zone
Business Process Choreographer Explorer	Client local time zone
Business Space	Client local time zone
Administrative scripts	UTC or the scripting client's local time
APIs	UTC

For example, the deleteCompletedProcessInstances script can be given time stamp values for the -validFromUTC, -completedAfterLocal, -completedAfterUTC, -completedBeforeLocal, and -completedBeforeUTC parameters. The parameter name suffixes show whether the time must be specified in UTC or in the scripting client's local time.

For time zones where daylight saving time is observed, the local times displayed are adjusted for daylight saving time if the date and time being displayed falls in the period when daylight saving is observed.

The administrative script parameter -validFromUTC is used to distinguish between different template versions and must always be specified exactly to the second. For other script parameters that take a time, like -completedAfterLocal, -completedAfterUTC, -completedBeforeLocal, and -completedBeforeUTC, if you specify a date with no time, it defaults to `00:00:00`.

Migrating process instances to a new process template version by running a script

After deploying a new version of a process template, new process instances are based on the new version, but existing instances based on the old template version

continue running until they reach an end state. You can use the migrateProcessInstances.py script to migrate running instances.

Before you begin

The following conditions must be met:

- The application server where the templates are deployed must be running.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have administrator authority, include the wsadmin -user and -password options to specify a user ID that has administrator authority.

About this task

Use the migrateProcessInstances.py script to migrate instances of a specific process template version to the latest version, or a specified version. Instances that are in an end state (finished, terminated, or failed) are not migrated. Only instances of the specified template with the same version as the specified "valid from" value are migrated. If you prefer to write a script to migrate instances, an MBean interface is available.

Procedure

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

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   `Windows` On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

2. Migrate the instances of process templates that are no longer valid. Enter the command:

   `Linux` `UNIX` On Linux and UNIX platforms, enter:

   ```
   install_root/bin/wsadmin -f migrateProcessInstances.py
           [([-node node_name] -server server_name ) | ( -cluster cluster_name )]
           ( -templateName template_name)
           (-sourceValidFromUTC timestamp )
           [(-targetValidFromUTC timestamp )]
           [(-slice slice_size
   ```

   `Windows` On Windows platforms, enter:

   ```
   install_root\bin\wsadmin -f migrateProcessInstances.py
           [( [-node node_name] -server server_name ) | ( -cluster cluster_name )]
           ( -templateName template_name)
           (-sourceValidFromUTC timestamp )
           [(-targetValidFromUTC timestamp )]
           [(-slice slice_size
   ```

   Where:

   -node *node_name*
   > 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 where Business Process Choreographer is configured and where the template is deployed. 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 where the process template is deployed. This is required if the Business Process Choreographer is configured for a cluster. You can specify the cluster name or the server name and node name.

-templateName *template_name*

The name of the process template to be migrated.

-sourceValidFromUTC *timestamp*

The timestamp specifies which version of the named template will have its instances migrated.

The *timestamp* string specifies the date from which the template is valid, in Coordinated Universal Time (UTC), and must have the following format: 'yyyy-mm-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-01-31T13:40:50`. In the administrative console this date is displayed in local time of the server, so make sure that you take the server time zone into account.

-targetValidFromUTC *timestamp*

This optionally specifies which version of the named process template the instances will be migrated to. If this parameter is not specified, the latest available version of the template will be used. The *timestamp* string has the same format as for the sourceValidFromUTC parameter.

-slice *slice_size*

This parameter is optional. The value *slice_size* specifies how many process instances are migrated in one transaction. The default value is 10.

3. When the script runs, it outputs the name of the node and server where the migration is running. Check the `SystemOut.log` file on the server to see the progress information and whether the migration of any instances caused any exceptions. For example, because instances are not in a suitable state or because a problem occurred during migration.

Results

The instances have been migrated to the new template version.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Querying and replaying failed messages, using administrative scripts

Use the administrative scripts 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. Do not use the wsadmin `-conntype none` option.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

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 administrative scripts are located.

   Linux   **UNIX**   On Linux and UNIX platforms, enter the following command:

   cd *install_root*/ProcessChoreographer/admin

   Windows   On Windows platforms, enter the following command:

   cd *install_root*\ProcessChoreographer\admin

2. Query the number of failed messages on both the retention and hold queues.

   Linux   **UNIX**   On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh -f queryNumberOfFailedMessages.py
            [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
            [ -bfm | -htm ]
   ```

   Windows   On Windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin -f queryNumberOfFailedMessages.py
            [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
            [ -bfm | -htm ]
   ```

   Where:

   -node *nodeName*
   > The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   > The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   > The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

   -bfm | -htm
   > These keywords are optional and mutually exclusive. 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.

If you want to check for a server on the local node, enter:

```
wsadmin -f queryNumberOfFailedMessages.py -server serverName
```

3. Replay all failed messages on the hold queue, retention queue, or both queues.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

```
install_root/bin/wsadmin.sh -f replayFailedMessages.py
        (([-node nodeName] -server server_name) | (-cluster cluster_name))
        -queue replayQueue
        [ -bfm | -htm ]
```

   `Windows` On Windows platforms, enter the following command:

```
install_root\bin\wsadmin -f replayFailedMessages.py
        (([-node nodeName] -server server_name) | (-cluster cluster_name))
        -queue replayQueue
        [ -bfm | -htm ]
```

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)

-node *nodeName*
> The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

-server *serverName*
> The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

-cluster *clusterName*
> The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

-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.

Refreshing people query results, using administrative scripts

The results of a people query are static. Use the administrative scripts to refresh people queries.

Before you begin

The following conditions must be met:

- The application server, on which the messages are to be queried or replayed, must be running because a server connection is required.

- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

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 administrative scripts are located.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   `cd install_root/ProcessChoreographer/admin`

   `Windows` On Windows platforms, enter the following command:

   `cd install_root\ProcessChoreographer\admin`

2. Force the people assignment to be evaluated again.

   - `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

     ```
     install_root/bin/wsadmin.sh -f refreshStaffQuery.py
             [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
             [-processTemplate templateName |
             (-taskTemplate templateName [-nameSpace nameSpace]) |
              -userlist username{,username}...]
     ```

   - `Windows` On Windows platforms, enter the following command:

     ```
     install_root\bin\wsadmin -f refreshStaffQuery.py
             [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
             [-processTemplate templateName |
             (-taskTemplate templateName [-nameSpace nameSpace]) |
              -userlist username{,username}...]
     ```

   Where:

   -node *nodeName*
   > The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   > The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   > The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

-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. The refresh is not performed for the default user, but for the staff queries that model task roles. If the refresh fails, then the queries for the fallback user are not refreshed, for example, for the process administrators.

-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.

>   **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.

3. Optional: If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the `SystemOut.log` file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the `soap.client.props` file, or adjusting the script parameters to reduce the amount of work done on the server.

Related tasks

"Refreshing people query results, using the administrative console" on page 323
The results of a people query are static. Use the administrative console to refresh people queries.

"Refreshing people query results, using the refresh daemon" on page 324
Use this method if you want to set up a regular and automatic refresh of all expired people query results.

Deleting Business Process Choreographer objects

Various database objects accumulate in a running system, for example, audit log entries, task and process instances, task and process templates, reporting data, and people queries. Regularly running administrative scripts to delete objects that are no longer needed from the Business Process Choreographer databases can help prevent wasting storage space.

Related concepts

"Cleanup procedures for Business Process Choreographer" on page 315
An overview of the runtime objects that can be deleted from the database after they are no longer needed, and the tools available.

Deleting audit log entries, using administrative scripts

Use the administrative scripts 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, on 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.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

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 administrative scripts are located.

   Linux **UNIX** On Linux and UNIX platforms, enter the following command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   Windows On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

2. Delete the entries in the audit log table.

   Windows On Windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin -f deleteAuditLog.py
           (([-node nodeName] -server server_name) | (-cluster cluster_name))
           ( -all | -timeUTC timestamp | -timeLocal timestamp
                  | -processtimeUTC timestamp | -processtimeLocal timestamp )
           [-slice size]
   ```

   Linux **UNIX** On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh -f deleteAuditLog.py
           (([-node nodeName] -server server_name) | (-cluster cluster_name))
           ( -all | -timeUTC timestamp | -timeLocal timestamp
                  | -processtimeUTC timestamp | -processtimeLocal timestamp )
           [-slice size]
   ```

   Where:

   -cluster *cluster_name*
     The name of the cluster where Business Process Choreographer is configured. Required if the Business Flow Manager is configured for a WebSphere cluster.

   -node *node_name*
     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 where Business Process Choreographer is configured. Required if the cluster name is not specified.

   -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.

-timeLocal *timestamp*

Use this option to specify the deletion cutoff date and local time on the server. Only audit log entries that are older than the time you specify for *timestamp* are deleted. 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 local time on the server.

-timeUTC *timestamp*

Use this option to specify the deletion cutoff date and time in Coordinated Universal Time (UTC). Only audit log entries that are older than the time you specify for *timestamp* are deleted. 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 UTC.

-processTimeLocal *timestamp*

Use this option to specify the deletion cutoff date and local time on the server. Only audit log entries that belong to a process that finished before the time you specify for *timestamp* are deleted. 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 local time on the server.

-processTimeUTC *timestamp*

Use this option to specify the deletion cutoff date and time in UTC. Only audit log entries that belong to a process that finished before the time you specify for *timestamp* are deleted. 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 UTC.

-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.

The -timeLocal, -timeUTC, -processTimeLocal, and -processTimeUTC options are mutually exclusive.

3. Optional: If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the SystemOut.log file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the soap.client.props file, or adjusting the script parameters to reduce the amount of work done on the server.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Deleting completed process instances

Use an administrative script 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

The following conditions must be met:

- The application server, on which instances are to be deleted, must be running. Do not use the wsadmin -conntype none option, because a server connection is required.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin -user and -password options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin -profileName *profile* option to specify the profile.

About this task

A top-level process instance is considered completed if it is in one of the following end states: finished, terminated, or failed. You specify criteria to selectively delete top-level process instances and all their associated data (such as activity instances, child process instances, and inline task instances) from the database.

Procedure

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

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   `cd *install_root*/ProcessChoreographer/admin`

   `Windows` On Windows platforms, enter the following command:

   `cd *install_root*\ProcessChoreographer\admin`

2. Delete process instances from the database.

   - `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

     ```
     install_root/bin/wsadmin.sh –f deleteCompletedProcessInstances.py
       (([-node nodeName] -server server_name) | (-cluster cluster_name))
       (-all | [-finished] [-terminated] [-failed] )
       [-templateName templateName
       [-validFromUTC timestamp]]
       [-startedBy userID]
       [(-completedAfterLocal timestamp)|(-completedAfterUTC timestamp)]
       [(-completedBeforeLocal timestamp)|(-completedBeforeUTC timestamp)]
     ```

   - `Windows` On Windows platforms, enter the following command:

     ```
     install_root\bin\wsadmin –f deleteCompletedProcessInstances.py
       (([-node nodeName] -server server_name) | (-cluster cluster_name))
       (-all | [-finished] [-terminated] [-failed] )
       [-templateName templateName
       [-validFromUTC timestamp]]
       [-startedBy userID]
       [(-completedAfterLocal timestamp)|(-completedAfterUTC timestamp)]
       [(-completedBeforeLocal timestamp)|(-completedBeforeUTC timestamp)]
     ```

   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 where Business Process Choreographer is

configured. 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 where Business Process Choreographer is configured. Required if the Business Process Choreographer is configured on a 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. The `-all` option means all end states;`finished`, `terminated`, and `failed`. If you do not specify `-all`, you must specify one or more of the end states.

-templateName *templateName*
> Optionally, specifies the name of the process template whose instances will be deleted. If there are multiple process templates sharing the same name but with different `validFrom` dates the instances for all process templates with that name are deleted unless you use the validFrom parameter to specify a particular template.

-validFromUTC *timestamp*
> The date from which the template is valid in Coordinated Universal Time (UTC). This option can only be used with the templateName option. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-11-20T12:00:00`.

-startedBy *userID*
> Optionally, only deletes completed process instances that were started by the given User ID.

-completedAfterLocal *timestamp*
> Optionally, specifies that only instances that completed after the given local time are deleted. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-11-20T12:00:00`. If you only specify a date, the time will default to `00:00:00` local time on the server.

-completedAfterUTC *timestamp*
> Optionally, specifies that only instances that completed after the given time in Coordinated Universal Time are deleted. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-11-20T12:00:00`. If you only specify a date, the time will default to `00:00:00` UTC.

-completedBeforeLocal *timestamp*
> Optionally, specifies that only instances that completed before the given local time are deleted. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-11-20T12:00:00`. If you only specify a date, the time will default to `00:00:00` local time on the server.

-completedBeforeUTC *timestamp*
> Optionally, specifies that only instances that completed before the given time in Coordinated Universal Time are deleted. The *timestamp* string has the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-11-20T12:00:00`. If you only specify a date, the time will default to `00:00:00` UTC.

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 -f deleteCompletedProcessInstances.py
        -node myNode -server myServer
        -finished
        -startedBy Antje
```

3. Optional: If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the SystemOut.log file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the soap.client.props file, or adjusting the script parameters to reduce the amount of work done on the server.

Results

The completed process instances have been deleted from the database.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Deleting completed task instances

Use an administrative script to selectively delete from the Business Process Choreographer database any top-level task instances that have reached an end state of finished, terminated, expired, or failed.

Before you begin

The following conditions must be met:

- The application server, on which instances are to be deleted, must be running. Do not use the wsadmin -conntype none option, because a server connection is required.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin -user and -password options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin -profileName *profile* option to specify the profile.

About this task

A top-level task instance is considered completed if it is in one of the following end states: finished, terminated, expired, or failed. You specify criteria to selectively delete top-level task instances and all their associated data (such as instance custom properties, escalation instances, subtask instances and follow-on task instances) from the database. If a top-level task instance is in the forwarded state, then it is only considered to be completed, if its follow-on task is in one of the above end states.

Sometimes follow-on task instances form a chain of follow-on tasks; where all but the last task instance are in the forwarded state and the last task instance in the chain is in some other state. In this case, a top-level task instance that is in the

forwarded state is considered to be completed if the last task instance in the chain is in one of the following end states: finished, terminated, expired, or failed.

Normally, an inline task instance is not considered to be a top-level task instance, and cannot be deleted using the deleteCompletedTaskInstances.py script because the inline task instance belongs to a business process, which means you must use deleteCompletedProcessInstances.py to delete the completed process instance that the inline task belongs to. However, any inline invocation task instance that was created using the Human Task Manager API or the Service Component Architecture (SCA) API is treated as a top-level task instance, and can be deleted using the deleteCompletedTaskInstances.py script.

Procedure

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

   Linux **UNIX** On Linux and UNIX platforms, enter the following command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   Windows On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

2. Delete task instances from the database.

   - **Linux** **UNIX** On Linux and UNIX platforms, enter the following command:

     ```
     install_root/bin/wsadmin.sh –f deleteCompletedTaskInstances.py
       (([-node nodeName] -server serverName) | (-cluster clusterName))
       (-all | [-finished] [-terminated] [-failed] [-expired] )
       [-templateName templateName -nameSpace nameSpace
       [-validFromUTC timestamp]]
       [-createdBy userID ]
       [(-completedAfterLocal timeStamp)|(-completedAfterUTC timeStamp)]
       [(-completedBeforeLocal timeStamp)|(-completedBeforeUTC timeStamp)]
     ```

   - **Windows** On Windows platforms, enter the following command:

     ```
     install_root\bin\wsadmin –f deleteCompletedTaskInstances.py
       (([-node nodeName] -server serverName) | (-cluster clusterName))
       (-all | [-finished] [-terminated] [-failed] [-expired] )
       [-templateName templateName -nameSpace nameSpace
       [-validFromUTC timestamp]]
       [-createdBy userID ]
       [(-completedAfterLocal timeStamp)|(-completedAfterUTC timeStamp)]
       [(-completedBeforeLocal timeStamp)|(-completedBeforeUTC timeStamp)]
     ```

   Where:

   -node *nodeName*
   > Optional when specifying the server name. This name identifies the node. The default is the local node. You can either specify the server name and node name or the cluster name.

   -server *serverName*
   > The name of the server where Business Process Choreographer is configured. Required if the cluster name is not specified.

   -cluster *clusterName*
   > The name of the cluster where Business Process Choreographer is configured. Required if the Business Process Choreographer is configured on a cluster. You can either specify the cluster name or the server name and node name.

-all | [-finished] [-terminated] [-failed] [-expired]
> Specifies which task instances are to be deleted according to their state. The `-all` option means all end states: `finished`, `terminated`, `failed`, and `expired`. If you do not specify `-all`, you must specify one or more of the end states.

-templateName *templateName*
> Optionally, specifies the name of the task template whose instances will be deleted. If you specify this option you must also specify the nameSpace parameter. If there are multiple task templates sharing the same name but with different `validFromUTC` dates the instances for all task templates with that name are deleted unless you use the validFromUTC parameter to specify a particular template.

-nameSpace *nameSpace*
> Optionally, specifies the namespace of the task template to be deleted. If you specify this option you must also specify the templateName parameter. If there are multiple task templates sharing the same name but with different `validFromUTC` dates the instances for all task templates with that name are deleted unless you use the validFromUTC parameter to specify a particular template.

-validFromUTC *timestamp*
> The date and time from which the template is valid. This option can only be used with the templateName and nameSpace options. The *timestamp* string must be specified in Coordinated Universal Time (UTC) in the following format: 'yyyy-MM-ddThh:mm:ss' (year, month, day, T, hours, minutes, seconds). For example, `2009-11-20T12:00:00`.

-createdBy *userID*
> Optionally, only deletes completed task instances that were created by the given User ID.

-completedAfterLocal *timestamp*
> Optionally, specifies that only instances that completed after the given local time on the server are deleted. A top-level task instance in the forwarded state is considered to have completed after the given time if the last task instance in its chain of follow-on tasks was completed after the given time. The format for the *timestamp* string is the same as for -validFromUTC, except that the time part is optional for this parameter. If you only specify a date, the time defaults to `00:00:00` local time on the server.

-completedAfterUTC *timestamp*
> Optionally, specifies that only instances that completed after the given UTC time are deleted. A top-level task instance in the forwarded state is considered to have completed after the given time if the last task instance in its chain of follow-on tasks was completed after the given time. The format for the *timestamp* string is the same as for -validFromUTC, except that the time part is optional for this parameter. If you only specify a date, the time defaults to `00:00:00` UTC .

-completedBeforeLocal *timestamp*
> Optionally, specifies that only instances that completed before the given local time on the server are deleted. A top-level task instance in the forwarded state is considered to have completed before the given time if the last task instance in its chain of follow-on tasks was completed before the given time. The format for the *timestamp* string is the same as for

-validFromUTC, except that the time part is optional for this parameter. If you only specify a date, the time defaults to `00:00:00` local time on the server.

-completedBeforeUTC *timestamp*
Optionally, specifies that only instances that completed before the given UTC time are deleted. A top-level task instance in the forwarded state is considered to have completed before the given time if the last task instance in its chain of follow-on tasks was completed before the given time. The format for the *timestamp* string is the same as for -validFromUTC, except that the time part is optional for this parameter. If you only specify a date, the time defaults to `00:00:00` UTC .

For example, to delete the task instances running on node *myNode* in server *myServer* that are in the finished state, and were created by the user Erich, run the following command:

```
wsadmin –f deleteCompletedTaskInstances.py
        -node myNode -server myServer
        -finished
        -createdBy Erich
```

3. Optional: If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the `SystemOut.log` file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the `soap.client.props` file, or adjusting the script parameters to reduce the amount of work done on the server.

Results

The completed task instances have been deleted from the database.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Deleting data from the reporting database

Use an administrative script to selectively delete from the reporting database of Business Process Choreographer Explorer, all of the data for process instances that match specified conditions. Deleting unnecessary data can improve the performance generating reports.

Before you begin

The following conditions must be met:
- This script must be run in connected mode, that is, the application server must be running. Do not use the wsadmin `-conntype none` option.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

About this task

You can delete reporting information for process instances from the reporting database in three ways:

- To delete reporting data for process instances that have reached the end state deleted before a specified time, you must provide one of the following parameters: `-deletedBeforeUTC` or `-deletedBeforeLocal`.
- To delete reporting data for process instances of a specific template version regardless of its current state, you must provide both of the following parameters: `-templateName` and `-validFromUTC`.
- To delete reporting 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* `-validFromUTC` *timestamp* `-state` *state* `-reachedBeforeUTC` *timestamp* and `-reachedBeforeUTC` *timestamp* or `-reachedBeforeLocal` *timestamp,* where `-templateName` *template_name* and `-validFromUTC` *timestamp* are optional.

To use any one of these ways, perform the following:

Procedure

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

   Linux UNIX On Linux and UNIX platforms, enter the following command:

   cd *install_root*/ProcessChoreographer/admin

   Windows On Windows platforms, enter the following command:

   cd *install_root*\ProcessChoreographer\admin

2. Enter the command to delete reporting data for specific process instances from the database.

   - Linux UNIX On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh
       -f observerDeleteProcessInstanceData.py
       ( [-node node_name] -server server_name) | (-cluster cluster_name )
       [ -dataSource dataSource_JNDI_name ]
       [ -dbSchemaName dbSchemaName]
       (
         ( (-deletedBeforeLocal timestamp)|(-deletedBeforeUTC timestamp) )
         | ( -templateName template_name -validFromUTC timestamp )
         | ( -force [-templateName template_name -validFromUTC timestamp]
            -state state (-reachedBeforeLocal timestamp)|
                                          (-reachedBeforeUTC timestamp) ) ) )
   ```

   - Windows On Windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin
       -f observerDeleteProcessInstanceData.py
       ( [-node node_name] -server server_name) | (-cluster cluster_name )
       [ -dataSource dataSource_JNDI_name ]
       [ -dbSchemaName dbSchemaName]
       (
         ( (-deletedBeforeLocal timestamp)|(-deletedBeforeUTC timestamp) )
         | ( -templateName template_name -validFromUTC timestamp )
         | ( -force [-templateName template_name -validFromUTC timestamp]
            -state state (-reachedBeforeLocal timestamp)|
                                          (-reachedBeforeUTC timestamp) ) ) )
   ```

   Where:

   -node *node_name*
   >   This name identifies the node where the Business Process Choreographer

Event Collector is configured. The default is the local node. This parameter is optional if you specify the server parameter. Do not specify the node if you provide the cluster parameter.

-server *server_name*
> The name of the server where the Business Process Choreographer Event Collector is configured. 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 where the Business Process Choreographer Event Collector is configured. If you specify this parameter, you must not specify the server parameter.

-dataSource *datasource_JNDI_name*
> Because a server or cluster can have multiple reporting databases, this parameter identifies which database the command will act on. The default is jdbc/BPEDB.

-dbSchemaName *dbSchemaName*
> Use this parameter if the reporting database is set up with a specific schema name.

-deletedBeforeLocal
> Deletes all reporting data for process instances that reached the state deleted before the specified local time. The date and time, *timestamp*, are expressed 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 local time on the server.

-deletedBeforeUTC *timestamp*
> Deletes all reporting data for process instances that reached the state deleted before the specified time in Coordinated Universal Time (UTC). The date and time, *timestamp*, are expressed 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 UTC.

-templateName *template_name*
> Deletes all reporting data for instances that belong to the specified template version.

> **-validFromUTC** *timestamp*
> > If you specify the templateName option, this option is required to distinguish between different possible versions of the template. The date and time, *timestamp*, 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 reporting 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 either -reachedBeforeLocal or -reachedBeforeUTC. The options -templateName and -validFromUTC are optional.

> **-state** *state*
> > Specifies one of the following states: running, terminated, suspended, failed, finished, compensated.

-reachedBeforeLocal *timestamp*

Specifies a cutoff time in the server's local time, before which the specified state must have been reached. The date and time, *timestamp*, are expressed 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` local time on the server.

-reachedBeforeUTC *timestamp*

Specifies a cutoff time in UTC, before which the specified state must have been reached. The date and time, *timestamp*, are expressed 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` UTC.

For example, to delete all reporting data for instances of the process template *my_template*, which is valid from midday on January 2, 2007 UTC, that are running on node *my_node* in server *my_server* that were started before midday (local time on the server) on July 20, 2007, run the following command:

```
wsadmin —f observerDeleteProcessInstanceData.py
        -node my_node -server my_server
        -force -templateName my_template -validFromUTC 2007-01-02T12:00:00
                -state running -reachedBeforeLocal 2007-07-20T12:00:00
```

Results

If successful, the tool reports the number of instances for which reporting 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.

What to do next

If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the `SystemOut.log` file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the `soap.client.props` file, or adjusting the script parameters to reduce the amount of work done on the server.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Deleting process templates that are no longer valid

Use the administrative script to delete, from the Business Process Choreographer database, business process templates that are no longer valid.

Before you begin

The following conditions must be met:
- This script must be run in connected mode, that is, the application server on which templates are to be deleted must be running.
- You must have either operator or deployer authority.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.

- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

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 administrative scripts are located.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   `Windows` On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

2. Delete, from the database, process templates that are no longer valid.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh -f deleteInvalidProcessTemplate.py
           (([-node nodeName] -server server_name) | (-cluster cluster_name))
           -templateName templateName
           -validFromUTC timestamp
   ```

   `Windows` On Windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin -f deleteInvalidProcessTemplate.py
           (([-node nodeName] -server server_name) | (-cluster cluster_name))
           -templateName templateName
           -validFromUTC timestamp
   ```

   Where:

   -cluster *cluster_name*
   The name of the cluster where Business Process Choreographer is configured. Required if the Business Process Choreographer is configured on a cluster. You can specify the cluster name or the server name and node name.

   -node *node_name*
   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 where Business Process Choreographer is

ocnfigured. 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 to be deleted.

-validFromUTC *timestamp*
The date and time from which the template is valid in Coordinated Universal Time (UTC). 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

3. Optional: If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the `SystemOut.log` file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the com.ibm.SOAP.requestTimeout property in the `soap.client.props` file, or adjusting the script parameters to reduce the amount of work done on the server.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Deleting human task templates that are no longer valid

Use the administrative scripts to delete, from the Business Process Choreographer database, human task templates that are no longer valid.

Before you begin

The following conditions must be met:

- This script must be run in connected mode, that is, the application server on which templates are to be deleted must be running.
- You must have either operator or deployer authority.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If you are not working with the default profile, use the wsadmin **-profileName** *profile* option to specify the profile.

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 administrative scripts are located.

   `Linux`　`UNIX`　On Linux and UNIX platforms, enter the following command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   `Windows`　On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

2. Delete, from the database, human task templates that are no longer valid.

   `Linux`　`UNIX`　On UNIX and Linux platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh -f deleteInvalidTaskTemplate.py
       (([-node nodeName] -server server_name) | (-cluster cluster_name))
       -templateName templateName
       -validFromUTC timestamp
       -nameSpace nameSpace
   ```

   `Windows`　On Windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin -f deleteInvalidTaskTemplate.py
       (([-node nodeName] -server server_name) | (-cluster cluster_name))
       -templateName templateName
       -validFromUTC timestamp
       -nameSpace nameSpace
   ```

   Where:

   -cluster *cluster_name*
   > The name of the cluster where Business Process Choreographer is configured. Required if Business Process Choreographer is configured on a cluster. You can specify the cluster name or the server name and node name.

   -node *node_name*
   > 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 where Business Process Choreographer is configured. 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 task template to be deleted.

   -validFromUTC *timestamp*
   > The date and time from which the template is valid in Coordinated Universal Time (UTC). 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.

3. Optional: If the script triggers long-running work on the server, the script might fail if the connection timeout is not long enough to complete the action. Check the `SystemOut.log` file on the server to see whether you need to restart the script. If the timeout happens often, consider increasing the value of the

com.ibm.SOAP.requestTimeout property in the `soap.client.props` file, or adjusting the script parameters to reduce the amount of work done on the server.

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on the client, interface, or parameter name being used.

Removing unused people query results, using administrative scripts

Use the administrative scripts to remove unused people query results from the database.

Before you begin

The following conditions must be met:

- The application server, through which unused people queries are to be deleted, must be running because a server connection is required.
- If Business Process Choreographer is configured on a cluster, then at least one cluster member must be running.
- If WebSphere administrative security is enabled, and your user ID does not have operator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has operator authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

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 administrative scripts are located.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   `cd install_root/ProcessChoreographer/admin`

   `Windows` On Windows platforms, enter the following command:

   `cd install_root\ProcessChoreographer\admin`

2. Remove the unused lists of people.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh -f cleanupUnusedStaffQueryInstances.py
        ( [-node node_name] -server server_name) | (-cluster cluster_name )
   ```

   `Windows` On Windows platforms, enter the following command:

```
install_root\bin\wsadmin -f cleanupUnusedStaffQueryInstances.py
      ( [-node node_name] -server server_name) | (-cluster cluster_name )
```

Where:

-cluster *clusterName*
> The name of the cluster where Business Process Choreographer is
> configured. Required if Business Process Choreographer is configured on a
> cluster.

-node *nodeName*
> Optional when specifying the server name. This name identifies the node.
> The default is the local node.

-server *serverName*
> The name of the server where Business Process Choreographer is
> configured. Required if the cluster name is not specified.

Results

The number of entries deleted from the database is displayed.

Administering query tables

Use the administrative script, manageQueryTable.py script to administer query
tables in Business Process Choreographer, which were developed using the Query
Table Builder. Unlike predefined query tables, which are available out-of-the-box,
you must deploy composite and supplemental query tables on WebSphere Process
Server before you can use them with the query table API.

About this task

When query tables are deployed, the query table definition is stored in the
Business Process Choreographer database. Additional database artifacts are not
created in the current version of WebSphere Process Server. Any changes to
composite and supplemental query tables, including deployment, update, and
undeployment, are visible to the query table API without restarting the server.

Query tables are deployed on a stand-alone server that is running or in a cluster
with at least one member running. The undeployment of supplemental and
composite query tables is performed also on running servers. For supplemental
query tables, the related physical database objects, typically a database view or
database table, must be created if they do not exist before the usage of the query
table.

For supplemental query tables, the user, or administrator, is responsible for
providing the related database table or view.

For composite query tables, the information is composed of the existing database
tables or views that relate to the predefined or supplemental query tables. Data is
not duplicated in the current version of WebSphere Process Server.

Supplemental query tables which are referenced by deployed composite query
tables must not be updated or undeployed.

Using manageQueryTable.py you can update composite and supplemental query
tables and get their XML definitions. You can also get a list of query tables that are
available on your system. For supplemental query tables, the related physical
database objects, typically a database view or a database table, must be created if

Administering Business Process Choreographer   **353**

they do not exist before the usage of the query table.

Related reference

"manageQueryTable.py script"
The manageQueryTable.py script deploys, undeploys, and updates query tables in Business Process Choreographer. It is also used to list query tables and to list the XML definition of a query table.

manageQueryTable.py script

The manageQueryTable.py script deploys, undeploys, and updates query tables in Business Process Choreographer. It is also used to list query tables and to list the XML definition of a query table.

Purpose

The manageQueryTable.py script is run in batch mode. Use this script to deploy, undeploy, or update query tables in Business Process Choreographer. You can also get a list of deployed query tables and get the XML definition of a query table.

Location

The manageQueryTable.py script is located in the Business Process Choreographer `admin` directory:

- � Linux ▪ UNIX On Linux and UNIX platforms: In the directory `install_root`/ProcessChoreographer/admin

- ▪ Windows On Windows platforms: In the directory `install_root\ProcessChoreographer\admin`

Running the script in a stand-alone server environment

The configuration script is run using the wsadmin command. In a stand-alone server environment:

- This script must be run in connected mode, that is, the application server must be running.
- If WebSphere administrative security is enabled, include the `-user` and `-password` options. To deploy, undeploy, or update query tables, the user specified must have administrator or deployer authority. To list the XML definition of a query table or to get a list of query tables, the user specified must have operator, administrator, or deployer authority.
- If you are not configuring the default profile, add the `-profileName` option.

Running the script in a network deployment environment

The configuration script is run using the wsadmin command. In a network deployment environment:

- Run the script on the deployment manager node.
- This script must be run in connected mode, that is, the application server or at least one cluster member and the deployment manager must be running.
- If WebSphere administrative security is enabled, include the `-user` and `-password` options. To deploy, undeploy, or update query tables, the user specified must have administrator or deployer authority. To list the XML definition of a query table or to get a list of query tables, the user specified must have operator, administrator, or deployer authority.
- If you are not configuring the default profile, add the `-profileName` option.

Running the script

 On Linux and UNIX platforms, enter the following command:

install_root/bin/wsadmin.sh —f manageQueryTable.py *parameters*

Windows On windows platforms, enter the following command:

install_root\bin\wsadmin —f manageQueryTable.py *parameters*

Parameters

You can supply the following parameters:

```
[ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
( ( -deploy (qtdFile | jarFile)) |
  ( -undeploy queryTableName)   |
  ( -update definition (qtdFile | jarFile)) |
  ( -query names -kind (composite|predefined|supplemental)) |
  ( -query definition -name queryTableName))
```

-node *nodeName*
> The name of the node where Business Process Choreographer is configured.
> This is optional when specifying the server name. The default is the local node.

-server *serverName*
> The name of the server where Business Process Choreographer is configured.
> This is required a cluster name is not specified.

-cluster *clusterName*
> The name of the cluster where Business Process Choreographer is configured.
> This is required if Business Process Choreographer is configured on a cluster.

-deploy *qtdFile* | *jarFile*
> The file name, including the fully qualified path, of either the query table
> definition XML file to be deployed or a JAR file that contains the definitions.
> Use this option to deploy a query table. On Windows, you must use either "/"
> or "\\\\" as the path separator. For example, to specify the file
> c:\temp\myQueryTable.qtd you must specify it as c:/temp/myQueryTable.qtd or
> c:\\\\temp\\\\myQueryTable.qtd.

-undeploy *queryTableName*
> The name of the query table. Use this option to undeploy a query table.

-update definition *qtdFile* | *jarFile*
> The file name, including the fully qualified path, of either the query table
> definition XML file to be updated or a JAR file that contains the definitions.
> Use this option to update an existing query table. On Windows, you must use
> either "/" or "\\\\" as the path separator. For example, to specify the file
> c:\temp\myQueryTable.qtd you must specify it as c:/temp/myQueryTable.qtd or
> c:\\\\temp\\\\myQueryTable.qtd.
>
> If a JAR file is provided, it can contain multiple QTD files and property files
> for each QTD file, which contain display names and descriptions. Use the
> Query Table Builder to export query table definitions as a JAR file.

-query names –kind {composite|predefined|supplemental}
> The type of query table: composite, predefined, or supplemental. Use this
> option to list the names of deployed query tables of a particular type.

-query definition –name *queryTableName*
> The name of the query table, in upper case. Use this option to list the XML
> definition of a deployed supplemental or composite query table.

-profileName *profileName*
>   The name of a user-defined profile. Specify this option if you are not working
>   with the default profile.

Related tasks

"Deploying composite and supplemental query tables"
Before supplemental and composite query tables can be used in Business Process
Choreographer, use the manageQueryTable.py script to deploy them. Before query
tables can be used with the query table API they must be deployed on the related
Business Process Choreographer container. Query tables do not need to be started,
and the server or cluster does not need to be restarted for them to be available
after deployment.

"Undeploying composite and supplemental query tables" on page 358
Use the manageQueryTable.py script to remove composite and supplemental query
tables in Business Process Choreographer.

"Updating composite and supplemental query tables" on page 360
Use the manageQueryTable.py script to update composite and supplemental query
tables in Business Process Choreographer. Updates to query tables can be made
while applications are running, and they are available after the update, without
restarting the server or cluster.

"Retrieving a list of query tables" on page 362
Use the manageQueryTable.py script to get a list of query tables that are available
in Business Process Choreographer. You can list predefined, supplemental, and
composite query tables.

"Administering query tables" on page 353
Use the administrative script, manageQueryTable.py script to administer query
tables in Business Process Choreographer, which were developed using the Query
Table Builder. Unlike predefined query tables, which are available out-of-the-box,
you must deploy composite and supplemental query tables on WebSphere Process
Server before you can use them with the query table API.

"Retrieving the XML definitions of query tables" on page 364
Use the manageQueryTable.py script to get the XML definition of composite and
supplemental query tables in Business Process Choreographer. You cannot use the
script to retrieve the XML definitions of predefined query tables. The XML format
of query table definitions is not a published interface. It is not supported to
manually change the definition of a query table definition. To do this, load the
query table definition into the Query Table Builder and apply modifications there.

Deploying composite and supplemental query tables

Before supplemental and composite query tables can be used in Business Process
Choreographer, use the manageQueryTable.py script to deploy them. Before query
tables can be used with the query table API they must be deployed on the related
Business Process Choreographer container. Query tables do not need to be started,
and the server or cluster does not need to be restarted for them to be available
after deployment.

Before you begin

Before you begin this procedure, the following conditions must be met:
- The application server on which the query tables are to be deployed 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.

- If WebSphere administrative security is enabled, and your user ID does not have administrator or deployer authority, include the wsadmin `-user` and `-password` options to specify a user ID that has administrator or deployer authority.

About this task

Complete the following steps to deploy composite and supplemental query tables in Business Process Choreographer.

Procedure

1. Change to the Business Process Choreographer subdirectory where the manageQueryTable.py script is located.

   - **Linux** **UNIX** On Linux and UNIX platforms: In the directory *install_root*/ProcessChoreographer/admin

   - **Windows** On Windows platforms: In the directory *install_root*\ProcessChoreographer\admin

2. To deploy a query table file or a JAR file:

   Linux **UNIX** On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh —f manageQueryTable.py
         [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
         -deploy qtdFile|jarFile
   ```

   Windows On windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin —f manageQueryTable.py
         [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
         -deploy qtdFile|jarFile
   ```

   Where:

   -node *nodeName*
   : The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   : The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   : The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

   -deploy *qtdFile* | *jarFile*
   : The file name, including the fully qualified path, of either the query table definition XML file to be deployed or a JAR file that contains the definitions.

   For example:

   ```
   wsadmin.sh —f manageQueryTable.py —server server1 —deploy sample.qtd
   ```

Related tasks

"Updating composite and supplemental query tables" on page 360
Use the manageQueryTable.py script to update composite and supplemental query tables in Business Process Choreographer. Updates to query tables can be made while applications are running, and they are available after the update, without restarting the server or cluster.

"Retrieving a list of query tables" on page 362
Use the manageQueryTable.py script to get a list of query tables that are available in Business Process Choreographer. You can list predefined, supplemental, and composite query tables.

"Retrieving the XML definitions of query tables" on page 364
Use the manageQueryTable.py script to get the XML definition of composite and supplemental query tables in Business Process Choreographer. You cannot use the script to retrieve the XML definitions of predefined query tables. The XML format of query table definitions is not a published interface. It is not supported to manually change the definition of a query table definition. To do this, load the query table definition into the Query Table Builder and apply modifications there.

"Undeploying composite and supplemental query tables"
Use the manageQueryTable.py script to remove composite and supplemental query tables in Business Process Choreographer.

Related reference

"manageQueryTable.py script" on page 354
The manageQueryTable.py script deploys, undeploys, and updates query tables in Business Process Choreographer. It is also used to list query tables and to list the XML definition of a query table.

Undeploying composite and supplemental query tables

Use the manageQueryTable.py script to remove composite and supplemental query tables in Business Process Choreographer.

Before you begin

Before you begin this procedure, the following conditions must be met:
- The application server on which the query tables are to be undeployed 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.
- If WebSphere administrative security is enabled, and your user ID does not have administrator or deployer authority, include the wsadmin -user and -password options to specify a user ID that has administrator or deployer authority.
- Ensure that no applications are installed and running that reference a query table that is to be undeployed. If a supplemental query table is undeployed, it must not be referenced, as attached query table, by any composite query table.

About this task

Complete the following steps to undeploy composite and supplemental query tables in Business Process Choreographer.

Procedure

1. Change to the directory where the Business Process Choreographer administrative scripts are located.

- `Linux` `UNIX` On Linux and UNIX platforms: In the directory *install_root*/ProcessChoreographer/admin

- `Windows` On Windows platforms: In the directory *install_root*\ ProcessChoreographer\admin

2. Undeploy the query table:

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   wsadmin.sh –f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
           -undeploy queryTableName
   ```

   `Windows` On windows platforms, enter the following command:

   ```
   wsadmin –f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
           -undeploy queryTableName
   ```

   Where:

   -node *nodeName*
   : The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   : The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   : The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

   -undeploy *queryTableName*
   : The name of the query table, in upper case, to be undeployed.

   For example:

   ```
   wsadmin –f manageQueryTable.py –server server1 –undeploy COMPANY.SAMPLE
   ```

Related tasks

"Updating composite and supplemental query tables"
Use the manageQueryTable.py script to update composite and supplemental query tables in Business Process Choreographer. Updates to query tables can be made while applications are running, and they are available after the update, without restarting the server or cluster.

"Retrieving a list of query tables" on page 362
Use the manageQueryTable.py script to get a list of query tables that are available in Business Process Choreographer. You can list predefined, supplemental, and composite query tables.

"Retrieving the XML definitions of query tables" on page 364
Use the manageQueryTable.py script to get the XML definition of composite and supplemental query tables in Business Process Choreographer. You cannot use the script to retrieve the XML definitions of predefined query tables. The XML format of query table definitions is not a published interface. It is not supported to manually change the definition of a query table definition. To do this, load the query table definition into the Query Table Builder and apply modifications there.

"Deploying composite and supplemental query tables" on page 356
Before supplemental and composite query tables can be used in Business Process Choreographer, use the manageQueryTable.py script to deploy them. Before query tables can be used with the query table API they must be deployed on the related Business Process Choreographer container. Query tables do not need to be started, and the server or cluster does not need to be restarted for them to be available after deployment.

Related reference

"manageQueryTable.py script" on page 354
The manageQueryTable.py script deploys, undeploys, and updates query tables in Business Process Choreographer. It is also used to list query tables and to list the XML definition of a query table.

Updating composite and supplemental query tables

Use the manageQueryTable.py script to update composite and supplemental query tables in Business Process Choreographer. Updates to query tables can be made while applications are running, and they are available after the update, without restarting the server or cluster.

Before you begin

Before you begin this procedure, the following conditions must be met:

- The application server on which the query tables are deployed 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.
- If WebSphere administrative security is enabled, and your user ID does not have administrator or deployer authority, include the wsadmin `-user` and `-password` options to specify a user ID that has administrator or deployer authority.
- If you are not working with the default profile, use the wsadmin `-profileName` *profile* option to specify the profile.

About this task

Complete the following steps to update composite and supplemental query tables in Business Process Choreographer.

Procedure

1. Change to the directory where the Business Process Choreographer administrative scripts are located.

   - `Linux` `UNIX` On Linux and UNIX platforms: In the directory *install_root*/ProcessChoreographer/admin

   - `Windows` On Windows platforms: In the directory *install_root*\ProcessChoreographer\admin

2. Update the query table using either a query table definition XML file or a JAR file that contains the definitions. If property files are already deployed, they will be overwritten.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh –f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
             -update definition qtdFile|jarFile
   ```

   `Windows` On windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin –f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
             -update definition qtdFile|jarFile
   ```

   Where:

   -node *nodeName*
   : The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   : The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   : The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

   -update definition *qtdFile* | *jarFile*
   : The file name, including the fully qualified path, of either the query table definition XML file to be updated or a JAR file that contains the definitions.

   For example:

   ```
   wsadmin –f manageQueryTable.py –server server1
   –update definition sample_v2.qtd
   ```

Related tasks

"Retrieving a list of query tables"
Use the manageQueryTable.py script to get a list of query tables that are available in Business Process Choreographer. You can list predefined, supplemental, and composite query tables.

"Retrieving the XML definitions of query tables" on page 364
Use the manageQueryTable.py script to get the XML definition of composite and supplemental query tables in Business Process Choreographer. You cannot use the script to retrieve the XML definitions of predefined query tables. The XML format of query table definitions is not a published interface. It is not supported to manually change the definition of a query table definition. To do this, load the query table definition into the Query Table Builder and apply modifications there.

"Deploying composite and supplemental query tables" on page 356
Before supplemental and composite query tables can be used in Business Process Choreographer, use the manageQueryTable.py script to deploy them. Before query tables can be used with the query table API they must be deployed on the related Business Process Choreographer container. Query tables do not need to be started, and the server or cluster does not need to be restarted for them to be available after deployment.

"Undeploying composite and supplemental query tables" on page 358
Use the manageQueryTable.py script to remove composite and supplemental query tables in Business Process Choreographer.

Related reference

"manageQueryTable.py script" on page 354
The manageQueryTable.py script deploys, undeploys, and updates query tables in Business Process Choreographer. It is also used to list query tables and to list the XML definition of a query table.

Retrieving a list of query tables

Use the manageQueryTable.py script to get a list of query tables that are available in Business Process Choreographer. You can list predefined, supplemental, and composite query tables.

Before you begin

Before you begin this procedure, the following conditions must be met:
- The application server on which the query tables are deployed 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.
- If WebSphere administrative security is enabled, and your user ID does not have operator, administrator, or deployer authority, include the wsadmin -user and -password options to specify a user ID that has operator, administrator, or deployer authority.
- If you are not working with the default profile, use the wsadmin -profileName *profile* option to specify the profile.

About this task

Complete the following steps to get a list of query tables in Business Process Choreographer.

Procedure

1. Change to the directory where the Business Process Choreographer administrative scripts are located.

   - Linux   UNIX   On Linux and UNIX platforms: In the directory *install_root*/ProcessChoreographer/admin

   - Windows   On Windows platforms: In the directory *install_root*\ProcessChoreographer\admin

2. To list query tables in the command prompt window:

   Linux   UNIX   On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh –f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
           -query names
           -kind (composite|predefined|supplemental)
   ```

   Windows   On windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin –f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
           -query names
           -kind (composite|predefined|supplemental)
   ```

   Where:

   -node *nodeName*
   The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

   -kind (composite|predefined|supplemental)
   The type of query table to be listed, whether composite, predefined, or supplemental. If there are no query tables of the selected kind, none is returned.

   For example:

   ```
   wsadmin.sh –f manageQueryTable.py –server server1
           –query names –kind composite
   ```

"Updating composite and supplemental query tables" on page 360
Use the manageQueryTable.py script to update composite and supplemental query
tables in Business Process Choreographer. Updates to query tables can be made
while applications are running, and they are available after the update, without
restarting the server or cluster.

"Retrieving the XML definitions of query tables"
Use the manageQueryTable.py script to get the XML definition of composite and
supplemental query tables in Business Process Choreographer. You cannot use the
script to retrieve the XML definitions of predefined query tables. The XML format
of query table definitions is not a published interface. It is not supported to
manually change the definition of a query table definition. To do this, load the
query table definition into the Query Table Builder and apply modifications there.

"Deploying composite and supplemental query tables" on page 356
Before supplemental and composite query tables can be used in Business Process
Choreographer, use the manageQueryTable.py script to deploy them. Before query
tables can be used with the query table API they must be deployed on the related
Business Process Choreographer container. Query tables do not need to be started,
and the server or cluster does not need to be restarted for them to be available
after deployment.

"Undeploying composite and supplemental query tables" on page 358
Use the manageQueryTable.py script to remove composite and supplemental query
tables in Business Process Choreographer.

Related reference

"manageQueryTable.py script" on page 354
The manageQueryTable.py script deploys, undeploys, and updates query tables in
Business Process Choreographer. It is also used to list query tables and to list the
XML definition of a query table.

Retrieving the XML definitions of query tables

Use the manageQueryTable.py script to get the XML definition of composite and
supplemental query tables in Business Process Choreographer. You cannot use the
script to retrieve the XML definitions of predefined query tables. The XML format
of query table definitions is not a published interface. It is not supported to
manually change the definition of a query table definition. To do this, load the
query table definition into the Query Table Builder and apply modifications there.

Before you begin

Before you begin this procedure, the following conditions must be met:
- The application server on which the query tables are deployed 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.
- If WebSphere administrative security is enabled, and your user ID does not have
  operator, administrator, or deployer authority, include the wsadmin -user and
  -password options to specify a user ID that has operator, administrator, or
  deployer authority.
- If you are not working with the default profile, use the wsadmin -profileName
  profile option to specify the profile.

About this task

Complete the following steps to retrieve the XML definition of composite and supplemental query tables in Business Process Choreographer.

Procedure

1. Change to the directory where the Business Process Choreographer administrative scripts are located.

   - Linux | UNIX On Linux and UNIX platforms: In the directory *install_root*/ProcessChoreographer/admin

   - Windows On Windows platforms: In the directory *install_root*\ProcessChoreographer\admin

2. To list the XML definition of a query table in the command prompt window.

   Linux | UNIX On Linux and UNIX platforms, enter the following command:

   ```
   install_root/bin/wsadmin.sh —f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
           -query definition
           -name queryTableName
   ```

   Windows On windows platforms, enter the following command:

   ```
   install_root\bin\wsadmin —f manageQueryTable.py
           [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
           -query definition
           -name queryTableName
   ```

   Where:

   -node *nodeName*
   > The name of the node where Business Process Choreographer is configured. This is optional when specifying the server name. The default is the local node.

   -server *serverName*
   > The name of the server where Business Process Choreographer is configured. This is required a cluster name is not specified.

   -cluster *clusterName*
   > The name of the cluster where Business Process Choreographer is configured. This is required if Business Process Choreographer is configured on a cluster.

   -name *queryTableName*
   > The name of the query table, in upper case, whose XML definition is to be listed.

   For example:

   ```
   wsadmin.sh —f manageQueryTable.py —server server1
           —query definition —name COMPANY.SAMPLE
   ```

Related tasks

"Updating composite and supplemental query tables" on page 360
Use the manageQueryTable.py script to update composite and supplemental query
tables in Business Process Choreographer. Updates to query tables can be made
while applications are running, and they are available after the update, without
restarting the server or cluster.

"Retrieving a list of query tables" on page 362
Use the manageQueryTable.py script to get a list of query tables that are available
in Business Process Choreographer. You can list predefined, supplemental, and
composite query tables.

"Deploying composite and supplemental query tables" on page 356
Before supplemental and composite query tables can be used in Business Process
Choreographer, use the manageQueryTable.py script to deploy them. Before query
tables can be used with the query table API they must be deployed on the related
Business Process Choreographer container. Query tables do not need to be started,
and the server or cluster does not need to be restarted for them to be available
after deployment.

"Undeploying composite and supplemental query tables" on page 358
Use the manageQueryTable.py script to remove composite and supplemental query
tables in Business Process Choreographer.

Related reference

"manageQueryTable.py script" on page 354
The manageQueryTable.py script deploys, undeploys, and updates query tables in
Business Process Choreographer. It is also used to list query tables and to list the
XML definition of a query table.

Listing templates

The bpcTemplates.jacl script provides an option to list all the process templates
and task templates that are installed and the time from which each template is
valid.

Procedure

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

   `Linux`   `UNIX`   On Linux and UNIX platforms, enter the following
   command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   `Windows`   On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

2. List all the templates.

   `Linux`   `UNIX`   On Linux and UNIX platforms, enter:

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl -list [application_name]
   ```

   `Windows`   On Windows platforms, enter:

   ```
   install_root\bin\wsadmin -f bpcTemplates.jacl -list [application_name]
   ```

   If you specify the optional application name, *application_name*, only templates
   belonging to that application will be listed. Otherwise, the default is to list all
   templates belonging to all applications. The list of templates includes the time
   from which each template is valid. For example:

   ```
   Found 5 template(s):
   JavaSnippetsApp: Process template Process_A, valid from 2008-12-17T11:36:00 (UTC)
   JavaSnippetsApp: Process template Process_B, valid from 2008-12-17T11:36:00 (UTC)
   ```

```
OrderProcessingApp: Process template ChargingProcess, valid from 2009-04-03T19:52:54 (UTC)
OrderProcessingApp: Process template OrderProcess, valid from 2009-04-03T19:52:54 (UTC)
OrderProcessingApp: Process template ShippingProcess, valid from 2009-04-03T19:52:54 (UTC)
Done.
```

Related concepts

"How time zones are handled in Business Process Choreographer" on page 331
When times are displayed or passed as parameters, the time zone used depends on
the client, interface, or parameter name being used.

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 to work with your assigned tasks. 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. Using reporting, you can retrieve statistical information based on these events and create reports on processes and activities.

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 life cycle 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 substitution for users. You can also change the priority and business category for human tasks, and add or update custom properties.

- With the reporting function of Business Process Choreographer Explorer you can monitor the history of process instances, activity instances, or inline human tasks. If your Business Process Choreographer Explorer configuration includes the reporting function 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.

- 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 concepts

"Business Process Choreographer Explorer user interface"
Business Process Choreographer Explorer is a standalone Web application that
provides a set of administration functions for managing business processes and
human tasks and for reporting on process and activity events. The interface
consists of a taskbar, a navigation pane, and the workspace.

Business Process Choreographer Explorer user interface

Business Process Choreographer Explorer is a standalone Web application that
provides a set of administration functions for managing business processes and
human tasks and 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
Explorer user interface.
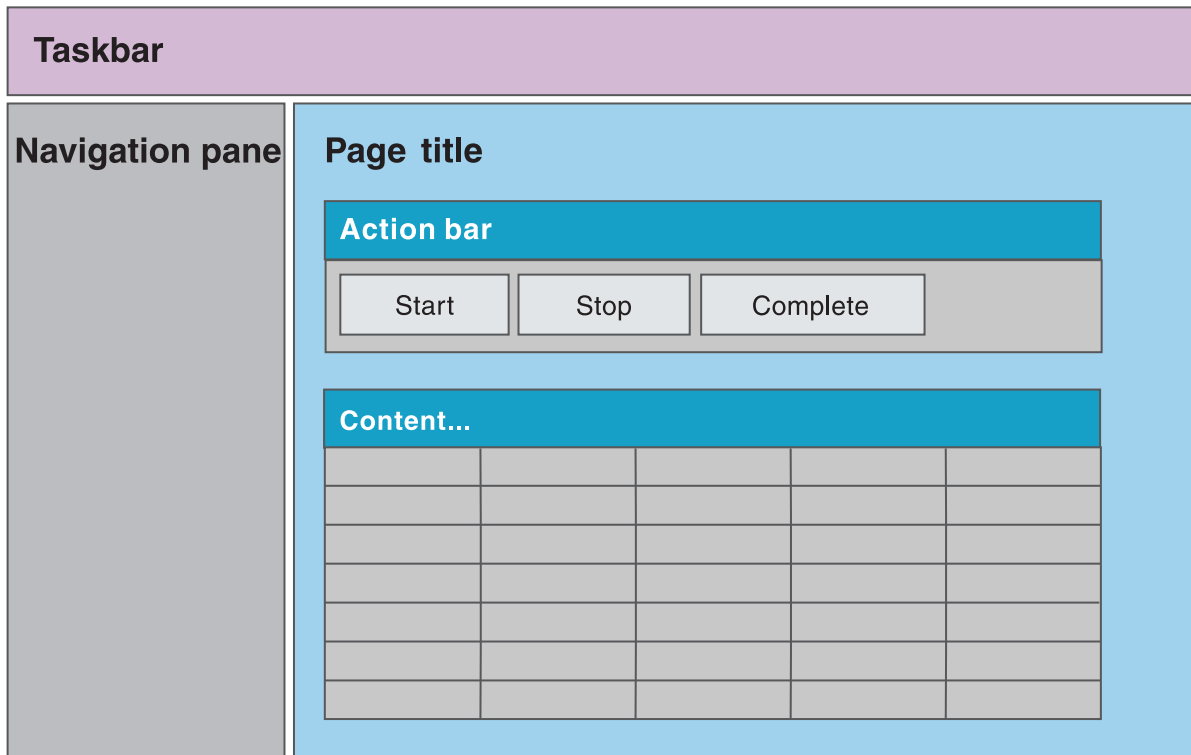
The user interface has the following main areas.

Taskbar

For all users, the taskbar offers options for logging out of Business Process
Choreographer Explorer and for accessing online help. In addition, the options **My
Substitutes** and **Define Substitutes** are available for specifying absence settings.
These options are available when substitution is enabled for the Human Task
Manager in Business Process Choreographer and the Virtual Member Manager
service is configured for WebSphere Application Server security.

My Substitutes
  Select this option to specify substitutes for a user's tasks.

Define Substitutes

Select this option to define absence settings for users.

If you have system administrator rights, the taskbar also includes the following options:

Customize

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

Define Views

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

Navigation pane

If the Views tab is selected, 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 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.

If the Reports tab is selected, the navigation pane 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. Use the predefined lists and charts to get state and event information for runtime entities, for example, to get process and activity snapshot charts. The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

Page title

If the Views tab is selected, 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. For information about a page click the **Help** icon ⓘ on the respective page.

If the Reports tab is selected, the workspace contains pages that you use to view predefined lists and charts, specify report definitions, and to view reports. 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. For information about a page click the **Help** icon ⓘ on the respective page.

Business Process Choreographer Explorer Views tab

Use the Views tab of Business Process Choreographer Explorer 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:

- Collapse and expand a group.

  Click the arrow beside an item in the navigation pane to expand or collapse the item.
- Navigate to a view.

  Click the view name to navigate to that view.
- 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. The **Show pop-up menu** icon ( ) indicates that 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 ( ).

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 in the navigation pane, and do not initially have a pop-up menu. When the navigation pane is changed using **Customize**, these predefined then have the **Predefined view** icon in front of them, which makes it possible to move them up or down.

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.

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. All views show items, independent of additional filters, to which you are authorized. For example, you see only the terminated processes you are allowed to see. 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:

Currently Valid

This view shows a list of process templates are the currently valid version of each process. That is, they are the newest started version of a process whose valid from date is not in the future. 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.

All Versions

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

Process Instances

The process instances group contains the following views:

Started By Me

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

Administered By Me

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

Critical Processes

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

Terminated Processes

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

Failed Compensations

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

Activity Instances

The activity instances group contains the following view:

Stopped Activities

This view shows the activities that are in the stopped state.

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. You can also change the priority of a task and change its business category.

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. You can also change the priority of a task and change its business category.

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. You can also change the priority of a task and change its business category.

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. You can also change the priority of a task and change its business category.

My Escalations

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

Business Process Choreographer Explorer Reports tab

Use the Reports tab of Business Process Choreographer Explorer to manage reports for specific processes and activities that were processed by Business Process Choreographer. You can 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. Use the predefined lists and charts for a drill-down approach to get state and event information for runtime entities. For example, lists, process and activity snapshot charts, and process and activity instances by period charts are available. The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

Available actions

The following actions are available in the navigation pane:

• Collapse and expand a group.

Click the arrow beside an item in the navigation pane to expand or 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 delete the report definition, click the **Delete** icon ( ).

  – To edit the report definition, click the **Edit** icon ( ).

  – To copy the report definition, click the **Copy** icon ( ).

  – To export the report result, click the **Export** icon ( ).

  – To run a report asynchronously, click the **Asynchronous Report** icon ( ).
    - After the asynchronous report completes successfully, the **Asynchronous Report Completed** icon ( ) is displayed in the navigation pane. Click the name of the report to view your results.
    - If the asynchronous report does not complete successfully, the **Asynchronous Report 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.

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.

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. The event collector application must be installed and running in order to use the reporting function.

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. In addition, you can extend the URL to go directly to the details of a process, task, or escalation.

```
http://app_server_host:port_no/context_root?oid_type=oid
```

For example:

```
http://hostname:9080/bpc?piid=PI:90030109.7232ed16.d33c67f6.beb30076
```

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.

oid_type
> Optional. The type of object that you want display. This parameter can take one of the following values:
>
> **aiid**   Activity instance ID
>
> **piid**   Process instance ID
>
> **ptid**   Process template ID
>
> **tkiid**   Task instance ID
>
> **tktid**   Task template ID
>
> **esiid**   Escalation instance ID

oid Optional. The value of the object ID.

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

Results

If you specified an object ID, the details page for the object is displayed. If you did not specify an object ID, the initial Business Process Choreographer Explorer page is displayed. By default, the initial page is 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. Furthermore, during configuration (or later) users can choose to add the reporting function to create reports on processes and activities and to retrieve statistical information on events.

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 system administrator roles, you can customize both the predefined views that are shown to your users and the default view that they see when they log in.

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 inquiries. 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 set of views, the default login view, and to define new views for 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**.

      After saving your changes, the predefined views appear with icons in front of them in the navigation pane, which allows you to move them up and down in the list.

      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. Define new 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 For ... And Define Customized Views page, where ... is the type of view, for example Process Templates, select a query table for your view.

A default query table is set for your view definition. You can either select a different query table, or choose not to use query tables in your view definition.

Note: If you use a query table, you cannot specify additional search criteria here for the view. All the search criteria must be defined in the query table definition.

If you are not using a query table, specify search criteria. Use the Process Criteria tab, the Task Criteria tab, and the Property Filters tab to limit the search results, for example, to a specific process template. When defining instance views, you can also use the User Roles tab to limit the search results to users, groups, or roles.

d. If you are using query tables and the query table definition has parameters, specify the query parameters that are needed on the Query Properties tab.

The parameter names that you specify must match the names in the query table definition. You can also provide default values for the parameters, and specify whether a default value can be overwritten when the query for the view is run.

e. Use the View Properties tab to select the list columns and list properties, such as ordering properties and the results threshold, to include in the view.

In addition, in View Settings, you can specify the actions to add to the action bar in the view. To select the actions to be included in the view or search that you are about to run:

- In Available Actions, select an action or actions, and click **Add**.
- To remove an action, select the action in Actions for View, and click **Remove**.
- The sequence of the actions in the action bar can be specified by moving the actions up and down in Actions for View.

If this is a task, process, or activity instance view, click **View Settings** to specify the items that are included in the view for system administrators and system monitors.

- 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**.

f. Enter a display name for the view in the **View Name** field, and click **Check and Save**.

The search is run to check for errors. If it runs without errors, the view is saved.

The new view appears in your navigation pane. Users see the new view when they next log into Business Process Choreographer Explorer. The views can be moved up or down in the navigation pane.

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 one of the system administrator roles.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Click **Define Views** in the taskbar.
2. In the Search and Define Customized Views page, select **Search For Process Templates And Define Customized Views**.
3. Click **Property Filters** → **Custom Property Filters**.
   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 **Check and Save**.
   The search is run to check for errors. If it runs without errors, the view is saved.

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 one of the system administrator roles.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Click **Define Views** in the taskbar.
2. In the Search and Define Customized Views page, select **Search For Process Instances And Define Customized Views**.
3. Click **Custom 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 **Check and Save**.

   The search is run to check for errors. If it runs without errors, the view is saved.

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. Independent of your roles, you can add your own
views to your navigation pane. For example, you can add a new view to monitor
the progress of a specific task or process. You can specify the information shown,
the filter and sort criteria, and also the actions provided in the view.

About this task

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

Procedure

1. In the section of the Views tab navigation pane, for example, Process Templates,

   where you want to define the new view, click the **New search** icon ( ).

2. In the Search For ... And Define Customized Views page, where ... is the type
   of view, for example Process Templates, select a query table for your view.

   A default query table is set for your view definition. You can either select a
   different query table, or choose not to use query tables in your view definition.

   Note: If you use a query table, you cannot specify additional search criteria
   here for the view. All the search criteria must be defined in the query table
   definition.

   If you are not using a query table, specify search criteria. Use the Process
   Criteria tab, the Task Criteria tab, and the Property Filters tab to limit the
   search results, for example, to a specific process template. When defining
   instance views, you can also use the User Roles tab to limit the search results to
   users, groups, or roles.

3. If you are using query tables and the query table definition has parameters,
   specify the query parameters that are needed on the Query Properties tab.

   The parameter names that you specify must match the names in the query
   table definition. You can also provide default values for the parameters, and
   specify whether a default value can be overwritten when the query for the
   view is run.

4. Use the View Properties tab to select the list columns and list properties, such
   as ordering properties and the results threshold, to include in the view.

   In addition, in View Settings, you can specify the actions to add to the action
   bar in the view. To select the actions to be included in the view or search that
   you are about to run:

   - In Available Actions, select an action or actions, and click **Add**.
   - To remove an action, select the action in Actions for View, and click **Remove**.
   - The sequence of the actions in the action bar can be specified by moving the
     actions up and down in Actions for View.

If this is a task, process, or activity instance view, click **View Settings** to specify the items that are included in the view for system administrators and system monitors. If you are a system administrator and or a 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**.

5. Enter a display name for the view in the **View Name** field, and click **Check and Save**.

The search is run to check for errors. If it runs without errors, the view is saved. Use the Summary tab to check the settings that are currently set for the view.

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 370
Business Process Choreographer Explorer is a standalone Web application that provides a set of administration functions for managing business processes and human tasks and for reporting on process and activity events. 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" on page 384

- "Footer"
- "Menu bar"
- "Login page"
- "Navigator" on page 385
- "Content panels" on page 385
- "Command bar" on page 385
- "Lists" on page 385
- "Details panel" on page 386
- "Message data" on page 386
- "Tabbed panes" on page 386
- "Search pages" on page 386
- "Error details" on page 387

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.

Style name	Description
.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.
.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.

Style name	Description
.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.

Style name	Description
.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.
.errorDetails	Tabbed pane with error details.
.errorDetailsStack	Tabbed pane with an exception stack.
.errorDetailsMessage	Text style for error message.

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. Use Business Process Choreographer Explorer to work with process instances and task instances and to report on business processes and human tasks.

Restricting process administration to system administrators

Switching off the default instance-based administration means that only users in the BPESystemAdministrator role can administer process instances, scope instances, and activity instances. Also, only users in the BPESystemMonitor role will be able to view or monitor process instances. This can improve performance because it reduces the database load and the rate that the database grows.

Before you begin

Before restricting administration and monitoring to the BPESystemAdministrator and BPESystemMonitor roles, make sure that these roles are mapped to a reasonable set of users for the `BPEContainer` application.

About this task

To improve performance, you can set a custom property that disables all administration tasks, including ones that were modeled. Only members of the appropriate administration and monitoring roles can administer processes, activities, and scopes.

Note: Changing this setting affects which user IDs can perform administrative actions on newly created process instances, activity instances, and scope instances. This might cause disruption to people or automated processes that perform administrative actions using user IDs that are not in the appropriate role.

Procedure

1. Using the administrative console, depending on whether Business Process Choreographer is configured on a cluster or on a server, perform one of the following:

   Cluster
   Click **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name*.

   Server Click **Servers** → **Server Types** → **WebSphere application servers** → *server_name*.

2. In the **Business Integration** section, expand **Business Process Choreographer** and click **Business Flow Manager** → **Custom Properties**.

3. Click **New** to add a new custom property.

4. Enter the name `ProcessAdministration` and the value `useSystemAdminAuthorizationOnly`.

   Note: Deleting this custom property will reverse the authorization restriction, but only for new process instances, activities, and scopes.

5. Save the changes.
6. Activate the changes by restarting the server of cluster where Business Process Choreographer is configured.

Results

Process and activity administration and monitoring are restricted to users in the BPESystemAdministrator and BPESystemMonitor roles. This change only affects new instances. Any existing instances that were created before the changes were made continue to be subject to the administration settings that were active at the time.

Related concepts

Alternate process administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

Related information

Java EE roles for business processes
Java EE roles are set up when Business Process Choreographer is configured. For Java EE role-based authorization, you must have a user registry configured and application security enabled.

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; for long-running processes other process life cycle actions, such as suspending, resuming, or terminating process instances; or repairing activities.

Business process administration—frequently asked questions

Answers to a set of frequently asked questions about administering business processes.

- "How is administering business process applications different when development mode is enabled?" on page 391
- "What happens if a process template is in the started state, but the application it belongs to is in the stopped state?" on page 391

- "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?" on page 392

How is administering business process applications different when development mode is enabled?

When development mode is enabled on a stand-alone server, it is possible to stop and uninstall business process applications even when the application has running instances of business processes or human tasks. In a production system, making sure that development mode is not enabled can protect your long-running instances from being accidentally stopped or uninstalled.

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.

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

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

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 SCA 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.

Stopping and starting process templates with administrative scripts

Administrative scripts provide an alternative to the administrative console for stopping and starting process and task templates. Use the administrative script to stop and start all templates that belong to an application.

Before you begin

Before you begin this procedure, the following conditions must be met:
- If WebSphere administrative security is enabled, and your user ID does not have operator or administrator authority, include the wsadmin -user and -password options to specify a user ID that has operator or administrator authority.

- The application server, on which the 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

The following steps describe how to use an administrative script to stop and start the process and task templates that belong to an application.

Procedure

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

   `Linux`   `UNIX`   On Linux and UNIX platforms, enter the following command:

   cd *install_root*/ProcessChoreographer/admin

   `Windows`   On Windows platforms, enter the following command:

   cd *install_root*\ProcessChoreographer\admin

2. To stop the process and task templates:

   `Windows`   On Windows systems, enter:

   install_root\bin\wsadmin -f bpcTemplates.jacl
                              -stop *application_name*

   `Linux`   `UNIX`   On UNIX and Linux systems, enter:

   install_root/bin/wsadmin -f bpcTemplates.jacl
                              -stop *application_name*

   Where:

   -stop *application_name*
       All templates that belong to the named application will be stopped.

   Existing instances of the process and task templates continue to run until they end normally. When the application stops, you cannot create new instances from the stopped templates.

3. To start the process and task templates:

   `Windows`   On Windows systems, enter:

   install_root\bin\wsadmin -f bpcTemplates.jacl
                              -start *application_name*

   `Linux`   `UNIX`   On UNIX and Linux systems, enter:

   install_root/bin/wsadmin -f bpcTemplates.jacl
                              -start *application_name*

   The templates belonging to the application start. You can use Business Process Choreographer Explorer to start process instances from the process templates.

Managing the process life cycle

After a process starts, it goes through various states until it ends. As a process administrator, you can take various actions on a process throughout its life cycle.

Starting a new process instance

You can start a new process instance from any of the process templates that you are authorized to use.

About this task

All of the installed and started process templates with the newest valid from date 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 **Currently Valid** under Process Templates in the Views tab 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 automatically in the Web browser. For long-running processes that are not automatically deleted after the process completes, a process output message is available on the process instance view. To see the output message, select the instance in a process list in Business Process Choreographer Explorer, and open 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 available.

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 Views tab 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 are color coded in the diagram, depending on their state. All states have an associated icon. For example, completed activities are indicated by a check mark. For more information, see the online help for the page.

3. To act on an activity, click the activity, and select **Show Activity Details**.

   Click an activity in the process state view to open a context menu. In this menu, you can display the activity details, skip the activity (mark an activity for skipping), or select it as the source for a jump to a different activity in the process. You can also repair switch activities that failed due to a problem with the evaluation of a case condition.

   The available actions are displayed. Select the action of your choice.

Viewing and modifying the variables of an activity

View and modify the activity variables of a process instance using Business Process Choreographer Explorer.

Before you begin

To see all of the variables of an activity, you need at least scope reader or process reader authority. To modify a variable you need scope administrator or process administrator authority.

About this task

You can access all variables that are visible to an activity and modify the variable values.

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Process Instance page. Do one of the following:
   - Click **View Process State**. Then click the relevant activity in the process state diagram, and click **Show Activity Variables**. You see the variables that are visible for the selected activity. Use the list to select a different activity in this process instance and display the visible variables.
   - Click **Activity Variables**. Use the list to select an activity in this process instance and to display the visible variables.
   - Click **Skip Activities**. Select an activity, and then click **Set Variables**. You see the variables that are visible for the selected activity. Use the list to select a different activity in this process instance and to display the visible variables.
2. Select a variable name to see the actual value.
3. Modify the value, and click **Save** to update the value settings of the variable.

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 the process instance.

Before you begin

To suspend and resume process instances, you must be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

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 Views tab 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

Alternate process administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

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 be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

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 Views tab 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.

Related concepts

Alternate process administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

Deleting process instances using Business Process Choreographer Explorer

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 be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

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 Views tab 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

Alternate process administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

Migrating process instances using Business Process Choreographer Explorer

Process instances can be migrated to the currently valid version of the process. You might want to do this, for example, when a newer version of the process becomes available.

Before you begin

To migrate process instances, you must be a process administrator or system administrator.

About this task

When you deploy a new version of a process, new process instances are based on this version of the process. However, existing process instances which are based on a former process template continue to run until they reach an end state. You can migrate existing process instances to the currently valid version of the process. The currently valid process template is the newest started version of a process, whose valid from date is not in the future.

You can also use the migrateProcessInstances.py script to migrate process instances and to migrate them in bulk. For more information, see the related information section.

In Business Process Choreographer Explorer, perform the following steps to migrate process instances.

Procedure

1. In the Views tab, click **All Versions** to see all installed process templates.
2. Select one or more process templates and click **Instances** to see the associated process instances.
3. Select the relevant process instance entries and click **Migrate** to migrate them to the currently valid version of the process. Do not select process instances that are based on the currently valid version of the process because they cannot be migrated.

Results

The selected process instances are migrated to the new process template. If an error occurs for an item, basic error information is shown and an error indicator with a link to more error details is also provided.

Managing work authorizations

After a process instance has started, you might need to manage work authorizations for the process, 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 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 process instance can have several work items associated with it because different people can have different roles.

Sometimes, you might need to change a process instance assignment after they are started, for example, to transfer a work item from the original owner to someone else. You might also need to create additional process work items or delete work items that are not needed anymore.

Creating process work items

Work items of a process instance are used to manage administrator or reader authorization for the process instance. You might want to create process work items for new administrators, for example, when a user claims the ownership of a process instance because the user ID of the starter of the process instance will be deleted. You might also want to create work items if a user needs to see or modify variable values. 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 process instance, you must be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action. The process instance must be in the running, failing, or terminating execution state. Work items are automatically propagated to all enclosed activity instances.

About this task

In Business Process Choreographer Explorer, complete the following steps to create a work item.

Procedure

1. Display the process instances that you administer.

   Click **Administered By Me** under Process Instances in the Views tab navigation pane.
2. Select the check box next to one or more process instances for which you want to create a work item, and click **Create Work Items**. The Create Process Work Items page is displayed.
3. Create the work items.
   a. In the **New Owner** field, enter the user ID for which the work item will be created.
   b. Select one or more roles from the **Reason** list.

      The reason `Administrator` gives the user administrator authority for the process instance, while `Reader` gives reader authority.
   c. Click **Create**.

Results

A work item is created for each role that you specify for the new work item owner.

Related tasks

You might want to delete process work items, for example, if you created process work items in error or if they are generated for someone who no longer works for the company.

"Transferring process work items if you are the administrator of the process"
You might need to change a process work item assignment. For example, you might want to transfer a process work item to another user because the ownership of the process instance needs to be claimed by another person. This might happen, for example, in a long-running process if the organization has changed since the process started.

You can transfer the ownership of a process instance by having a person with process administrator authorization take ownership of the process instance. You might want to do this, for example, in situations where the process starter is no longer with the company.

Transferring process work items if you are the administrator of the process

You might need to change a process work item assignment. For example, you might want to transfer a process work item to another user because the ownership of the process instance needs to be claimed by another person. This might happen, for example, in a long-running process if the organization has changed since the process started.

Before you begin

The following conditions apply when you transfer a process work item:

- You must be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.
- The process instance must be in the running, failing, or terminating execution state.
- A work item with an assignment reason `starter` cannot be transferred.
- When work items with the assignment reasons `administrator` or `reader` are transferred, the transfer is automatically propagated to all enclosed activity instances.

About this task

In Business Process Choreographer Explorer, complete the following steps to transfer a process work item.

Procedure

1. Display the process instances that you can administer.

   Click **Administered By Me** under Process Instances in the Views tab navigation pane.

2. Display the work items for a process instance.

   In the Process Instances Administered By Me page, select the check box next to one or more process instances, and click **Work Items**.

3. Transfer the work item.
   a. In the **New Owner / Group Name** field, enter either the user ID or the name of the group to which the work item will be transferred. Work items that are assigned to a user can only be transferred to another user, and work items that are assigned to a group can only be transferred to another group.
   b. Select one or more work items and click **Transfer**.

Results

The transferred process work item with the new work item owner appears in the list of work items.

Related concepts

Alternate process administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

Related tasks

"Creating process work items" on page 400
Work items of a process instance are used to manage administrator or reader authorization for the process instance. You might want to create process work items for new administrators, for example, when a user claims the ownership of a process instance because the user ID of the starter of the process instance will be deleted. You might also want to create work items if a user needs to see or modify variable values. This might happen, for example, in a long-running process if the organization has changed since the process started.

"Deleting process work items"
You might want to delete process work items, for example, if you created process work items in error or if they are generated for someone who no longer works for the company.

"Transferring the ownership of process instances" on page 412
You can transfer the ownership of a process instance by having a person with process administrator authorization take ownership of the process instance. You might want to do this, for example, in situations where the process starter is no longer with the company.

Deleting process work items

You might want to delete process work items, for example, if you created process work items in error or if they are generated for someone who no longer works for the company.

Before you begin

The following conditions apply when you delete a process work item:

- You must be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.
- The process instance must be in the running, failing, or terminating execution state.

- A work item with an assignment reason `starter` cannot be deleted.
- A work item with an assignment reason `administrator` can only be deleted when it is not the last work item with this assignment reason for the process instance.
- A work item for a specific user cannot be deleted when the work item is assigned to everybody.
- The deletion of a process reader or administrator work item is automatically propagated to all enclosed activities.

About this task

In Business Process Choreographer Explorer, complete the following steps to delete a process work item.

Procedure

1. Display the process instances that you administer.

   Click **Administered By Me** under Process Instances in the Views tab navigation pane.

2. Display the work items for a process instance.

   In the Process Instances Administered By Me page, select one or more process instances, and click **Work Items**.

3. Delete the work items.

   Select one or more work items and click **Delete**.

Results

The process work items are deleted.

Related concepts

Alternate process administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

Related tasks

"Creating process work items" on page 400
Work items of a process instance are used to manage administrator or reader authorization for the process instance. You might want to create process work items for new administrators, for example, when a user claims the ownership of a process instance because the user ID of the starter of the process instance will be deleted. You might also want to create work items if a user needs to see or modify variable values. This might happen, for example, in a long-running process if the organization has changed since the process started.

"Transferring process work items if you are the administrator of the process" on page 401
You might need to change a process work item assignment. For example, you might want to transfer a process work item to another user because the ownership of the process instance needs to be claimed by another person. This might happen, for example, in a long-running process if the organization has changed since the process started.

"Transferring the ownership of process instances" on page 412
You can transfer the ownership of a process instance by having a person with process administrator authorization take ownership of the process instance. You might want to do this, for example, in situations where the process starter is no longer with the company.

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.

The failure behavior of your process is controlled by the **Continue On Error** setting of the process template. If **Continue On Error** is set to no, any unexpected failure causes the affected activity to go to the stopped state.

If **Continue On Error** is set to yes (or if it is not set because your process was created with a WebSphere Integration Developer version earlier than 6.1.2) and an unexpected failure occurs, the default fault handler is invoked and ultimately causes the process to end in a failed state. The latter happens because with an unexpected failure there is no appropriate fault handler in the directly surrounding scope. When there is no explicit fault handler defined for the current fault and the default fault handler is invoked, it terminates the current scope and propagates the fault to the surrounding scope. Ultimately, this will cause the process to end in the failed state.

For invoke, Java snippet, human task, and custom activities you can model a dedicated **Continue On Error** setting and override the process setting. However, if

you leave the default value the same as for the process, you can repair failure
situations for these activity types. The setting at the activity level controls only the
behavior of faults that are generated by the implementation of the activity. Faults
that occur during the evaluation of the join condition, or during the evaluation of
the transition condition of outgoing links are still controlled by the setting at the
process level. Therefore, for example, an invoke activity can go to the stopped state
(if, for example, the evaluation of its join condition failed) even if the **Continue On
Error** setting at the activity level is set to yes.

If your activity stops, the process remains in the running state. You then have
several options in Business Process Choreographer Explorer to repair the process
and continue navigation.

Procedure
- To view process instances with activities in the stopped state, define your own
  process instance search. Or, click **Stopped Activities** under **Activity Instances** in
  the navigation pane, and then click the relevant process instance of the failed
  activity.
- 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.

Analyzing the cause of a failed process
Check information about an exception which caused a process to fail. If the process
is in the failed state, you cannot repair the instance itself but you might be able to
fix the cause of the problem to prevent the failure of future instances.

Before you begin

The process must be in the failed state.

About this task

Any exception that occurs during process navigation which is not one of the faults
defined for the process can lead to a failed process.

Complete the following steps in Business Process Choreographer Explorer.

Procedure
1. In the Views tab, navigate to the Process Instance page of the process.

   For example, define a new process instance search to look for processes in the
   failed state, and click **Details** for the details.
2. Select the **Error Details** tab to see more information about the failure of your
   process.
3. Repair the cause of the failure to avoid further failures of instances of this
   process template.

Modifying the variables of a stopped activity

Check the variables of an activity, and repair the process variables if they caused the activity to stop.

Before you begin

The process must be in the running state. To see the variables of an activity that are visible to it, you need at least scope reader or process reader authority. To modify a variable, you must be a scope administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

About this task

During the lifetime of a process, problems can occur due to incorrect or missing values in the variables that control the process behavior. You can access all variables that are visible to an activity and repair the process by modifying the variable values. After this, you can continue process navigation.

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Process Instance page.

   For example, on the Critical Processes page, click the name of a process instance. On the Process Instance page, click the **Activities** tab, and click the name of the stopped activity.

   Alternatively, to view stopped activities, click **Stopped Activities** under **Activity Instances** in the navigation pane. Then click the relevant activity.
2. Click the **Variables** button to get a list of all of the variables that are visible to the activity.
3. Select a single variable name to see the value.
4. Modify the value, and click **Save** to update the value settings of a single variable.

Restarting activities

You can restart an activity using new input data, for example, if you repaired the variables of an activity.

Before you begin

Generally, you can restart an activity if it is in the stopped state and the `stopReason` is STOP_REASON_IMPLEMENTATION_FAILED or STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED. Depending on the type of activity, you can restart the activity if it is in a state other than the stopped state. For more information on restarting activities in other states, see the related information on state transitions diagrams for activities.

About this task

To restart an activity, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, 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. Depending on the **Stop Reason** and the activity kind, you can specify the input data that is needed to start the activity again.

   You can, optionally, specify that the process **Continue on Error** setting is overridden for this activity. Deselect **Continue on Error** if you want the activity to stop again if an error occurs when the activity restarts.

3. If an expiration time is set for the activity, specify the expiration behavior for the restarted activity.

4. Click **Restart**.

Forcing the completion of activities

If you are aware that an activity is not going to complete in a timely manner, for example, because the invoked service is no longer available, you can force the completion of the activity so that the process flow can continue. You might also want to force the completion of an activity if you cannot repair the cause of the failure. For example, if the evaluation of a wait expression in a wait activity repeatedly causes the activity to stop, you might want to force the activity to complete.

Before you begin

Generally, you can force an activity to complete if it is in the stopped state and the `stopReason` is STOP_REASON_IMPLEMENTATION_FAILED, STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED, or STOP_REASON_EXIT_CONDITION_FALSE. Depending on the activity kind a force complete is also available in other states. For more information on forcing the completion of activities in other states, see the related information on state transitions diagrams for activities.

About this task

To force the completion of an activity, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Activity page for the activity, and click **Force Complete**.

2. Specify the data that is needed to complete the activity.

   You can only provide data for activities that have output variables, that is, invoke, human task, pick, and receive activities.

3. Click **Force Complete** again.

Rescheduling activities

You can reschedule activities using new date and time data in Business Process Choreographer Explorer.

Before you begin

Generally, you can reschedule pick, invoke, and staff activities if they are in the running, waiting, ready, or claimed state. Also, you can repair an activity that stopped because the timeout expression cannot be evaluated.

To reschedule an activity, you must be a system administrator or an administrator of the activity, an enclosing scope, or the process. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

About this task

To reschedule an activity, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Activity Instance List page. Select the activity you want to reschedule, and click **Reschedule**. Alternatively, click the name of the activity you want to reschedule and, on the Activity page, click **Reschedule**.
2. On the Reschedule Activity page, specify a date and time to reschedule the activity. Alternatively, you can specify that the activity is never rescheduled or that it is immediately rescheduled.
3. Then click **OK**.

Repairing stopped activities

Due to the dynamic quality of Business Process Choreographer Explorer, you can manually intervene in the process navigation. You can repair activities that stopped because problems occurred, for example, during an expression evaluation.

Before you begin

Use this procedure for any of the following situations:
- An activity stopped because the evaluation of a join condition failed. The **stopReason** is *Activation failed*.
- A switch activity stopped because the evaluation of a case condition failed. The **stopReason** is *Implementation failed*.
- A while or repeatUntil activity stopped because the evaluation of a loop condition failed. The **stopReason** is *Implementation failed*.
- A forEach activity stopped because the evaluation of a loop condition failed. The **stopReason** is *Implementation failed*.
- An activity stopped because the evaluation of a transition condition failed. The **stopReason** is *Follow-on navigation failed*.
- An activity stopped because the exit condition evaluated to `false` when evaluated on the completion of the activity. The **stopReason** is *Exit condition is false*.

About this task

Usually the administrator tries to force a retry of the activity or to force the completion of the activity. For activity failures that cannot be repaired with these actions, you can override the navigation of the activity using Business Process Choreographer Explorer. For details about the problem, click **Error Details** on the

Activity page. You might need to modify variable values before you continue with the repair action. Furthermore, you can jump from one activity of a process instance to another activity, which is described in the topic on jumping activities. You might want to use a skip activity option to mark a failing activity to be skipped in subsequent process instances. Also, you might want to skip the processing of a stopped activity or mark it to be skipped when it gets processed again.

To repair a stopped activity, complete the relevant step in Business Process Choreographer Explorer.

Procedure

1. To view stopped activities, click **Stopped Activities** under **Activity Instances** in the navigation pane, and then click the relevant activity.
2. You can now take the relevant action to repair the pending activity.
   - An activity stopped because the evaluation of a join condition failed. The **stopReason** is *Activation failed*. Do the following:
     a. In the Views tab, navigate to the Activity page for the activity, and click **Repair Join**.
     b. Select the relevant option to continue processing. You can specify that the join condition is reevaluated, and the navigation of the process instance continues. Alternatively, you can specify that the value of the join condition for the activity is set to true or false to determine if the navigation of the current branch continues.

        If a value of `True` is specified, the activity is started. If a value of `False` is specified, the behavior depends on the suppressJoinFailure process attribute. If this attribute is set to `yes`, the activity is skipped and the status of all outgoing links of this activity is set to `false`. Otherwise, the joinFailure standard fault of Business Process Execution Language is given.
     c. Click **Continue** to force the activity navigation.
   - A switch activity stopped because the evaluation of a case condition failed. The **stopReason** is *Implementation failed*. Do the following:
     a. In the Views tab, navigate to the Activity page for the activity, and click **Force Case Navigation**.
     b. Select the branch that is to be followed during the navigation. The branches are enumerated according to their position in the model. You can only select one branch.
     c. Click **Submit** to force the case navigation.
   - A while or repeatUntil activity stopped because the evaluation of a loop condition failed. The **stopReason** is *Implementation failed*. Do the following:
     a. In the Views tab, navigate to the Activity page for the activity, and click **Next Iteration** or **End Loop** to force the navigation of the activity.
   - A forEach activity stopped because the evaluation of a loop condition failed. The **stopReason** is *Implementation failed*. Do the following:
     a. In the Views tab, navigate to the Activity page for the activity, and click **Repair For Each**.
     b. Specify the relevant values to continue processing. Specify a value for the start counter and the final counter. If the forEach activity has an early exit condition, also specify a value for the number of iterations to be completed.
     c. Click **Continue** to force the activity navigation.

- An activity stopped because the evaluation of a transition condition failed. The **stopReason** is *Follow-on navigation failed*. Do the following:

    a. In the Views tab, navigate to the Activity page for the activity, and click **Force Navigation**.

    b. Select the names of the links that are to be followed during navigation. The displayed names of the links are the names that are determined in WebSphere Integration Developer during the modeling of the process. You can select an arbitrary number of links.

    c. Click **Submit** to force the activity navigation.

- An activity stopped because the exit condition evaluated to `false` when evaluated on the completion of the activity. The **stopReason** is *Exit condition is false*. Do the following:

    a. In the Views tab, navigate to the Activity page for the activity, and click **Restart** or **Force Complete**.

    b. Specify the data that is needed to restart or complete the activity.

    c. Click **Restart** or **Force Complete** to force the activity navigation.

Related concepts

"Activity jump targets" on page 414
When you jump from one activity of a process instance to another activity of the process instance using Business Process Choreographer Explorer, you can select the target activity from a list of possible target activities. This topic describes the restrictions that apply when you select an activity that serves as a target activity when you perform a jump action.

Related tasks

"Skipping activities" on page 415
You can skip an activity so that it is not included in the processing of the process instance.

"Repairing stopped activities using the process state view"
Using the process state view in Business Process Choreographer Explorer, you can manually repair activities that stopped.

Repairing stopped activities using the process state view

Using the process state view in Business Process Choreographer Explorer, you can manually repair activities that stopped.

Before you begin

You can use this procedure if the evaluation of a join condition, loop condition, case condition, transition condition, or a forEach counter value failed and the corresponding activity is stopped.

To perform this task, you must be a process administrator or system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

About this task

To repair a stopped activity using the process state view, complete the relevant step in Business Process Choreographer Explorer.

Procedure

1. To view the stopped activities of a process instance, click **Process Instances** →
   Critical Processes in the navigation pane. Then select the check box next to the
   relevant process instance, and click **View Process State**. One or more stopped
   activities or activity gateways might be listed. Alternatively, select **Activity
   Instances** → **Stopped Activities**.

2. Depending on the problem that occurred, you can now use repair actions to
   repair the activity.

   - An activity stopped because the evaluation of a join condition failed. The
     stopReason is *Activation failed*. Do the following:

     a. Click the activity gateway or, for a single incoming link, click the activity.

     b. Click **Repair Join** and choose to either reevaluate the condition or to
        force the result of the evaluation to true or false.

   - The evaluation of the loop condition failed for a While or Repeat Until
     activity. The **stopReason** is *Implementation failed*. Do the following:

     – Click the activity, and select either **Next Iteration** to continue the loop or
       End Loop and to end the loop.

   - The evaluation of a case condition failed for a switch activity. The
     stopReason is *Implementation failed*. Do the following:

     – Click the activity, and click **Force Case Execution**. Select the case to be
       executed.

   - An activity stopped because the evaluation of a transition condition failed.
     The **stopReason** is *Follow-on navigation failed*. Do the following:

     a. Click the activity gateway or, for a single outgoing link, click the link.

     b. Click **Repair Follow-on Navigation** . The target nodes and links of the
        available branches are highlighted.

     c. To select one or more branches, click the target nodes or links, and select
        Select this Branch.

     d. Then click a target node, link, or the source node, and click **Force
        Navigation** to force the navigation of the selected branches.

   - The evaluation of counter values failed for a forEach activity. The
     stopReason is *Implementation failed*. Do the following:

     a. Click the activity, and click **Repair For Each**.

     b. Enter start and final counter values and, optionally, the number of
        iterations to run to either continue or end the loop.

Related tasks

"Repairing stopped activities" on page 408
Due to the dynamic quality of Business Process Choreographer Explorer, you can
manually intervene in the process navigation. You can repair activities that stopped
because problems occurred, for example, during an expression evaluation.

Repairing correlation sets

You can view and modify the correlation sets for an activity using Business Process
Choreographer Explorer. Additionally, you can repair activity correlation sets that
are incorrect due to runtime faults or repair actions, or due to jump operations in
the process.

Before you begin

To repair the correlation sets that are associated with an activity, you must be a
process administrator or system administrator. However, if Business Flow Manager

is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

About this task

You might want to repair an activity correlation set if, for example, the activity is in the stopped state because the correlation set values do not match the expected values.

In Business Process Choreographer Explorer, perform the following steps.

Procedure

1. In the Views tab, navigate to the Activity page for the activity, and select the Correlation tab to see the correlation sets that are associated with the activity.
2. Click **Repair Correlation Sets** to modify the correlation sets.
3. If there is more than one correlation set defined, use the drop-down list to select the correlation set to modify.
   - If the correlation set is uninitialized, you can specify the values for the correlation set, and click **Initialize** to save the values.
   - If the correlation set is already initialized and you want to change the values, first click **Uninitialize** to remove the existing values. Then you can specify the new values for the correlation set, and click **Initialize** to save these values.

Results

All correlation sets contain the correct property values.

What to do next

If the activity is stopped, you can continue to repair the process by restarting the activity.

Transferring the ownership of process instances

You can transfer the ownership of a process instance by having a person with process administrator authorization take ownership of the process instance. You might want to do this, for example, in situations where the process starter is no longer with the company.

Before you begin

To transfer the ownership of a process instance, a process instance administrator or the system administrator for business processes claims ownership of the process instance. The process instance for which process ownership is claimed can be in any state.

You need administrative rights to claim the ownership of a process instance. A system or process administrator can assign these rights by creating or transferring a process work item with the `administrator` reason.

About this task

To claim the ownership of 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 Views tab navigation pane.

2. Claim ownership of the process.

   Select the check box next to the process instance or instances, and click **Claim Ownership**.

Results

You now have ownership of the process instance, are the process starter, and have process administrator rights for the process instance.

Related tasks

"Transferring process work items if you are the administrator of the process" on page 401
You might need to change a process work item assignment. For example, you might want to transfer a process work item to another user because the ownership of the process instance needs to be claimed by another person. This might happen, for example, in a long-running process if the organization has changed since the process started.

"Creating process work items" on page 400
Work items of a process instance are used to manage administrator or reader authorization for the process instance. You might want to create process work items for new administrators, for example, when a user claims the ownership of a process instance because the user ID of the starter of the process instance will be deleted. You might also want to create work items if a user needs to see or modify variable values. This might happen, for example, in a long-running process if the organization has changed since the process started.

"Deleting process work items" on page 402
You might want to delete process work items, for example, if you created process work items in error or if they are generated for someone who no longer works for the company.

Jumping activities

You can jump from one activity of a process instance to another activity of the process instance. You can select to complete the source activity before you jump to a target activity.

Before you begin

To perform this action, you must either be a system administrator or be a process or scope administrator of the scope or a parent scope to which the source and target activities belong. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action.

About this task

To jump from one activity to another, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Process State page of the process instance.
2. Click the relevant activity in the process state diagram.

   Note that jump actions are only available if the **Detail level** slider of the process state diagram is at the most detailed level.
3. To go to another activity, click **Jump to Another Activity**.

   This option only available for activities in the running state (for example, ready, claimed, running, stopped, or waiting).

   The process state diagram is displayed again, and only activities that qualify as target activities are selectable. For information on target activities, refer to the related information on activity jump targets.
4. Select a target activity to choose an action to perform.

   The actions that are available depend on the source activity.
5. Choose an action to perform.
   - To complete the source activity before you jump to the target activity, click **Complete Source Activity and Jump**.

     The **Complete Source Activity and Jump** option is only available for target activities if the source activity is a human task activity in the claimed state. This option completes the source activity before you jump to a target activity.
   - To force the completion of the source activity before you jump to the target activity, click **Force Complete Source Activity and Jump**. Then click **Force Complete and Jump** to complete the activity with the data that you provide.

     The **Force Complete Source Activity and Jump** option is available for target activities if the source activity is a human task activity in the ready, claimed, or stopped state. It is also available for an invoke activity in the running or the stopped state, a receive or a wait activity in the waiting or the stopped state, and all other basic activities in the stopped state. This option forces the completion of the source activity before you jump to a target activity.
   - To skip the activity and jump to another activity, click **Skip Source Activity and Jump**.
   - Click **Cancel Jump** to cancel the jump action.

Activity jump targets:

When you jump from one activity of a process instance to another activity of the process instance using Business Process Choreographer Explorer, you can select the target activity from a list of possible target activities. This topic describes the restrictions that apply when you select an activity that serves as a target activity when you perform a jump action.

During the navigation of a process instance, you can only jump from an activity to activities that are directly nested either in the same sequence or cyclic flow. Additionally, you can jump within a flow if the source and target activity are connected by a series of flow links and there are no other links connected to any of the activities in between.

- You can perform activity jumps within sequence activities. This means that the source activity and target activity of the jump must be in the same sequence and both are not nested in other structured activities.
- You can perform activity jumps within flow activities. In this case, the source activity and target activity of the jump can be directly nested in a flow activity and there must be only one path in the control flow from source to target.

- Additionally, you can jump outside of a scope if there is only one activity contained in the scope. For example, it is possible to jump from an invoke activity with an attached handler.
- You can also perform activity jumps within cyclic flows, whereby the source activity and target activity of the jump are also directly nested in the cyclic flow and are not nested in other structured activities.

Related tasks

"Repairing stopped activities" on page 408
Due to the dynamic quality of Business Process Choreographer Explorer, you can manually intervene in the process navigation. You can repair activities that stopped because problems occurred, for example, during an expression evaluation.

Skipping activities

You can skip an activity so that it is not included in the processing of the process instance.

About this task

To mark an activity to be skipped, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Process State page of the process instance.
2. Click the relevant activity in the process state diagram.

   Note that skip and jump actions are only available if the **Detail level** slider of the process state diagram is at the most detailed level.
3. Perform one of these skip actions.
   - Click **Skip Activity** to mark this activity to be skipped. The activity is then

     indicated by the skip requested icon . Activities that are skipped are

     indicated by the skipped icon .

     To perform this action, you must be a process administrator or a scope administrator of the scope or a parent scope to which the source and target activities belong.

     The **Skip Activity** action is available for any activity state. An activity that is in an end state is marked to be skipped but the state of the activity remains unchanged until it is reached again by the navigation. Therefore, if an activity is already in an end state, the activity is skipped as soon as it is activated again.
   - To unmark the activity to be skipped, click **Cancel Skip**. This cancels a previously selected skip activity request.
   - Alternatively, to skip the activity and jump to another activity, click **Jump to Another Activity**.

     The diagram is displayed again, and only activities that qualify as target activities are selectable. Note that the options available depend on the source activity.

     To skip the activity and jump to another activity, click **Skip Source Activity and Jump**.

Related tasks

"Repairing stopped activities" on page 408
Due to the dynamic quality of Business Process Choreographer Explorer, you can manually intervene in the process navigation. You can repair activities that stopped because problems occurred, for example, during an expression evaluation.

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. Undo actions are only necessary for activities that perform actions which do not participate in the transaction of the microflow.

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 Views tab 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.

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.

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 SCA module under **Additional Properties**, click **Human tasks**, and then select the task templates.

3. To stop the task templates, click **Stop**.

4. To start the task templates, click **Start**.

Stopping and starting task templates with the administrative scripts

Administrative scripts provide an alternative to the administrative console for stopping and starting task and process templates. Use the administrative scripts to stop and start all templates that belong to an application.

Before you begin

Before you begin this procedure, the following conditions must be met:

- If WebSphere administrative security is enabled, and your user ID does not have operator or administrator authority, include the wsadmin `-user` and `-password` options to specify a user ID that has operator or administrator authority.

- The application server, on which the 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

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 templates are put into the start state.

Procedure

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

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   `cd install_root/ProcessChoreographer/admin`

   `Windows` On Windows platforms, enter the following command:

   `cd install_root\ProcessChoreographer\admin`

2. To stop the task and process templates:

`Windows` On Windows systems, enter:

```
install_root\bin\wsadmin -f bpcTemplates.jacl
                      -stop application_name
```

`Linux` `UNIX` On UNIX and Linux systems, enter:

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                      -stop application_name
```

Where:

-stop *application_name*
    All templates that belong to the named application will be stopped.

Existing instances of the templates continue to run until they end normally.

3. Start the task and process templates.

`Windows` On Windows systems, enter:

```
install_root\bin\wsadmin -f bpcTemplates.jacl
                      -start application_name
```

`Linux` `UNIX` On UNIX and Linux systems, enter:

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                      -start application_name
```

The templates belonging to the application start. You can use Business Process Choreographer Explorer to work with task instances associated with the task template.

Creating and starting a task instance

You can create and start a task instance from any of the task templates that you are authorized to use.

About this task

All of the installed and started task templates with the newest valid from date 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 Views tab 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.

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.

   In the Views tab, 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.

Suspending and resuming task instances

You can suspend task instances using Business Process Choreographer Explorer. 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 Views tab 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**.

Restarting task instances

You can restart task instances using Business Process Choreographer Explorer. You might want to do this, for example, for a human task that is already running but not progressing as expected, or for a task that has reached an unexpected or undesired end state, such as failed or expired. Additionally, you can change the input message values of tasks before you restart them. You can restart a task that you want to reuse in order to initiate the same work again. This could be a human task that has finished, for example an invocation task or a collaboration task. Typically, you would restart this task with a changed input message.

Before you begin

The task instance can be a collaboration, invocation, or to-do task. The task instance can be in any state except inactive. Additionally:

- An invocation task cannot be in the running state.
- A to-do task cannot be in an end state, that is, it cannot be finished, failed, terminated, or expired. If the to-do task is forwarded, then the follow-on task cannot be in an end state.
- An inline to-do task cannot be in the ready state.

The task instance can be escalated, suspended, or waiting for subtasks. The caller must be the starter, originator, or an administrator of the task instance.

Restarting the task instance causes the people resolution to be newly performed and all timers to be reset. Any subtasks or follow-on tasks are deleted. Any escalations are cancelled and reset into the inactive state. For invocation tasks, the logged-on user becomes the starter of the restarted task instance.

About this task

To restart a task instance, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Task page for the task, and click **Restart**.

   For example, on the Task Instances Administered By Me page, select the check box for the task instance, and click **Restart**.

2. Click **Restart** to start the task again with the information you provide.

Rescheduling task instances

You can reschedule task instances using new date and time data in Business Process Choreographer Explorer.

Before you begin

You can update, reschedule, stop, and restart the expiration, deletion, and due times of a task. When a task is in the inactive state, after the task is created and before it is started, the task originator can update the reschedule time only to never reschedule or to immediately reschedule. This influences the scheduling of the time setting when the task is started. The task originator can modify the task duration after the task is created and before it is started.

When a task is in the ready, claimed, or running state, you can reschedule the due time and expiration time setting.

To change the due time setting of the task, you must be the owner, starter, originator, editor, or administrator of the task. To change the expiration time setting, you must be the originator or administrator of the task.

About this task

To reschedule an task, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. In the Views tab, navigate to the Task instance page for the task and click **Reschedule**.

   For example, on the Process Instances Administered By Me page, click the name of a process instance. On the Process Instance page, click the **Tasks** tab, and click the name of the task you want to reschedule. On the Task instance page, click **Reschedule**.

2. On the Reschedule Task page, select the time setting to modify, and specify a date and time for which to reschedule the task. Alternatively, you can specify that the task is never rescheduled or that it is immediately rescheduled.

3. Click **OK** to reschedule the task.

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 Views tab 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.

Managing work assignments

After a task or process instance has started, you might need to manage work assignments for the task or process, 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 or a process 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 or process instance assignment after they are started, for example, to transfer a work item from the original owner to someone else. You might want to 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.

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 Views tab navigation pane.
2. Select the check box next to the task that you want to transfer, and click **Transfer**.
3. Transfer the task.

a. In the **New Owner / Group Name** field, specify the user ID or the name of the group to which the work item will be transferred. You can transfer the task only to another potential owner of the task or the task administrator. Work items that are assigned to a user can only be transferred to another user, and work items that are assigned to a group can only be transferred to another group.

b. Select one or more work items and click **Transfer**.

Results

The transferred task appears on the list of tasks belonging to the new task owner.

Related tasks

"Specifying absence settings" on page 425
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 426
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.

"Creating task work items" on page 427
You might want to create task 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 task 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.

"Transferring task work items if you are the starter, originator, or administrator of the task"
You might need to change a task work assignment after work begins on the task. For example, you might want to transfer a task 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.

"Deleting task work items" on page 428
You might want to delete task work items, for example, if you created task work items in error or if task work items are generated for someone who no longer works for the company.

Transferring task work items if you are the starter, originator, or administrator of the task

You might need to change a task work assignment after work begins on the task. For example, you might want to transfer a task 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 task 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	Reason	Task state	Work items can be transferred to the following user roles:
Owner	Owner	Claimed	Potential owner, administrator.

Role	Reason	Task state	Work items can be transferred to the following user roles:
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 Views tab 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 one or more tasks, and click **Work Items**.

3. Transfer the work item.

   a. In the **New Owner / Group Name** field, enter the user ID or the name of the group to which the work item will be transferred. Work items that are assigned to a user can only be transferred to another user, and work items that are assigned to a group can only be transferred to another group.

   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 tasks

"Specifying absence settings for users" on page 426
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.

"Creating task work items" on page 427
You might want to create task 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 task 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.

"Transferring tasks that you own" on page 422
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.

"Deleting task work items" on page 428
You might want to delete task work items, for example, if you created task work items in error or if task work items are generated for someone who no longer works for the company.

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 people directory provider is required for substitution. You must also have substitution enabled for the Human Task Manager in Business Process Choreographer. The **My Substitutes** option is then visible in the taskbar.

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 template. Complete the following steps in Business Process Choreographer Explorer.

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 TaskSystemAdministrator 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 tasks

"Transferring tasks that you own" on page 422
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.

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.

Before you begin

You must have TaskSystemAdministrator rights to perform this task. Also, the Virtual Member Manager people directory provider is required for substitution. You must have substitution enabled for the Human Task Manager in Business Process Choreographer. The **Define Substitutes** option is then visible in the taskbar.

About this task

Complete the following steps in Business Process Choreographer Explorer.

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 tasks

"Transferring task work items if you are the starter, originator, or administrator of the task" on page 423
You might need to change a task work assignment after work begins on the task. For example, you might want to transfer a task 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 422
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.

Creating task work items

You might want to create task 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 task 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 Views tab navigation pane.

2. Select the check box next to one or more task instances for which you want to create a work item, and click **Create Work Items**. The Create Task 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.

Related tasks

"Transferring task work items if you are the starter, originator, or administrator of the task" on page 423
You might need to change a task work assignment after work begins on the task. For example, you might want to transfer a task 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 422
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.

"Deleting task work items"
You might want to delete task work items, for example, if you created task work items in error or if task work items are generated for someone who no longer works for the company.

Deleting task work items

You might want to delete task work items, for example, if you created task work items in error or if task 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 task work item.

Procedure

1. Display the task instances that you administer.

   Click **Administered By Me** under Task Instances in the Views tab navigation pane.

2. Display the work items for a task instance.

   In the Task Instances Administered By Me page, select one or more task instances, and click **Work Items**.

3. Delete the work items.

   Select one or more work items and click **Delete**.

Results

The task work items are deleted.

Related tasks

"Creating task work items" on page 427
You might want to create task 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 task 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.

"Transferring task work items if you are the starter, originator, or administrator of the task" on page 423
You might need to change a task work assignment after work begins on the task. For example, you might want to transfer a task 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 422
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.

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.

Complete the following steps in Business Process Choreographer Explorer.

Procedure

To view escalations, click **My Escalations** under Task Instances in the Views tab navigation pane.
- To view information about an escalation, click the escalation ID.
- To view information about an escalated task, click the task name.

Sending e-mails for escalations

When a task becomes overdue, it might result in an escalation. You can set up your system to send e-mails to designated people to inform them about the escalation.

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.

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 Views tab 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, events 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 Explorer, for example, to analyze process performance issues, 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 in the Reports tab of Business Process Choreographer Explorer. The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later. Also, the event collector application must be installed and configured.

Snapshot reports

Use snapshot reports in Business Process Choreographer Explorer 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 Explorer 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.
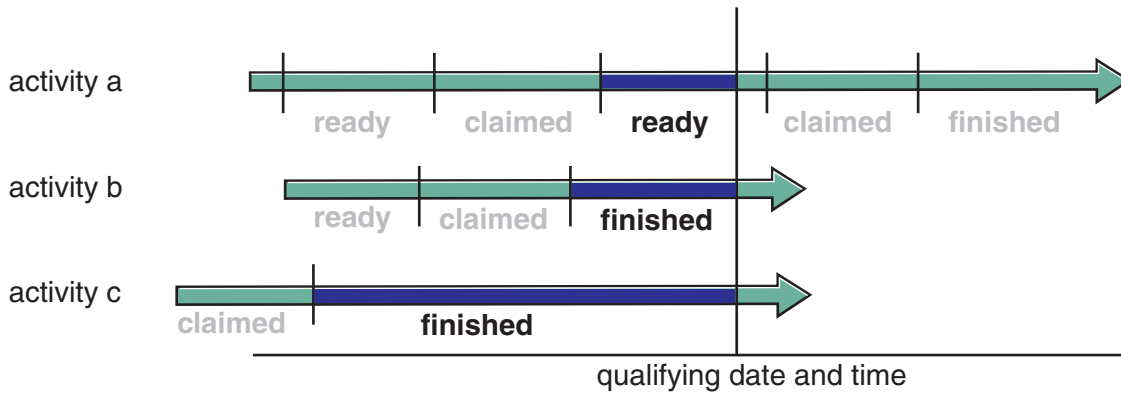
The snapshot includes one activity in the ready state (activity a) and two activities in the finished state (activities b and c).

Configuration parameter ReportAtSnapshotRange

If the reporting 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.
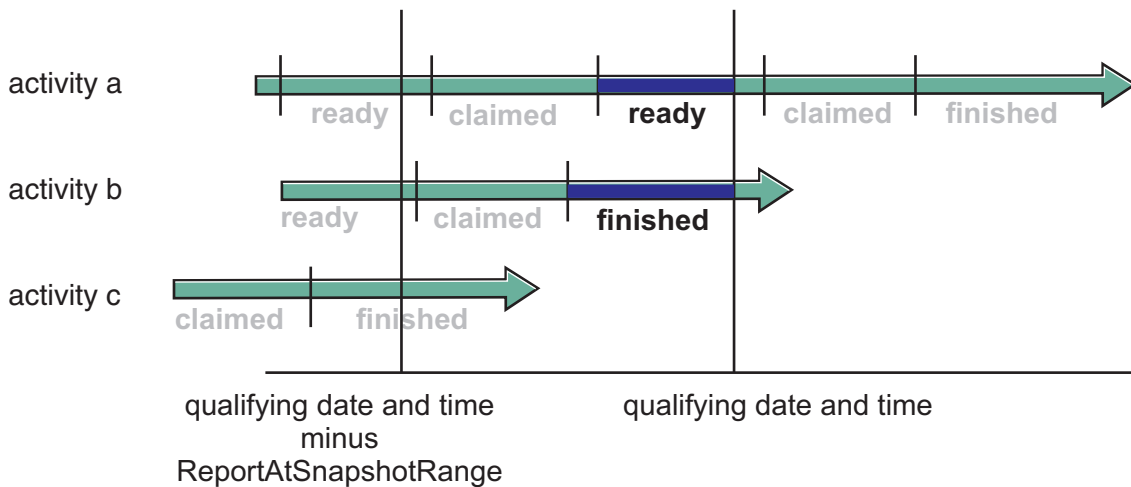
The snapshot includes one activity in the ready state (activity a) and one activity in the finished state (activity b). The report does not return the status of activity c.

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

"Using the predefined lists and charts" on page 435
Predefined lists and charts in Business Process Choreographer Explorer provide a drill-down approach to get you 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 436
Use predefined snapshot charts in Business Process Choreographer Explorer to see the distribution of process instance or activity instance states for a specified date and time.

"Example: Using the predefined charts" on page 439
This scenario gives you an example of how you could use the predefined charts in Business Process Choreographer Explorer.

"Creating user-defined snapshot reports" on page 445
You can define user-defined reports in Business Process Choreographer Explorer 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.

Period reports

Use period reports in Business Process Choreographer Explorer 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.
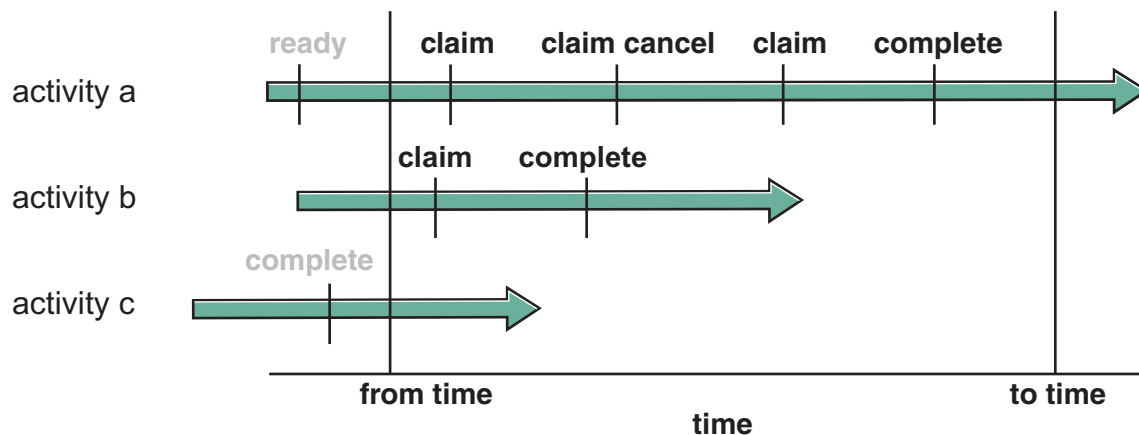
This means that if you query the number of completed events in this period, the result is two.

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

"Using the predefined lists and charts" on page 435
Predefined lists and charts in Business Process Choreographer Explorer provide a drill-down approach to get you 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 period chart" on page 437
Use the predefined period charts in Business Process Choreographer Explorer 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.

"Example: Using the predefined charts" on page 439
This scenario gives you an example of how you could use the predefined charts in Business Process Choreographer Explorer.

"Creating user-defined period reports" on page 447
You can create user-defined reports in Business Process Choreographer Explorer 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.

Time processing

In your report, consider the way that Business Process Choreographer Explorer 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

"Using the predefined lists and charts"
Predefined lists and charts in Business Process Choreographer Explorer provide a drill-down approach to get you 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.

Using the predefined lists and charts

Predefined lists and charts in Business Process Choreographer Explorer provide a drill-down approach to get you 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

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

"Snapshot reports" on page 431
Use snapshot reports in Business Process Choreographer Explorer to determine the states of activities or processes at a specific date and time.

"Period reports" on page 433
Use period reports in Business Process Choreographer Explorer to determine how often specific activity or process events occur over a period of time.

Related tasks

"Example: Using the predefined lists" on page 438
This scenario gives you an example of how you could use the predefined lists in Business Process Choreographer Explorer.

"Example: Using the predefined charts" on page 439
This scenario gives you an example of how you could use the predefined charts in Business Process Choreographer Explorer.

Creating a report using the predefined lists

Use the predefined lists in Business Process Choreographer Explorer 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.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Select a list type in the Reports tab navigation pane.

   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 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

Creating a predefined snapshot chart

Use predefined snapshot charts in Business Process Choreographer Explorer to see the distribution of process instance or activity instance states for a specified date and time.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Select the type of snapshot under **Charts** in the Reports tab navigation pane.

   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 431
Use snapshot reports in Business Process Choreographer Explorer to determine the
states of activities or processes at a specific date and time.

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer
processes timestamps and durations.

Related tasks

"Example: Using the predefined charts" on page 439
This scenario gives you an example of how you could use the predefined charts in
Business Process Choreographer Explorer.

Creating a predefined period chart

Use the predefined period charts in Business Process Choreographer Explorer 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

The Reports tab is visible only if reporting is configured. The reporting function
can be configured when you configure Business Process Choreographer Explorer,
however it can also be configured later.

About this task

For an example of a predefined period chart, use the predefined charts to see the
distribution of finished process instances over the last 12 months. To do this,
complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Select the type of period chart under **Charts** in the Reports tab navigation pane.

   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 433
Use period reports in Business Process Choreographer Explorer to determine how often specific activity or process events occur over a period of time.

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

Related tasks

"Example: Using the predefined charts" on page 439
This scenario gives you an example of how you could use the predefined charts in Business Process Choreographer Explorer.

Example: Using the predefined lists

This scenario gives you an example of how you could use the predefined lists in Business Process Choreographer Explorer.

About this task

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.

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. To do this, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Under **Lists** in the Reports tab navigation pane, 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 concepts

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

Related tasks

"Using the predefined lists and charts" on page 435
Predefined lists and charts in Business Process Choreographer Explorer provide a drill-down approach to get you 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.

Example: Using the predefined charts

This scenario gives you an example of how you could use the predefined charts in Business Process Choreographer Explorer.

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. To do this, complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Under **Charts** in the Reports tab navigation pane, 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 concepts

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

"Snapshot reports" on page 431
Use snapshot reports in Business Process Choreographer Explorer to determine the states of activities or processes at a specific date and time.

"Period reports" on page 433
Use period reports in Business Process Choreographer Explorer to determine how often specific activity or process events occur over a period of time.

Related tasks

"Using the predefined lists and charts" on page 435
Predefined lists and charts in Business Process Choreographer Explorer provide a drill-down approach to get you 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 436
Use predefined snapshot charts in Business Process Choreographer Explorer to see the distribution of process instance or activity instance states for a specified date and time.

"Creating a predefined period chart" on page 437
Use the predefined period charts in Business Process Choreographer Explorer 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 using Business Process Choreographer Explorer, 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 activities 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

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

"Snapshot reports" on page 431
Use snapshot reports in Business Process Choreographer Explorer to determine the states of activities or processes at a specific date and time.

"Period reports" on page 433
Use period reports in Business Process Choreographer Explorer to determine how often specific activity or process events occur over a period of time.

Related tasks

"Example: Using the user-defined reports" on page 450
This scenario gives you an example of how you could use the user-defined reports using Business Process Choreographer Explorer.

Attributes for Business Process Choreographer Explorer reports

Use attributes to define the content of your report in Business Process Choreographer Explorer, 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

Attribute	Description	Snapshot reports	Period reports
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
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 completed 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 Explorer

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 Explorer.

The following types of events can be caused by business process:

- "Process events"
- "Activity events"

Business Process Choreographer Explorer does not display business data sent in events.

Process events

The following table describes all process events that you can report on using Business Process Choreographer Explorer.

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
42009	Process terminating
42010	Process failing
42046	Process compensation failed
42079	Process migrated

In addition, the 21020 event code, process deletion time, is processed to update the process deleted attribute of process instances in the Business Process Choreographer Explorer reporting function. This event is cannot be queried in the same way as the process events listed.

Activity events

The following table describes all activity events that you can report on using Business Process Choreographer Explorer.

Code	Description
21006	Activity ready
21007	Activity started
21011	Activity completed
21021	Activity claim canceled

Code	Description
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
42063	Activity jumped
42065	Activity skipped on request
42070	Activity skipped on exit condition

Performance-relevant attributes

The time that is necessary to run a report definition using Business Process Choreographer Explorer 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 reporting 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.

Creating user-defined snapshot reports

You can define user-defined reports in Business Process Choreographer Explorer 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.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer. The report wizard guides you through the definition of the report.

Procedure

1. In the Reports tab 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 431
Use snapshot reports in Business Process Choreographer Explorer to determine the states of activities or processes at a specific date and time.

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

Related tasks

"Example: Using the user-defined reports" on page 450
This scenario gives you an example of how you could use the user-defined reports using Business Process Choreographer Explorer.

Related reference

"Attributes for Business Process Choreographer Explorer reports" on page 441
Use attributes to define the content of your report in Business Process Choreographer Explorer, and to filter the results. The attributes that are available depend on the report type.

"Business process events for Business Process Choreographer Explorer" on page 443
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 Explorer.

"Performance-relevant attributes" on page 444
The time that is necessary to run a report definition using Business Process Choreographer Explorer 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 in Business Process Choreographer Explorer 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.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer. The report wizard guides you through the definition of the report.

Procedure

1. In the Reports tab navigation pane, click the **New Report** icon (![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 (![icon]).

      - To delete an attribute, click the **Delete** icon (![icon]).

      - To change the position of an attribute in the report, click the **Up** icon (![icon]) or the **Down** icon (![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 433
Use period reports in Business Process Choreographer Explorer to determine how
often specific activity or process events occur over a period of time.

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer
processes timestamps and durations.

Related tasks

"Example: Using the user-defined reports"
This scenario gives you an example of how you could use the user-defined reports
using Business Process Choreographer Explorer.

Related reference

"Attributes for Business Process Choreographer Explorer reports" on page 441
Use attributes to define the content of your report in Business Process
Choreographer Explorer, and to filter the results. The attributes that are available
depend on the report type.

"Business process events for Business Process Choreographer Explorer" on page
443
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 Explorer.

"Performance-relevant attributes" on page 444
The time that is necessary to run a report definition using Business Process
Choreographer Explorer 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
using Business Process Choreographer Explorer.

About this task

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 their order complains about the long response
time they experienced. You want to know why this order took so long to process.

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. To do this,
complete the following steps in Business Process Choreographer Explorer.

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 Explorer cannot use 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. Under **Process Reports** in the Reports tab navigation pane, 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 Specify Report Content page, add **Process instance ID**, **Process work time**, **Process started** and **Process completed** to your report content.

e. On the Specify 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 Summary 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 **Back** 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 pane, 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 Specify Report Content page, add **Process instance ID**, **Activity name**, **Activity started** and **Activity completed** to your report content.

e. On the Specify 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 concepts

"Time processing" on page 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

"Snapshot reports" on page 431
Use snapshot reports in Business Process Choreographer Explorer to determine the states of activities or processes at a specific date and time.

"Period reports" on page 433
Use period reports in Business Process Choreographer Explorer to determine how often specific activity or process events occur over a period of time.

Related tasks

"Creating user-defined reports" on page 440
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 using Business Process Choreographer Explorer, and you can export the report results.

"Creating user-defined snapshot reports" on page 445
You can define user-defined reports in Business Process Choreographer Explorer 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.

"Creating user-defined period reports" on page 447
You can create user-defined reports in Business Process Choreographer Explorer 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.

Related reference

"Attributes for Business Process Choreographer Explorer reports" on page 441
Use attributes to define the content of your report in Business Process Choreographer Explorer, and to filter the results. The attributes that are available depend on the report type.

"Business process events for Business Process Choreographer Explorer" on page 443
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 Explorer.

"Performance-relevant attributes" on page 444
The time that is necessary to run a report definition using Business Process Choreographer Explorer 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.

Using saved user-defined report definitions

If you saved your report definitions in Business Process Choreographer Explorer, 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, using Business Process Choreographer Explorer. If your report contains parameters, you can set the values that you are interested in each time you run the report.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. To run a saved report definition, click the name of the report in the Reports tab 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 in Business Process Choreographer Explorer to continue working while the query is running.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. To run a saved report definition asynchronously, click the **Show pop-up menu** icon ( ) in the Reports tab navigation pane, and click the Asynchronous Search icon ( ).
2. 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 in Business Process Choreographer Explorer, you can export the report results for further external processing without running the report.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

This option is only available for saved user-defined report definitions that do not contain parameters. Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. To export the report results of a saved report definition, click the **Show pop-up menu** icon ( ) in the Reports tab navigation pane, 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/util/ bpcobserverexporter.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.

- █ Linux █   █ UNIX █   On Linux and UNIX platforms, enter:

  java -jar bpcobserverexporter.jar *options*

- █ Windows █   On Windows platforms, enter:

  java -jar bpcobserverexporter.jar *options*

You can specify options directly on the command line in the format -option value -option value ..., or specify the name of a properties file. In 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 29. 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 Explorer 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. Export with the export client only works for saved user-defined report definitions that do not contain parameters.

Editing and copying saved user-defined report definitions

You can change the settings of your saved report definitions in Business Process Choreographer Explorer, or use a copy of your report definition to create similar reports.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later.

About this task

Complete the following steps in Business Process Choreographer Explorer.

Procedure

1. Click the **Show pop-up menu** icon ( ) in the Reports tab navigation pane, 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 434
In your report, consider the way that Business Process Choreographer Explorer processes timestamps and durations.

Related reference

"Attributes for Business Process Choreographer Explorer reports" on page 441
Use attributes to define the content of your report in Business Process Choreographer Explorer, and to filter the results. The attributes that are available depend on the report type.

"Business process events for Business Process Choreographer Explorer" on page 443
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 Explorer.

"Performance-relevant attributes" on page 444
The time that is necessary to run a report definition using Business Process Choreographer Explorer 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 in Business Process Choreographer Explorer.

Before you begin

The Reports tab is visible only if reporting is configured. The reporting function can be configured when you configure Business Process Choreographer Explorer, however it can also be configured later. You cannot restore deleted report definitions.

About this task

Complete the following step in Business Process Choreographer Explorer.

Procedure

To delete a report definition, click the **Show pop-up menu** icon ( ) in the Reports tab navigation pane, and click the **Delete** icon ( ).

Results

The report name disappears from the navigation pane.

Part 4. Developing and deploying modules

459

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.

Comparison of the programming interfaces for interacting with business processes and human tasks

Enterprise JavaBeans (EJB), Web service, and Java Message Service (JMS), and Representational State Transfer Services (REST) generic programming interfaces are available for building client applications that interact with business processes and human tasks. Each of these interfaces has different characteristics.

The programming interface that you choose depends on several factors, including the functionality that your client application must provide, whether you have an existing end-user client infrastructure, whether you want to handle human workflows. To help you decide which interface to use, the following table compares the characteristics of the EJB, Web service, JMS, and REST programming interfaces.

	EJB interface	Web service interface	JMS message interface	REST interface
Functionality	This interface is available for both business processes and human tasks. Use this interface to build clients that work generically with processes and tasks.	This interface is available for both business processes and human tasks. Use this interface to build clients for a known set of processes and tasks.	This interface is available for business processes only. Use this interface to build messaging clients for a known set of processes.	This interface is available for both business processes and human tasks. Use this interface to build Web 2.0-style clients for a known set of processes and tasks.

	EJB interface	Web service interface	JMS message interface	REST interface
Data handling	Supports remote artifact loading of schemas for accessing business object metadata. If the EJB client application is running in the same cell as the WebSphere Process Server that it connects to, the schemas that are needed for the business objects of the processes and tasks do not have to be available on the client, they can be loaded from the server using the remote artifact loader (RAL). RAL can also be used cross-cell if the client application runs in a full WebSphere Process Server server installation. However, RAL cannot be used in a cross-cell setup where the client application runs in a WebSphere Process Server client installation.	Schema artifacts for input data, output data, and variables, must be available in an appropriate format on the client.	Schema artifacts for input data, output data, and variables, must be available in an appropriate format on the client.	Schema artifacts for input data, output data, and variables, must be available in an appropriate format on the client.
Client environment	A WebSphere Process Server installation or a WebSphere Process Server client installation.	Any runtime environment that supports Web service calls, including Microsoft .NET environments.	Any runtime environment that supports JMS clients, including SCA modules that use SCA JMS imports.	Any runtime environment that supports REST clients.
Security	Java Platform, Enterprise Edition (Java EE) security.	Web services security.	Java Platform, Enterprise Edition (Java EE) security for the WebSphere Process Server installation. You can also secure the queues where the JMS client application puts the API messages, for example, using WebSphere MQ security mechanisms.	Client application that call the REST methods must use an appropriate HTTP authentication mechanism.

An operation can be exposed by multiple protocols. Observe the following general considerations if you use the same operation in different protocols.

- In Web service and REST interfaces, all object identifiers, such as PIID, AIID, and TKIID are represented by a string type. Only the EJB API interface expects a type-safe object ID.
- Operation overloading is only used for EJB methods and not for WSDL operations. In some cases, multiple WSDL operations exist, in other cases, only one WSDL operation exists that allows all of the parameter variations either by omission (minOccurs="0"), or null values (nillable="true").
- In some EJB methods, XML namespaces and local names are passed as separate parameters. Most WSDL operations use the QName XML schema type to pass these parameters.
- Asynchronous interactions with long-running WSDL request-response operations, such as the callWithReplyContext operation in the EJB interface or the callAsync operation in the WSDL interface, are represented by the call operation in the JMS interface.

- The EJB interface returns a set of API objects, which expose getter and setter methods for the contained fields. Web service and REST interfaces return complex-typed (XML or JSON) documents to the client.
- Some Human Task Manager services operating on human tasks are also available as Business Flow Manager services operating on activities that call a human task.

Queries on business process and task data

Instance data for long-running business processes and human tasks are stored persistently in the database and are accessible by queries. Also, template data for business process templates and human task templates can be accessed using a query interface.

The EJB query interfaces, query API, and query table API, are available with Business Process Choreographer.

Depending on the clients that access process or task related data, one or more of the interfaces can be the right choice. REST and Web services APIs are available in Business Process Choreographer for querying task and process list data. However, for high volume process list and task list queries, use the Business Process Choreographer EJB query table API or REST query table API for performance reasons.

Comparison of the programming interfaces for retrieving process and task data

Business Process Choreographer provides a query table API and a query API for retrieving process and task data. Each of these interfaces has different characteristics.

The query interface that you choose depends on several factors, including the functionality that your client application must provide, whether you have an existing end-user client infrastructure, and performance considerations. To help you decide which interface to use, the following table compares the characteristics of the query table and the query programming interfaces.

Characteristic	query table API	query API
Availability	The query table API is available for the Business Flow Manager EJB interface and the REST programming interface.	The query API is available for EJB, Web service, JMS, and REST programming interfaces.
Methods for content retrieval	The API provides the following methods: • queryEntities • queryEntityCount • queryRows • queryRowCount	The API provides the following methods: • query • queryAll • queryProcessTemplates • queryTaskTemplates
Methods for meta data retrieval	The API provides the following methods: • getQueryTableMetaData • findQueryTableMetaData	The API provides the following methods: • QueryResultSet.getColumnType
Query table name	Specifies the query table on which the query table API is run. Only one query table can be queried at any one time. For example, `queryEntities("CUST.TASKS", ...)`.	The SELECT clause specifies the columns and predefined database views on which the query runs. This specification is similar to an SQL select clause. For example, `query("TASK.TKIID, TASK.STATE, WORK_ITEM.REASON", ...)`.

Characteristic	query table API	query API
SELECT clause and selected attributes	Use the filter options of the query table API to specify the attributes that the query is to return. Because the query is run against one query table, the attributes are uniquely identifiable by their names.	Use the SELECT clause to specify attributes. The syntax of the attribute name is: *view_name.attribute_name*. For example, to search for task states, specify TASK.STATE in your query.
WHERE clause and filters	Use the queryCondition property on the query table API to further filter the result of queries. Query tables provide pre-filtered content if primary query table filters, authorization filters, or query table filters have been specified on the query table definition.	Use the WHERE clause to filter the result of the query.
WHERE clause and selection criteria	The WHERE clause of the query API is not needed in this form on the query table API. Use the queryCondition property on the query table API for additional filtering. Selection criteria in the query table definition select a particular property of the attached query table. This is achieved in addition to the filtering by the WHERE clause on the query API.	Selection criteria are not available for the query API. However, selection criteria are similar to the part of the WHERE clause that defines, for example, the name or locale of QUERY_PROPERTY, or TASK_CPROP, or TASK_DESC. For example, a WHERE clause of `QUERY_PROPERTY.NAME='xyz'` is the same as specifying `NAME='xyz'` as a selection criterion on the query table definition for the QUERY_PROPERTY attached query table.
Work items and authorization	Use the WORK_ITEM query table to access work items. You can customize the use of work items on the query table definition when the query table is developed and on the query table API, using the AuthorizationOptions object or the AdminAuthorizationOptions object. For example, to exclude everybody work items when querying the TASK query table, specify a queryCondition property `WI.EVERYBODY=0` or specify `setUseEverbody(Boolean.FALSE)` on the AuthorizationOptions property.	Use the WORK_ITEM view to access work items. All four types of work items are considered for the query result: everybody, individual, groups, and inherited work items. To filter the work items for a specific type of work item, customize the WHERE clause. For example, to exclude the everybody work items, specify `WORK_ITEM.EVERYBODY=0`, in the WHERE clause.
Parameters	You can use parameters in filters and selection criteria for composite query tables.	Parameters are not available for the query API unless stored queries are used.
Stored queries and query tables	The difference between a stored query and a query table is that stored queries are defined for one particular query, while a query table is defined for a particular set of queries. For example, the query table definition does not allow the specification of an order-by clause because this information is typically available only when the query is run.	You can use stored queries to run query that contains a predefined set of options.
Materialized views	Materialized views are not available for the query table API.	Materialized views use database technologies to provide performance improvements for queries.
Custom tables	Supplemental query tables offer the same functionality as custom tables.	Custom tables are used to include data in queries that is external to the Business Process Choreographer database schema.
queryAll and authorization options	The queryAll functionality is provided by the AdminAuthorizationOptions object, which can be passed to the query table API instead of the AuthorizationOptions object. The caller must be in the BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor, or TaskSystemMonitor.	The queryAll method which can be used by users that have the BPESystemAdministrator Java EE role to return all of the objects in the query result without being restricted by work items for a particular user or group.
Internationalization	For attributes of query tables and for the query table, localized display names and descriptions are available when query tables are used.	Names of the columns of the selected views, as they appear in the database or as they are specified in the select clause, are returned.

Query tables in Business Process Choreographer

Query tables support task and process list queries on data that is contained in the Business Process Choreographer database schema. This includes human task data and business process data that is managed by Business Process Choreographer, and external business data. Query tables provide an abstraction on the data of Business Process Choreographer that can be used by client applications. In this way, client applications become independent of the actual implementation of the query table. Query table definitions are deployed on Business Process Choreographer containers, and are accessible using the query table API.

There are three types of query tables:
- Predefined query tables
- Supplemental query tables
- Composite query tables
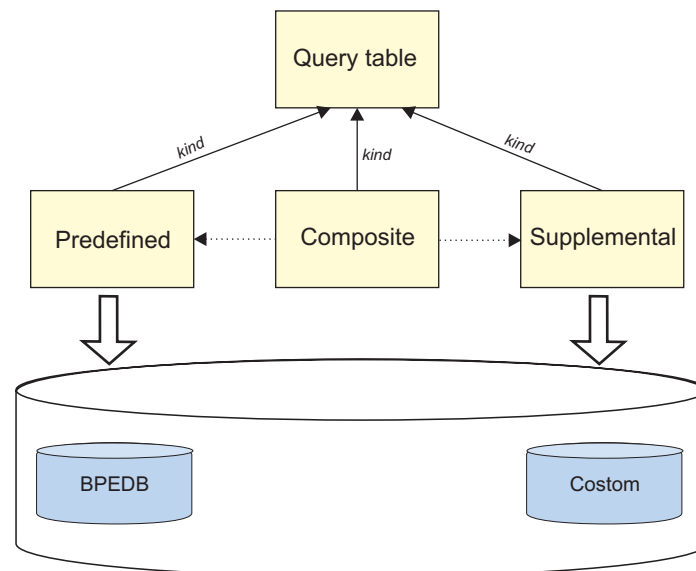
Figure 4. Query tables in Business Process Choreographer

Query tables are represented using similar models in the query table runtime, and you can use the query table API to query them. While predefined and supplemental query tables point directly to tables or views in the database, composite query tables compose parts of this data, which is represented in a single query table.

Query tables enhance the predefined database views and the existing query interfaces of Business Process Choreographer, and they:
- Are optimized for running process and task list queries, using performance optimized access patterns.
- Simplify and consolidate access to the information needed.
- Allow for the fine-grained configuration of authorization and filter options.

You can customize the query tables, for example, you can configure a query table so that it contains only those tasks or process instances that are relevant in a

particular scenario. It is also recommended that you use query tables where performance is important, such as with high volume process list and task list queries.

The Query Table Builder is provided as an Eclipse plug-in to:
- Develop composite and supplemental query tables
- Import and export query table definitions in XML format

You can download the Query Table Builder on the WebSphere Business Process Management SupportPacs site. Look for PA71 WebSphere Process Server - Query Table Builder. To access the link, see the related references section of this topic.

Predefined query tables

Predefined query tables provide access to the data in the Business Process Choreographer database. They are the query table representation of the corresponding predefined Business Process Choreographer database views, such as the TASK view or the PROCESS_INSTANCE view. These predefined query tables enhance the functionality and performance of the predefined database views because they are optimized for running process and task list queries.

The predefined query tables can be queried directly using the query table API. When you access the tables using the query table API, you are offered more options for configuration than when you use the query API.

Properties

Predefined query tables have the following properties:

Table 30. Properties of predefined query tables

Property	Description
Name	The query table name is the name of one of the predefined database views, in uppercase, for example, TASK.
Attributes	Attributes of predefined query tables define the pieces of information that are available for queries. These attributes are the names of columns, in uppercase, that are specified by the predefined database views.
	The attributes are defined with a name and a type. The type is one of the following:
	• **Boolean**: A boolean value
	• **Decimal**: A floating point number
	• **ID**: An object ID, such as TKIID of the TASK query table TASK
	• **Number**: An integer, short, or long
	• **String**: A string
	• **Timestamp**: A timestamp

Table 30. Properties of predefined query tables (continued)

Property	Description
Authorization	Predefined query tables use either instance-based or role-based authorization. • Predefined query tables with instance data require instance-based authorization. This means that only objects with a work item for the user who performs the query are returned. However, using the AdminAuthorizationOptions object, this verification can be reduced to a verification of the existence of a work item of any user. The user must have the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used for those queries. • Predefined query tables with template data require role-based authorization, which means that only users in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used, can access the contents of those query tables.

Predefined query tables with instance data

The following table shows the predefined query tables that contain instance data. These query tables:

• Can be used as the primary query of a composite query table.
• Use instance-based authorization if queried directly. This is accomplished with a join (SQL-) with the view that stores authorization information, that is, the predefined WORK_ITEM view or query table.
• Contain instance data, for example data of task instances or process instances.

Table 31. Predefined query tables containing instance data

Instance data	Query table name
Information about activities of a process instance.	ACTIVITY
	ACTIVITY_ATTRIBUTE
	ACTIVITY_SERVICE
Information about escalations belonging to human tasks.	ESCALATION
	ESCALATION_CPROP
	ESCALATION_DESC
Information about process instances.	PROCESS_ATTRIBUTE
	PROCESS_INSTANCE
	QUERY_PROPERTY
Information about human tasks.	TASK
	TASK_CPROP
	TASK_DESC

The WORK_ITEM query table also contains instance data, but this is not available as the primary query table or an attached query table. Work item information is available implicitly when querying query tables that use instance-based authorization. That is, attributes of the WORK_ITEM query table can be used when querying a query table with instance-based authorization, even though the

attributes are not explicitly specified by the query table.

Predefined query tables with template data

Predefined query tables with template data require role-based authorization. They can be queried only by administrators using the AdminAuthorizationOptions object.

The following table shows the predefined query tables that contain template data. These query tables:

- Can be used as the primary query table of a composite query table.
- Use role-based authorization if queried directly. This means that the caller using the API query method must be in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used, or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used, and AdminAuthorizationOptions must be used.
- Contain template data, for example, the template data of task templates or process templates.

Table 32. Predefined query tables containing template data

Template data	Query table name
Information about application components.	APPLICATION_COMP
Information about escalation templates.	ESC_TEMPL
	ESC_TEMPL_CPROP
	ESC_TEMPL_DESC
Information about process templates.	PROCESS_TEMPLATE
	PROCESS_TEMPL_ATTR
Information about task templates.	TASK_TEMPL
	TASK_TEMPL_CPROP
	TASK_TEMPL_DESC

"Supplemental query tables"
Supplemental query tables in Business Process Choreographer expose to the query table API business data that is not managed by Business Process Choreographer. With supplemental query tables, this external data can be used with data from the predefined query tables when retrieving business process instance information or human task information.

Composite query tables in Business Process Choreographer do not have a specific representation of data in the database; they comprise of a combination of data from related predefined and supplemental query tables. Use a composite query table to retrieve the information for a process instance list or task list, such as My To Dos.

Supplemental and composite query tables in Business Process Choreographer are developed during application development using the Query Table Builder. Predefined query tables cannot be developed or deployed. They are available when Business Process Choreographer is installed and provide a simple view on the artifacts in the Business Process Choreographer database schema.

Queries are run on query tables in Business Process Choreographer using the query table API, which is available on the Business Flow Manager EJB and the REST API.

You can use instance-based authorization, role-based authorization, or no authorization when you run queries on query tables.

Supplemental query tables

Supplemental query tables in Business Process Choreographer expose to the query table API business data that is not managed by Business Process Choreographer. With supplemental query tables, this external data can be used with data from the predefined query tables when retrieving business process instance information or human task information.

Supplemental query tables relate to database tables or database views in the Business Process Choreographer database. They are query tables that contain business data that is maintained by customer applications. Supplemental query tables provide information in a composite query table in addition to information that is contained in a predefined query table.

Supplemental query tables have the following properties:

Table 33. Properties of supplemental query tables

Property	Description
Name	The query table name must be unique in a Business Process Choreographer installation. When the query is run, this name is used to identify the query table that is queried. A query table is uniquely identified using its name, which is defined as *prefix.name*. The maximum length of *prefix.name* is 28 characters. The prefix must be different to the reserved prefix `'IBM'`, for example, `'COMPANY.BUS_DATA'`. Do not use a digit at the end of the table name. If a table is used multiple times within a query, the name of the table is extended with a number ranging from 0 to 9. For example, CUSTOM_VIEW0, CUSTOM_VIEW1 and so on. If there is already a digit at the end of your table name, Business Process Choreographer will remove this digit, which causes an QueryUnknownTableException.
Database name	The name of the related table or view in the database. Only uppercase letters may be used.
Database schema	The schema of the related table or view in the database. Only uppercase letters can be used. The database schema must be different to the database schema of the Business Process Choreographer database. Nevertheless, the table or view must be accessible with the same JDBC data source that is used to access the Business Process Choreographer database.
Attributes	Attributes of supplemental query tables define the pieces of information that are available for queries. These attributes must match the related name of the columns in the related database table or view. The attributes are defined with a name and a type. The name is defined in uppercase. The type is one of the following: • **Boolean**: A boolean value • **Decimal**: A floating point number • **ID**: An object ID of 16 bytes in length, such as TKIID of the TASK query table • **Number**: An integer, short, or long • **String**: A string • **Timestamp**: A timestamp
Join	Joins must be defined on supplemental query tables if they are attached in composite query tables. A join defines which attributes are used to correlate information in the supplemental query table with the information in the primary query table. When a join is defined, the source attribute and the target attribute must be of the same type.
Authorization	No authorization is specified for supplemental query tables, therefore, all authenticated users can see the contents.

Related concepts

"Predefined query tables" on page 468
Predefined query tables provide access to the data in the Business Process
Choreographer database. They are the query table representation of the
corresponding predefined Business Process Choreographer database views, such as
the TASK view or the PROCESS_INSTANCE view. These predefined query tables
enhance the functionality and performance of the predefined database views
because they are optimized for running process and task list queries.

"Composite query tables"
Composite query tables in Business Process Choreographer do not have a specific
representation of data in the database; they comprise of a combination of data from
related predefined and supplemental query tables. Use a composite query table to
retrieve the information for a process instance list or task list, such as My To Dos.

"Query table development" on page 480
Supplemental and composite query tables in Business Process Choreographer are
developed during application development using the Query Table Builder.
Predefined query tables cannot be developed or deployed. They are available when
Business Process Choreographer is installed and provide a simple view on the
artifacts in the Business Process Choreographer database schema.

"Query table queries" on page 499
Queries are run on query tables in Business Process Choreographer using the
query table API, which is available on the Business Flow Manager EJB and the
REST API.

"Authorization for query tables" on page 489
You can use instance-based authorization, role-based authorization, or no
authorization when you run queries on query tables.

Composite query tables

Composite query tables in Business Process Choreographer do not have a specific
representation of data in the database; they comprise of a combination of data from
related predefined and supplemental query tables. Use a composite query table to
retrieve the information for a process instance list or task list, such as My To Dos.

Composite query tables are designed by client developers and they allow for a
fine-grained configuration of filters and authorization options for optimized data
access when the query is run. They are realized with SQL, which is optimized for
task and process list queries.

It is recommended that you use composite query tables in production scenarios in
place of the standard Business Process Choreographer query APIs, because
composite query tables provide an abstraction over the actual implementation of
the query and thus enable query optimizations.

Furthermore, you can change composite query tables at runtime without
redeploying the client that accesses the query table.

The following figure provides an overview of the content of composite query
tables:
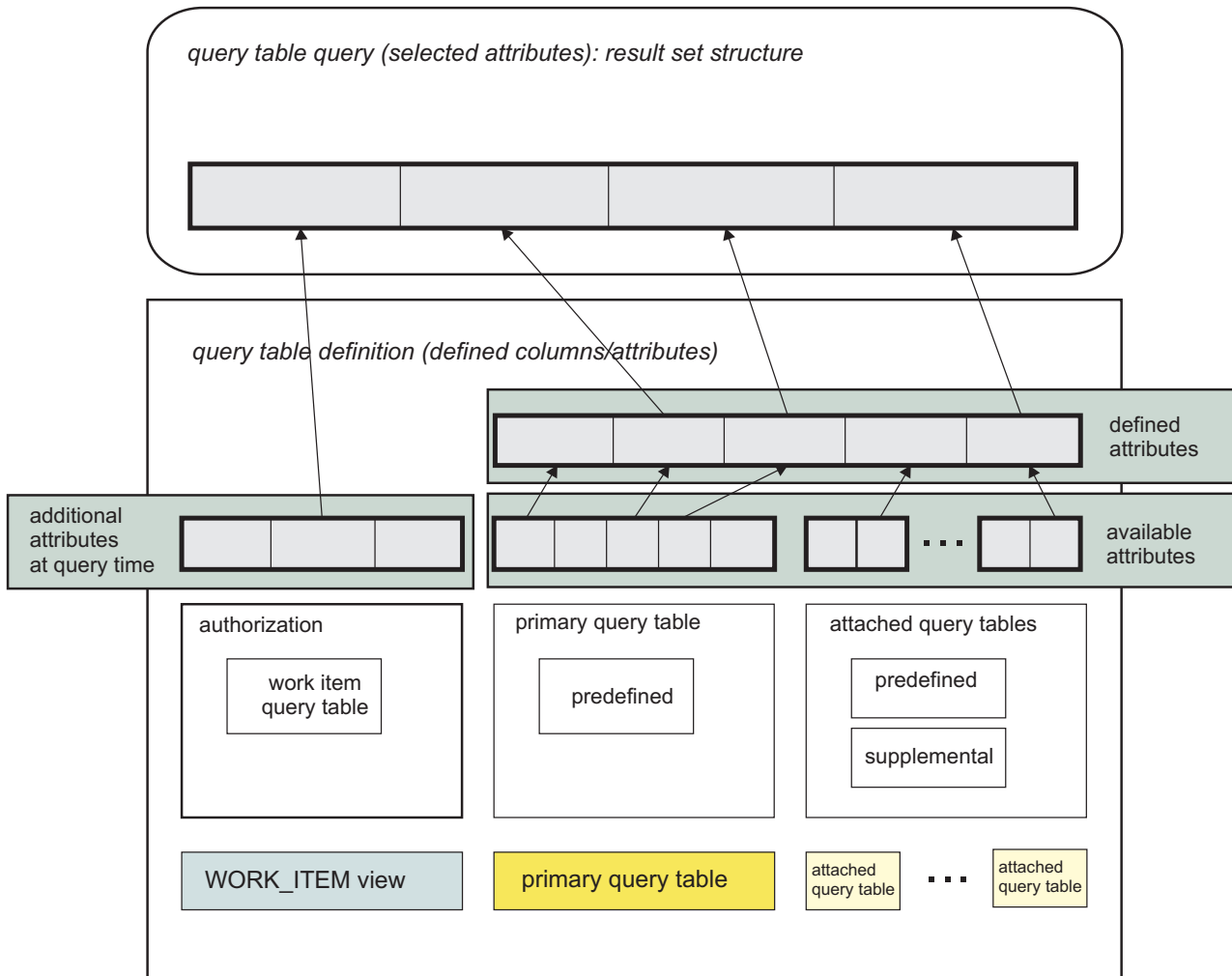
Figure 5. Composite query table content

All composite query tables are defined with one primary query table and zero or more attached query tables.

Primary query tables:
- Constitute the main information that is contained in a composite query table.
- Must be one of the predefined query tables.
- Uniquely identify each object in the composite query table by the primary key. For example, for the TASK predefined query table, this is the task ID TKIID.
- Authorize the contents of a query table using work items which are contained in the WORK_ITEM query table, if instance-based authorization is used.
- Determine the list of objects that are returned as rows of a table when querying the composite query table.

Attached query tables:
- Can be predefined query tables and supplemental query tables, which are already deployed on the system.
- Are available to provide information in addition to the information that is provided by the primary query table. For example, if TASK is the primary query

table, the description of the task provided in the TASK_DESC query table can be added to the contents of the composite query table.

Typically, the primary query table is chosen based on the purpose of the composite query table.
- If the composite query table describes a task list, the TASK query table is the primary query table.
- If the composite query table describes a process list, the PROCESS_INSTANCE query table is the primary query table.
- Lists of activities are retrieved using the ACTIVITY primary query table.
- Lists of human task escalations are retrieved using the ESCALATION primary query table.

The relationship between primary and attached query tables

The attached query table and the primary query table must have a one-to-one or one-to-zero relationship. If the one-to-one or one-to-zero relationship is violated, a runtime exception occurs when the query is run.

Primary query tables and attached query tables are correlated using a join attribute that is defined on the attached query table. This join attribute cannot be changed for predefined query tables, because it describes the relationship between the data in the various query tables of Business Process Choreographer. The join attribute is usually sufficient to maintain the one-to-one or one-to-zero relationship. For example, the CONTAINMENT_CTX_ID attribute is used on the TASK query table to attach the related process instance information that is identified by the PIID attribute on the PROCESS_INSTANCE query table.

When a one to many relationship exists, you must specify an additional criterion, known as selection criterion, in the Query Table Builder when you define the query table. For example, this could be `"LOCALE='en_US'"`. A task can have several descriptions that are identified using different locales for a single task.

Example 1:

The following figure provides a sample visualization of the selection criteria that is specified on attached query tables:
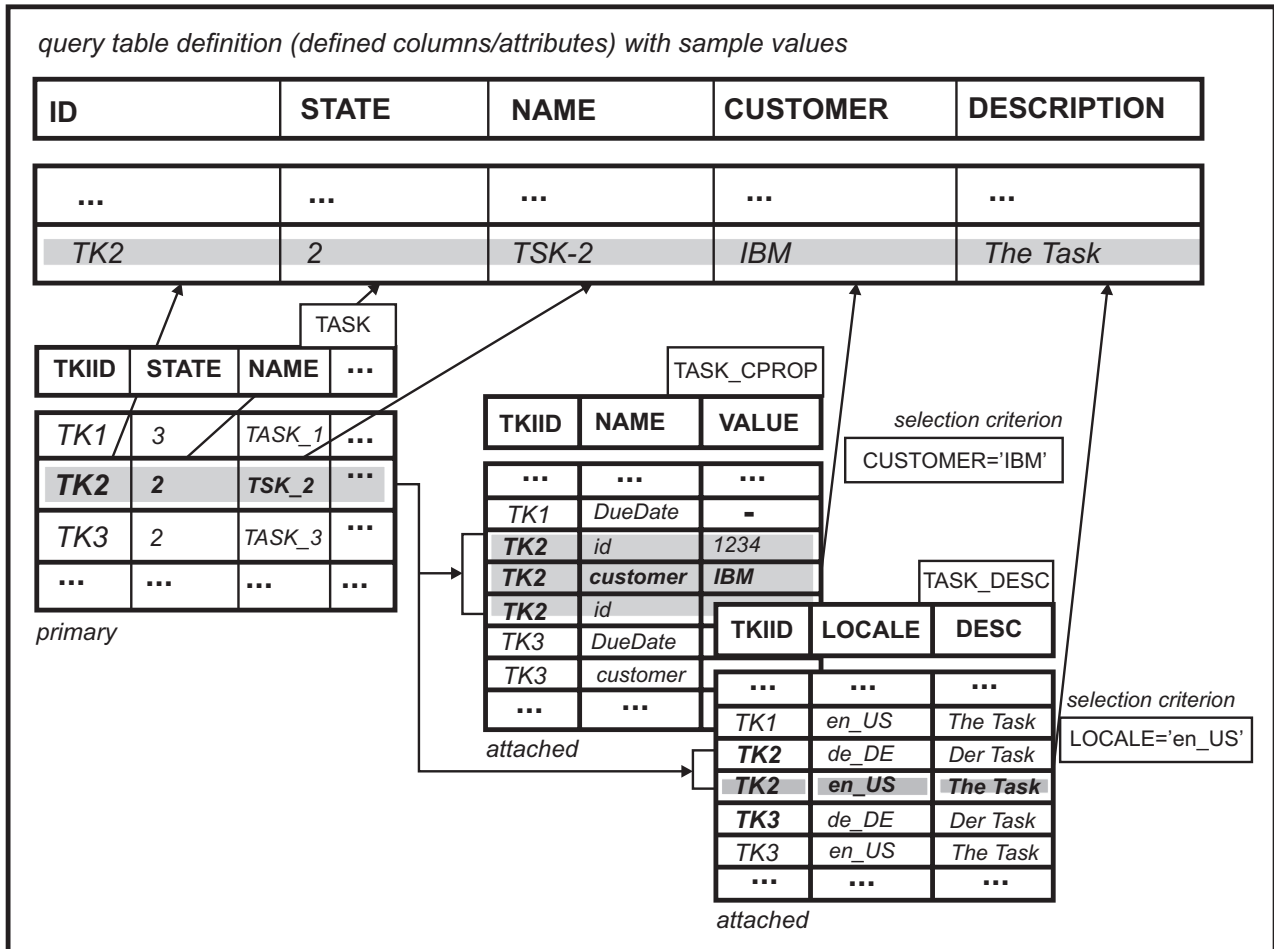
Figure 6. Composite query table with selection criteria

The composite query table contains the ID, STATE, NAME, CUSTOMER, and DESCRIPTION attributes.

- ID, STATE, and NAME are provided by the TASK primary query table.

- CUSTOMER is a custom property on TASK. Custom properties are stored in the TASK_CPROP query table. For a particular task, a custom property is uniquely identified using its name. This is reflected in the selection criterion `CUSTOMER='IBM'`.

- DESCRIPTION is the description of the task, which is stored in TASK_DESC query table. For each task instance, the task description for a particular task is uniquely identified by its locale. This is reflected in the selection criterion `LOCALE='en_US'`.

Example 2:

The focus of this example is on the relationship between the primary and the attached query tables, using TASK as the primary query table and TASK_DESC as the attached query table. When you define your composite query table, the LOCALE attribute of the TASK_DESC query table must be specified to ensure that there is a one-to-one or one-to-zero relationship between the primary query table and the attached query table. The table shows sample contents of a composite query table with a valid selection criterion for the TASK_DESC attached query table.

Table 34. Valid contents of a composite query table

TASK primary query table information	TASK_DESC attached query table information	
NAME	LOCALE	DESCRIPTION
task_one	en_US	This is a description.
task_two	en_US	This is a description.
...	...	...

The following table shows hypothetical invalid contents (in **bold** type) if the selection criterion is set incorrectly, which means that the one-to-one or one-to-zero relationship is violated.

Table 35. Invalid contents of a composite query table

Information from TASK (primary query table)	Information from TASK_DESC (attached query table)	
NAME	LOCALE	DESCRIPTION
task_one	en_US	This is a description.
task_one	**de_DE**	**Das ist eine Beschreibung.**
...	...	...

Properties

Composite query tables have the following properties:

Table 36. Properties of composite query tables

Property	Description
Name	The query table name must be unique within a Business Process Choreographer installation. When the query is run, this query table name is used to identify the query table that is queried. A query table is uniquely identified using its name, which is defined as *prefix.name* for composite query tables. The maximum length of the *prefix.name* is 28 characters. The prefix must be different from the reserved prefix 'IBM', for example, 'COMPANY.TODO_TASK_LIST'. Do not use a digit at the end of the table name. If a table is used multiple times within a query, the name of the table is extended with a number ranging from 0 to 9. For example, CUSTOM_VIEW0, CUSTOM_VIEW1 and so on. If there is already a digit at the end of your table name, Business Process Choreographer will remove this digit, which causes an QueryUnknownTableException.

Table 36. Properties of composite query tables (continued)

Property	Description
Attributes	Attributes of composite query tables define the pieces of information that are available for queries. The attributes are defined with a name, in uppercase. The type is inherited from the referenced attribute, which is one of the following: • **Boolean**: A boolean value • **Decimal**: A floating point number • **ID**: An object ID, such as TKIID of query table TASK • **Number**: An integer, short, or long • **String**: A string • **Timestamp**: A timestamp Attributes of composite query tables are defined using a reference to attributes of the primary query table or the attached query tables. The attributes of the composite query tables inherit the types and constants of referenced attributes. In addition to the attributes that are part of the query table definition, work item information can be queried at runtime. This is possible if the primary query table contains instance data, such as TASK or PROCESS_INSTANCE, and if instance-based authorization is used on the composite query table. For example, the query can be defined to return only human tasks of which the user is a potential owner.
Authorization	Each composite query table defines if instance-based, role-based, or no authorization is used when queries are run on it. If instance-based authorization is defined, only objects with a work item for the user who performs the query are returned. However, using AdminAuthorizationOptions this verification can be reduced to a verification of the existence of a work item of any user. The user must be in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used, for those queries, and AdminAuthorizationOptions must be passed to the query table API. If role-based authorization is defined, the user must be in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used, for those queries, and AdminAuthorizationOptions must be passed to the query table API. If no authorization is defined, the query is run without checks against the existence of work items of the related objects in the query table. All authenticated users can see the contents of the query table. Instance-based authorization can be defined if the primary query table contains instance data; role-based authorization can be defined if the primary query table contains template data. No authorization can be defined on composite query tables regardless of which primary query table is used.

Filters

Filters are used to limit the number of objects, or rows, that are contained in a composite query table.
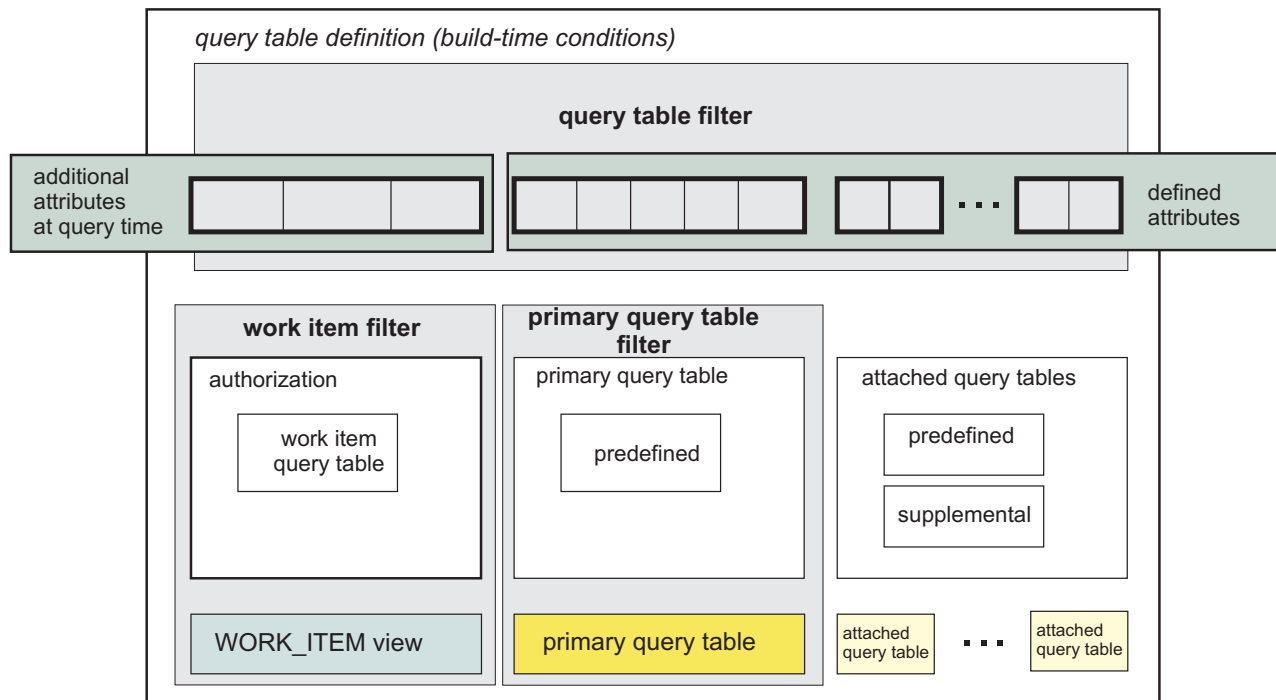
Figure 7. Filters in composite query tables

Filters in composite query tables can be defined during development on the:

- Primary query table, as the primary query table filter.
- Implicitly available WORK_ITEM query table which is responsible for authorization if the primary query table contains instance data. This filter is called the authorization filter, and is available only if the composite query table is configured to use instance-based authorization.
- Composite query table, as the query table filter.

Filters are defined during query table development. For example, a composite query table with the TASK primary query table can filter on tasks that are in the ready state ("STATE=STATE_READY" as the primary query table filter).

Authorization

Authorization for accessing the contents of a composite query table with a primary query table is similar to the authorization that is used to access the primary query table. The difference is that composite query tables can be configured to be more restrictive.

- If instance-based authorization is configured for use, the data contained in the composite query table is verified for existing work items in the WORK_ITEM query table. This verification is made against the primary query table. Everybody, individual, group, and inherited work items are used for the verification, depending on the configuration of the composite query table. If inherited work items are specified, objects that have a process instance as parent, such as a participating human task, with a related everybody, individual, or

group work item as configured, are contained in the composite query table. Typically, inherited work items are useful only for administrators.

- Composite query tables with a primary query table that contains template data must not be set to use instance-based authorization. If role-based authorization is used, queries can be run only by users that are in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used, and the AdminAuthorizationOptions object must be used.

Related concepts

"Predefined query tables" on page 468
Predefined query tables provide access to the data in the Business Process Choreographer database. They are the query table representation of the corresponding predefined Business Process Choreographer database views, such as the TASK view or the PROCESS_INSTANCE view. These predefined query tables enhance the functionality and performance of the predefined database views because they are optimized for running process and task list queries.

"Supplemental query tables" on page 471
Supplemental query tables in Business Process Choreographer expose to the query table API business data that is not managed by Business Process Choreographer. With supplemental query tables, this external data can be used with data from the predefined query tables when retrieving business process instance information or human task information.

Query table development

Supplemental and composite query tables in Business Process Choreographer are developed during application development using the Query Table Builder. Predefined query tables cannot be developed or deployed. They are available when Business Process Choreographer is installed and provide a simple view on the artifacts in the Business Process Choreographer database schema.

The Query Table Builder is available as an Eclipse plug-in and can be downloaded on the WebSphere Business Process Management SupportPacs site. Look for PA71 WebSphere Process Server - Query Table Builder. To access the link, see the related references section of this topic.

Query tables impact on the way applications are developed and deployed. The following steps describe the roles involved when you design and develop a Business Process Choreographer application that uses query tables.

Table 37. Query table development steps

Step	Who	Description
1. Analysis	Business analyst, client developer	Analyze which query tables are needed in the client application. Questions to be answered are: • How many task or process lists are provided to the user? Are there task or process lists that can share the same query table? • What kind of authorization is used? Instance-based authorization, role-based authorization, or none? • Are there other query tables already defined in the system that can be reused? • Must the query tables provide the content in multiple languages? If so, the selection criteria on attached query tables should be `LOCALE=$LOCALE`.
2. Query table development	Client developer, business analyst	Develop the query tables that are used in the client application. Try to specify the definition of the query tables such that the best performance is achieved with query table queries.
3. Query table deployment	Administrator	Query tables must be deployed to the runtime before they can be used. This step is done using the manageQueryTable.py wsadmin command.
4. Query table queries	Client developer	To run queries against query tables is the last step of query table development. The client developer must know the name of the query table and its attributes.

The following is sample code, which uses the query table API to query a query table. Examples 1 and 2 are provided to query the predefined query table TASK for simplicity reasons. Examples 3 and 4 query a composite query table, which is assumed to be deployed on the system. In application development, you should use composite query tables rather than directly querying the predefined query tables.

Example 1

```
// get the naming context and lookup the Business
// Flow Manager Enterprise JavaBeans home; note that the Business Flow
// Manager Enterprise JavaBeans home should be cached for performance
// reasons; also, it is assumed that there's an Enterprise JavaBeans
// reference to the local business flow manager Enterprise JavaBeans
Context ctx = new InitialContext();
LocalBusinessFlowManagerHome home =
 (LocalBusinessFlowManagerHome)
 ctx.lookup("java:comp/env/ejb/BFM");

// if the human task manager Enterprise JavaBeans is used, do:
// LocalHumanTaskManagerHome home =
//   (LocalHumanTaskManagerHome) ctx.lookup("java:comp/env/ejb/HTM");
// assuming that a EJB reference to the human task manager EJB
// has been defined

// create the business flow manager client-side stub
```

```
LocalBusinessFlowManager bfm = home.create();
// if the human task manager EJB is used, do:
// LocalHumanTaskManager htm = home.create();
// note that the human task manager Enterprise JavaBeans provides the
// same methods as the business flow manager Enterprise JavaBeans
// ************************************************
// ****************** example 1 ******************
// ************************************************

// execute a query against the TASK predefined query
// table; this relates to a simple My ToDo's task list
EntityResultSet ers = null;
ers = bfm.queryEntities("TASK", null, null, null);

// print the result to STDOUT
EntityInfo entityInfo = ers.getEntityInfo();
List attList = entityInfo.getAttributeInfo();
int attSize = attList.size();

Iterator iter = ers.getEntities().iterator();
while (iter.hasNext()) {
 System.out.print("Entity: ");
 Entity entity = (Entity) iter.next();
 for (int i = attSize - 1; i >= 0; i--) {
  AttributeInfo ai = (AttributeInfo) attList.get(i);
  System.out.print(
     entity.getAttributeValue(ai.getName()));
 }
 System.out.println();
}
```

Example 2

```
// ************************************************
// ****************** example 2 ******************
// ************************************************

// same example as example 1, but using the row-based
// query approach
RowResultSet rrs = null;
rrs = bfm.queryRows("TASK", null, null, null);

attList = rrs.getAttributeInfo();
attSize = attList.size();

// print the result to STDOUT
while (rrs.next()) {
 System.out.print("Row: ");
 for (int i = attSize - 1; i >= 0; i--) {
  AttributeInfo ai = (AttributeInfo) attList.get(i);
  System.out.print(
     rrs.getAttributeValue(ai.getName()));
 }
 System.out.println();
}
```

Example 3

```
// ************************************************
// ****************** example 3 ******************
// ************************************************

// execute a query against a composite query table
// that has been deployed on the system before;
// the name is assumed to be COMPANY.TASK_LIST
```

```
    ers = bfm.queryEntities(
      "COMPANY.TASK_LIST", null, null, null);
^
    //  print the result to STDOUT ...
```

Example 4

```
// **************************************************
// ******************* example 4 *******************
// **************************************************

// query against the same query table as in example 3,
// but with customized options
FilterOptions fo = new FilterOptions();

// return only objects which are in state ready
fo.setQueryCondition("STATE=STATE_READY");

// sort by the id of the object
fo.setSortAttributes("ID");

// limit the number of entities to 50
fo.setThreshold(50);

// only get a sub-set of the defined attributes
// on the query table
fo.setSelectedAttributes("ID, STATE, DESCRIPTION");

AuthorizationOptions ao = new AuthorizationOptions();

// do not return objects that everybody is allowed
// to see
ao.setEverybodyUsed(Boolean.FALSE);

ers = bfm.queryEntities(
  "COMPANY.TASK_LIST", fo, ao, null);

//  print the result to STDOUT ...
```

Related concepts

"Query table queries" on page 499
Queries are run on query tables in Business Process Choreographer using the query table API, which is available on the Business Flow Manager EJB and the REST API.

"Filters and selection criteria of query tables"
Filters and selection criteria are defined during query table development using the Query Table Builder, which uses a syntax similar to SQL WHERE clauses. Use these clearly defined filters and selection criteria to specify conditions that are based on attributes of query tables.

"Predefined query tables" on page 468
Predefined query tables provide access to the data in the Business Process Choreographer database. They are the query table representation of the corresponding predefined Business Process Choreographer database views, such as the TASK view or the PROCESS_INSTANCE view. These predefined query tables enhance the functionality and performance of the predefined database views because they are optimized for running process and task list queries.

"Supplemental query tables" on page 471
Supplemental query tables in Business Process Choreographer expose to the query table API business data that is not managed by Business Process Choreographer. With supplemental query tables, this external data can be used with data from the predefined query tables when retrieving business process instance information or human task information.

"Composite query tables" on page 473
Composite query tables in Business Process Choreographer do not have a specific representation of data in the database; they comprise of a combination of data from related predefined and supplemental query tables. Use a composite query table to retrieve the information for a process instance list or task list, such as My To Dos.

Filters and selection criteria of query tables

Filters and selection criteria are defined during query table development using the Query Table Builder, which uses a syntax similar to SQL WHERE clauses. Use these clearly defined filters and selection criteria to specify conditions that are based on attributes of query tables.

For information about installing the Query Table Builder, see the WebSphere Business Process Management SupportPacs site. Look for PA71 WebSphere Process Server - Query Table Builder. To access the link, see the related references section of this topic.

Attributes

Attributes used in an expression refer to query table attributes. Depending on the location of the expression, different attributes are available. For the client developer, query filters passed to the query table API are the only location where expressions can be used. For developers of composite query tables, various other locations exist where expressions can be used. The following table describes the attributes that are available at the different locations.

Table 38. Attributes for query table expressions

Where	Expression	Available attributes
Query table API	Query filter	• All attributes defined on the query table. • If instance-based authorization is used, all attributes defined on the WORK_ITEM query tables, prefixed with `'WI.'`. Examples: • `STATE=STATE_READY`, if the query table contains a STATE attribute and if a STATE_READY constant is defined for this attribute
Composite query table	Query table filter	• `STATE=STATE_READY AND WI.REASON=REASON_POTENTIAL_OWNER`, if the query table contains a STATE attribute and the query table uses instance-based authorization
	Primary query table filter	• All attributes defined for the primary query table. Example: • `STATE=STATE_READY`, if the query table contains a STATE attribute and a STATE_READY constant is defined for this attribute
	Authorization filter	• All attributes defined on the WORK_ITEM predefined query table, prefixed with `'WI.'`. Example: • `WI.REASON=REASON_POTENTIAL_OWNER`
	Selection criterion	• All attributes defined on the related attached query table. Example: • `LOCALE='en_US'`, if the attached query table contains a LOCALE attribute, such as the TASK_DESC query table

The following figure shows the various locations of filters and selection criteria in expressions, and includes examples:
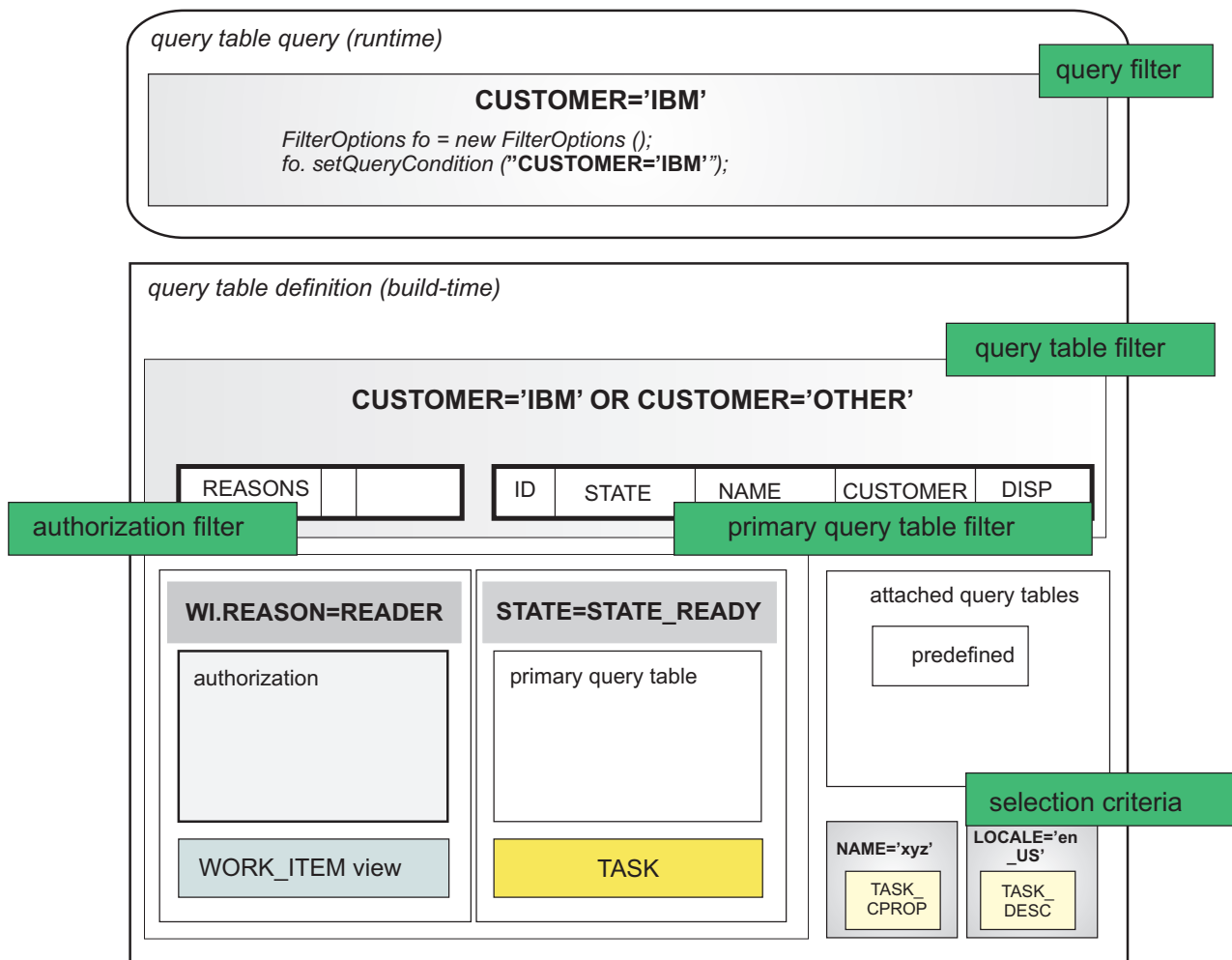
Figure 8. Filters and selection criteria in expressions

Expressions

Expressions have the following syntax:

```
expression  ::=  attribute binary_op value |
                 attribute unary_op |
                 attribute list_op list |
                 (expression) |
                 expression AND expression |
                 expression> OR expression
```

The following rules apply:

- AND takes precedence over OR. Subexpressions are connected using AND and OR.
- Brackets can be used to group expressions and must be balanced.

Examples:

- `STATE = STATE_READY`
- `NAME IS NOT NULL`
- `STATE IN (2, 5, STATE_FINISHED)`
- `((PRIORITY=1) OR (WI.REASON=2)) AND (STATE=2)`

An expression is executed in a certain scope which determines the attributes that are valid for the expression. Selection criteria, or query filters, are run in the scope of the query table on which the query is run.

The following example is for a query that is run on the predefined TASK query table:
```
'(STATE=STATE_READY AND WI.REASON=REASON_POTENTIAL_OWNER)
OR (WI.REASON=REASON_OWNER)'
```

Binary operators

The following binary operators are available:

binary_op ::= = | < | > | <> | <= | >= | LIKE | NOT LIKE

The following rules apply:
- The left-side operand of a binary operator must reference an attribute of a query table.
- The right-side operand of a binary operator must be a literal value, constant value, or parameter.
- The `LIKE` and `NOT LIKE` operators are only valid for attributes of attribute type `STRING`.
- The left-side operand and the right-side operand must be of compatible attribute types.
- User parameters must be compatible to the attribute type of the left-side attribute.

Examples:
- `STATE > 2`
- `NAME LIKE 'start%'`
- `STATE <> PARAM(theState)`

Unary operators

The following unary operators are available:

unary_op ::= IS NULL | IS NOT NULL

The following rules apply:
- The left-side operand of a unary operator must reference an attribute of a query table. Valid attributes depend on the location of the filter or selection criterion.
- All attributes can be checked for null values, for example: `CUSTOMER IS NOT NULL`.

Example:
```
DESCRIPTION IS NOT NULL
```

List operators

The following list operators are available:

list_op ::= IN | NOT IN

The following rules apply:
- The right-side of a list operator must not be replaced by a user parameter.

- User parameters can be used within the list on the right-side operand.

Example:
```
STATE IN (STATE_READY, STATE_RUNNING, PARAM(st), 1)
```

Lists are represented as follows:
list ::= *value* [, *list*]

The following rules apply:
- The right-side of a list operator must not be replaced by a user parameter.
- User parameters can be used within the list on the right-side operand.

Examples:
- `(2, 5, 8)`
- `(STATE_READY, STATE_CLAIMED)`

Values

In expressions, a value is one of the following:
- **Constant**: A constant value, which is defined for the attribute of a predefined query table. For example, `STATE_READY` is defined for the STATE attribute of the TASK query table.
- **Literal**: Any hardcoded value.
- **Parameter**: A parameter is replaced when the query is run with a specific value.

Constants are available for some attributes of predefined query tables. For information about constants that are available on attributes of predefined query tables, refer to the information about predefined views. Only constants that define integer values are exposed with query tables. Also, instead of constants, related literal values, or parameters can be used.

Examples:
- `STATE_READY` on the STATE attribute of the TASK query table can be used in a filter to check whether the task is in the ready state.
- `REASON_POTENTIAL_OWNER` on the REASON attribute of the WORK_ITEM query table can be used in a filter to check whether the user who runs the query against a query table is a potential owner.
- Query filter `STATE=STATE_READY` is the same as `STATE=2`, if the query is run on the TASK query table.

Literals can also be used in expressions. A special syntax must be used for timestamps and for IDs.

Examples:
- `STATE=1`
- `NAME='theName'`
- `CREATED > TS ('2008-11-26 T12:00:00')`
- `TKTID=ID('_TKT:801a011e.9d57c52.ab886df6.1fcc0000')`

Parameters in expressions allow for a dynamicity of composite query tables. There are user parameters and system parameters:

- User parameters are specified using PARAM (*name*). This parameter must be provided when the query is run. It is passed as an instance of the com.ibm.bpe.api.Parameter class into the query table API.
- System parameters are parameters that are provided by the query table runtime, without being specified when the query is run. The system parameters $USER and $LOCALE are available.
  - $USER, which is a string, contains the value of the user who runs the query.
  - $LOCALE, which is a string, contains the value of the locale that is used when the query is run. An example for the value of $LOCALE is 'en_US'.

You can specify a parameter in the selection criteria of an attached query table which selects on a specific locale. For example, if the primary query table is TASK in a composite query table and an attached query table is TASK_DESC. The following are examples of parameters:
- STATE=PARAM(theState)
- LOCALE=$LOCALE
- OWNER=$USER

Related concepts

"Query table development" on page 480
Supplemental and composite query tables in Business Process Choreographer are developed during application development using the Query Table Builder. Predefined query tables cannot be developed or deployed. They are available when Business Process Choreographer is installed and provide a simple view on the artifacts in the Business Process Choreographer database schema.

"Query table queries" on page 499
Queries are run on query tables in Business Process Choreographer using the query table API, which is available on the Business Flow Manager EJB and the REST API.

Related tasks

"Creating query tables for Business Process Choreographer Explorer" on page 518
You can use query tables instead of the EJB query API to improve the performance of Business Process Choreographer Explorer. To create the query tables, use the Query Table Builder.

Authorization for query tables

You can use instance-based authorization, role-based authorization, or no authorization when you run queries on query tables.

The authorization type is defined on the query table.
- Instance-based authorization indicates that objects in the query table are authorized using a work item. This is done by verifying if a suitable work item exists.
- Role-based authorization is based on Java EE roles. It indicates that the caller using the API query method must be in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used to see the contents of the query table. It is available for predefined query tables with template data and for composite query tables with a primary query table that contains template data. Objects in those query tables do not have related work items.
- When no authorization is specified, all authenticated users can see all contents of the query table, after filters are applied.

The type of authorization on predefined query tables and the type of authorization that can be configured on composite and supplemental query tables is outlined in the following table.

Table 39. Types of authorization for query tables

Query table	Instance-based authorization	Role-based authorization	No authorization
Predefined	Required for predefined query tables with instance data.	Required for predefined query tables with template data.	N/A
Composite	Can be turned off which means that no authorization is used and the security constraints are overridden. That is, every authenticated user can use the query table to retrieve data, independently of whether they are authorized for the respective objects. Composite query tables with a primary query table that contains template data must not be set to use instance-based authorization.	Can be turned off, for example for composite query tables with a primary query table that contains template data. This means that no authorization is used and the security constraints are overridden. That is, every authenticated user can use the query table to retrieve data, independently of whether they are authorized for the respective objects. Composite query tables with a primary query table that contains instance data must not be set to use role-based authorization.	All authenticated users can see all contents of the query table, after filters are applied.
Supplemental	Supplemental query tables must not be set to use instance-based authorization because they are not managed by Business Process Choreographer, and therefore it has no authorization information for the contents of these tables.	Supplemental query tables must not be set to use role-based authorization.	All authenticated users can see all contents of the query table, after filters are applied.

The following figure provides an overview of the available options for the authorization types, depending on the type of query table. Also, it outlines the different behaviors and the query table API and its authorization options.

Authorization	Instance-based authorization	None	Role-based authorization
Composite query table	primary query table with instance data	all	primary query table with template data
Predefined query tables	instance data	*n/a*	template data
Supplemental query tables	*n/a*	business data	*n/a*
Query with AuthorizationOptions	**(A)** Query result contains objects with work items related to the caller.	**(C)** Query result contains all objects that are in this query table.	*n/a*
Query with AdminAuthorization Options*	**(B)** Query result contains all objects that are in this query table.		**(D)** Query result contains all objects that are in this query table.

Figure 9. Instance-based authorization for query tables

*) If the onBehalfUser is set, (A) applies

Instance-based authorization for objects in the query result using work items depend on the authorization parameter that is passed to the query table API and on the setting of the instance-based authorization flag of the query table.

- (A) Queries on predefined or composite query tables using the AuthorizationOptions object return entities that correlate with a related work item for this particular user. This is also the case if the AdminAuthorizationOptions object is used and onBehalfUser is set. Standard clients which present task or process lists to users usually use this combination of query tables and query table API parameters.

- (B) The full content of a query table consists of the entities that have a related work item, as configured with the instance-based authorization of the query table. Instance-based authorization considers four types of work items: everybody, individual, group, and inherited. The caller using the API query method must be in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used. This combination of query tables and query table API parameters is intended for use in administrative scenarios where the full list of available tasks or processes must be shown or searched.
- (C) Queries on query tables that do not use instance-based or role-based authorization return the same result if AdminAuthorizationOptions or AuthorizationOptions is passed into the query table API. This is available for supplemental and composite query tables. There is no check on work items or Java EE roles, therefore all authenticated users see the full content. Clients that do not want to restrict object visibility by applying the instance-based or role-based authorization constraints that are provided by Business Process Choreographer can turn off authorization checks when query table definitions are developed. When using claim and complete, however, users must have related work items.
- (D) Template data in predefined query tables or composite query tables with role-based authorization configured can be accessed only with role-based authorization. This requires the caller using the API query method to be in the BPESystemAdministrator Java EE role if the Business Flow Manager EJB is used or the TaskSystemAdministrator Java EE role if the Human Task Manager EJB is used. The query table API can be used to access template information instead of the query API.

Work items and instance-based authorization

Instance-based authorization provided by Business Process Choreographer is based on work items. Each work item describes who has which rights on what object. This information is accessible using the WORK_ITEM query table, if instance-based authorization is used.

The table describes the different types of work items that are considered if instance-based authorization is used when a query is run against a query table:

Table 40. Work item types

Work item type	Description
everybody	Allows all users to access a specific object, such as a task or a process instance. In this case, the EVERYBODY attribute of the related work item is set to TRUE.
individual	Work items that are created for particular users. The OWNER_ID attribute of the related work item is set to a specific user. Multiple work items which differ in the OWNER_ID attribute can exist for an object, such as a task.
group	Work items that are created for users of a particular group. The GROUP_NAME attribute of the related work item is set to a specific group.
inherited	Readers and administrators of process instances are also allowed to inherit the access to the human tasks which belong to these process instances, including escalations. Checks for an inherited work item in task queries are performed with complex SQL joins at runtime, which impacts on performance.

Work items are created by Business Process Choreographer in different situations. For example, at task creation, work items are created for the different roles, such as reader and potential owner, if related people assignment criteria were specified.

The following table describes the types of work items that are created, depending on the people assignment criteria that are defined, if instance-based authorization is used when a query is run on a query table. Inherited work items do not appear in the table because they reflect a relationship that is not explicitly modeled during process application development.

Table 41. Work items and people assignment criteria

Work item type	Related people assignment criteria
everybody	Everybody
individual	All people assignment criteria except verbs *Nobody*, *Everybody*, and *Group*
group	Group

Authorization filter on composite query tables

On composite query tables, you can specify an authorization filter if instance-based authorization is used. This filter restricts the work items which are used for authorization, based on certain attributes of work items. For example, the authorization filter `"WI.REASON=REASON_POTENTIAL_OWNER"` on a composite query table with the TASK primary query table restricts the tasks that are returned when a person runs a query. The result contains only tasks that represent a to-do for that person, that is, the result is restricted to those tasks the person is authorized to claim. This filter can also be specified as the query table filter or as the query filter, but for query performance reasons, it is beneficial to specify these filters as the authorization filter.

Related concepts

"Predefined query tables" on page 468
Predefined query tables provide access to the data in the Business Process Choreographer database. They are the query table representation of the corresponding predefined Business Process Choreographer database views, such as the TASK view or the PROCESS_INSTANCE view. These predefined query tables enhance the functionality and performance of the predefined database views because they are optimized for running process and task list queries.

"Supplemental query tables" on page 471
Supplemental query tables in Business Process Choreographer expose to the query table API business data that is not managed by Business Process Choreographer. With supplemental query tables, this external data can be used with data from the predefined query tables when retrieving business process instance information or human task information.

"Composite query tables" on page 473
Composite query tables in Business Process Choreographer do not have a specific representation of data in the database; they comprise of a combination of data from related predefined and supplemental query tables. Use a composite query table to retrieve the information for a process instance list or task list, such as My To Dos.

"Authorization options for the query table API" on page 504
When you run a query on a query table in Business Process Choreographer, authorization options can be passed as input parameters to the methods of the query table API.

Related tasks

"Creating query tables for Business Process Choreographer Explorer" on page 518
You can use query tables instead of the EJB query API to improve the performance of Business Process Choreographer Explorer. To create the query tables, use the Query Table Builder.

Attribute types for query tables

Attribute types are needed in Business Process Choreographer when query tables are defined, when literal values are used in queries, and when values of a query result are accessed. Rules and mappings are available for each of the attribute types.

A subset of the types that are available in the Java programming language and databases is used to define the type of an attribute of a query table. Attribute types are an abstraction of the concrete Java type or database type. For supplemental query tables, you must use a valid database type to attribute type mapping.

The following table describes the attribute types:

Table 42. Attribute types

Attribute type	Description
ID	The ID which is used to identify a human task (TKIID), a process instance (PIID), or other objects. For example, IDs are used to claim or complete a particular human task, which is identified with the specified TKIID.
STRING	Task descriptions or query properties can be represented as a string.
NUMBER	Numbers are used for attributes, such as the priority on a task.

Table 42. Attribute types (continued)

Attribute type	Description
TIMESTAMP	Timestamps describe a point in time, such as the time when a human task is created, or a process instance is finished.
DECIMAL	Decimals can be used as the type for query properties, for example when defining a query property with a variable of XSD type double.
BOOLEAN	Booleans can have one of two values, true or false. For example, human tasks provide an attribute, autoClaim, which identifies whether the task is claimed automatically if only a single user exists as the potential owner for this task.

Database type to attribute type mapping

Use attribute types to define query tables in Business Process Choreographer, when you run queries on the query tables, and to access values of a query result.

The following table describes the database types and their mapping to attribute types:

Table 43. Database type to attribute type mapping

Database type	Attribute type
A binary type with 16 bytes. This is the type used for IDs such as TKIID on TASK of the Business Process Choreographer tables.	ID
A character based type. The length depends on the column in the database table that is referenced by the attribute of the query table.	STRING
An integer database type, such as integer, short, or long.	NUMBER
A timestamp database type.	TIMESTAMP
A decimal type, such as float or double.	DECIMAL
A type that is convertible to a Boolean value, such as a number. 1 is interpreted as *true*, and all other numbers as *false*.	BOOLEAN

Supplemental query tables typically refer to existing database tables and views, such that table or view creation is not necessary.

Example

Consider a table in a DB2 environment, CUSTOM.ADDITIONAL_INFO, which is to be represented in Business Process Choreographer as a supplemental query table. The following SQL statement creates the database table:

```
CREATE TABLE CUSTOM.ADDITIONAL_INFO
(
  PIID     CHAR(16) FOR BIT DATA,
  INFO     VARCHAR(220),
  COUNT    INTEGER
);
```

The following mapping of database column types to query table attribute types is used for a supplemental query table for the CUSTOM.ADDITIONAL_INFO table.

Table 44. Database types to attribute types mapping example

Database column and type	Query table attribute and type
PIID CHAR(16) FOR BIT DATA	PIID (ID)
INFO VARCHAR(220)	INFO (STRING)
COUNT INTEGER	COUNT (NUMBER)

Attribute type to literal representation mapping

Attribute types are used when query tables are defined in Business Process Choreographer, when queries are run on the query tables, and when values of a query result are accessed. Use this topic for information on attribute type to literal representation mapping.

Literal values can be used in expressions to define filter and selection criteria, such as in filters of composite query tables, and in filters that are passed to the query table API.

The following table describes the attribute types and their mapping to literal values. Placeholders are marked *italic*. Note that the attribute types ID and TIMESTAMP, which can be passed to the query table API, use a special syntax, which is also used by the query API.

Table 45. Attribute type to literal values mapping

Attribute type	Syntax and usage as literal value in expressions
ID	`ID ('string representation of an ID' )`
	When developing client applications, IDs are represented either as a string or as an instance of the com.ibm.bpe.api.OID interface. The string representation can be obtained from an instance of the com.ibm.bpe.api.OID interface using the toString method. The string must be enclosed in single quotation marks.
STRING	`'the string'`
	The string must be enclosed in quotes.
NUMBER	`number`
	The number as text, and no quotation marks. Constants are defined for some number attributes on predefined query tables, and can be used.
TIMESTAMP	`TS ('YYYY-MM-DDThh:mm:ss')`
	The timestamp must be specified as:
	• YYYY is the 4-digit year
	• MM is the 2-digit month of the year
	• DD is the 2-digit day of the month
	• hh is the 2-digit hour of the day (24-hour)
	• mm is the 2-digit minutes of the hour
	• ss is the 2-digit seconds of the minute The timestamp is interpreted as defined in the user's time zone.
DECIMAL	`number.fraction`
	The decimal number as text and no quotation marks; the `.fraction` part is optional.

Table 45. Attribute type to literal values mapping (continued)

Attribute type	Syntax and usage as literal value in expressions
BOOLEAN	`true, false`
	The Boolean value as text.

Example

- `filterOptions.setQueryCondition("STATE=2");`
- `filterOptions.setQueryCondition("STATE=STATE_READY");`
- `a selection criterion on an attached query table TASK_DESC:`
  `"LOCALE='en_US'"`
- `filterOptions.setQueryCondition(`
  `"PTID=ID('_PT:8001011e.1dee8e51.247d6df6.29a60000')");`

Attribute type to parameter mapping

Use attribute types when you define query tables in Business Process Choreographer, when you run queries on the query tables, and to access values of a query result.

The following table describes the attribute types and their mapping to parameter values that can be used in expressions to define filter and selection criteria, such as in filters of composite query tables, and in filters passed to the query table API.

Table 46. Attribute type to user parameter values mapping

Attribute type	Usage as parameter value in expressions
ID	`PARAM(name)`
	When developing client applications, IDs are represented either as a string or as an instance of the com.ibm.bpe.api.OID interface.
	As a parameter, both representations are valid. An array of bytes reflecting a valid OID can also be used (byte).
STRING	`PARAM(name)`
	The string representation of the object that is passed to the query table API at runtime by the toString method.
NUMBER	`PARAM(name)`
	A java.lang.Long, java.lang.Integer, java.lang.Short, or a java.lang.String representation of the number is passed to the query table API. Names of constants, as defined on some attributes of predefined query tables, can be also passed.
TIMESTAMP	`PARAM(name)`
	The following representations are valid:
	- A java.lang.String representation of the timestamp
	- Instances of com.ibm.bpe.api.UTCDate
	- Instances of java.util.Calendar

Table 46. Attribute type to user parameter values mapping (continued)

Attribute type	Usage as parameter value in expressions
DECIMAL	PARAM*(name)* A java.lang.Long, java.lang.Integer, java.lang.Short, java.lang.Double, java.lang.Float, or a java.lang.String representation of the decimal is passed to the query table API.
BOOLEAN	PARAM*(name)* Valid values are: • A java.lang.String representation of the boolean • A java.lang.Short, java.lang.Integer, java.lang.Long with appropriate values; 0 (for false), or 1 (for true) • A java.lang.Boolean object

Example

```
...
// this example shows a query against a composite query table
// COMP.TASKS with a parameter "customer"
java.util.List params = new java.util.ArrayList();

list.add(new com.ibm.bpe.api.Parameter("customer", "IBM");
// the business flow manager Enterprise JavaBeans or the
// human task manager Enterprise JavaBeans can be used to access query tables
service.bfm.queryEntities("COMP.TASKS", null, null, params);
...
```

Attribute type to Java object type mapping

Attribute types are used when query tables are defined in Business Process Choreographer, when queries are run on the query tables, and when values of a query result are accessed. Use this topic for information on attribute type to Java object type mapping.

The following table describes the attribute types and their mapping to Java object types in query result sets.

Table 47. Attribute type to Java object type mapping

Attribute type	Related Java object type
ID	com.ibm.bpe.api.OID
STRING	java.lang.String
NUMBER	java.lang.Long
TIMESTAMP	java.util.Calendar
DECIMAL	java.lang.Double
BOOLEAN	java.lang.Boolean

Example

```
...
// the following example shows a query against a composite query table
// COMP.TA; attribute "STATE" is of attribute type NUMBER
...
// run the query
// the business flow manager Enterprise JavaBeans or the
// human task manager Enterprise JavaBeans can be used to access query tables
```

```
EntityResultSet rs = bfm.queryEntities("COMP.TA",null,null,params);

// get the entities and iterate over it
List entities = rs.getEntities();
for (int i = 0 ; i < entities.size(); i++) {

        // work on a particular entity
        Entity en = (Entity) entities.get(i);

        // note that the following code could be written
        // more generalized using the attribute info objects
        // contained in ei.getAttributeInfo()

        // get attribute STATE
        Long state = (Long) en.getAttributeValue("STATE");
        ...
}
...
```

Attribute type compatibility

Use attribute types when you define query tables in Business Process Choreographer, when you run queries on the query tables, and to access values of a query result.

The following table shows the attribute types and their compatible attribute types, which can be used to define filters and selection criteria in query tables. Compatible attribute types are marked with **X**.

Table 48. Attribute type compatibility

Attribute type	ID	STRING	NUMBER	TIMESTAMP	DECIMAL	BOOLEAN
ID	X					
STRING		X				
NUMBER			X		X	
TIMESTAMP				X		
DECIMAL			X		X	
BOOLEAN						X

In query table expressions that specify filter and condition criteria, types of attributes or values that are compared must be compatible. For example, WI.OWNER_ID=1 is an invalid filter because the left-side operand is of type STRING, and the right-side operand is of type NUMBER.

Query table queries

Queries are run on query tables in Business Process Choreographer using the query table API, which is available on the Business Flow Manager EJB and the REST API.

A query is run on one query table only. Entity-based API methods and row-based API methods are used to retrieve content from query tables. Input parameters are passed into the methods of the query table API.

Related concepts

"Query table development" on page 480
Supplemental and composite query tables in Business Process Choreographer are developed during application development using the Query Table Builder. Predefined query tables cannot be developed or deployed. They are available when Business Process Choreographer is installed and provide a simple view on the artifacts in the Business Process Choreographer database schema.

"Predefined query tables" on page 468
Predefined query tables provide access to the data in the Business Process Choreographer database. They are the query table representation of the corresponding predefined Business Process Choreographer database views, such as the TASK view or the PROCESS_INSTANCE view. These predefined query tables enhance the functionality and performance of the predefined database views because they are optimized for running process and task list queries.

"Supplemental query tables" on page 471
Supplemental query tables in Business Process Choreographer expose to the query table API business data that is not managed by Business Process Choreographer. With supplemental query tables, this external data can be used with data from the predefined query tables when retrieving business process instance information or human task information.

"Composite query tables" on page 473
Composite query tables in Business Process Choreographer do not have a specific representation of data in the database; they comprise of a combination of data from related predefined and supplemental query tables. Use a composite query table to retrieve the information for a process instance list or task list, such as My To Dos.

"Filters and selection criteria of query tables" on page 484
Filters and selection criteria are defined during query table development using the Query Table Builder, which uses a syntax similar to SQL WHERE clauses. Use these clearly defined filters and selection criteria to specify conditions that are based on attributes of query tables.

Query table API methods

Queries are run on query tables in Business Process Choreographer using the query table API. Entity-based API methods and row-based API methods are available to retrieve content from query tables.

The following entity-based methods and row-based methods are provided to run queries on query tables in Business Process Choreographer using the query table API:

Table 49. Methods for queries run on query tables

Purpose	Methods
Query contents	• queryEntities • queryRows Both methods return contents of the query table. The queryEntities method returns content based on entities and queryRows returns content based on rows.
Query the number of objects	• queryEntityCount • queryRowCount Both methods return the number of objects in the query table, while the actual number can depend on whether the entity-based or the row-based approach is taken.

Entity-based queries, using the queryEntities method and the queryEntityCount method, assume that a query table contains uniquely identifiable entities, as defined by the primary key on the primary query table.

Row-based queries, using the queryRows method and the queryRowCount method, return a result set like JDBC, which is row-based, and provides first and next methods for navigating in it. The result set that is returned when you run a query on a query table using the query table API can be compared to QueryResultSet that is returned by the query API. In general, the number of rows is greater than the number of entities that are contained in a query table. The same entity, for example, a human task which is identified by its task ID, such as TKIID, might occur multiple times in the row result set.

A specific instance that is contained in any predefined query table exists only once in a Business Process Choreographer environment. Examples of instances are human tasks and business processes. Those instances are uniquely identified using an ID or a set of IDs. This is the TKIID for instances of human tasks and the PIID for process instances.

Composite query tables are composed of a primary query table and zero or more attached query tables. Objects that are contained in composite query tables are uniquely identified by the unique ID of the objects that are contained in the primary query table. The primary query table of a composite query table determines its entity type. For example, a composite query table with the TASK primary query table contains entities of the TASK type. The one-to-one or one-to-zero relationship between the primary and attached query tables ensures that the attached query tables do not result in duplicate entities.

Entity-based queries exploit the uniquely identifiable entities of a query table, as defined by the primary key on the primary query table. A client application programmer for user interfaces is typically interested in unique instances without duplicates, for example, to display a human task once only on the user interface. Unique instances are returned if the entity-based query table API is used.

Row-based queries can return duplicate rows of the primary query table if instance-based authorization is used.
- Information from the WORK_ITEM query table is retrieved with the query. For example, if the `WI.REASON` attribute is retrieved in addition to the attributes that are defined on the query table, multiple rows qualify for the result. This is because there can be multiple reasons why a user can access an entity, such as, a task or a process instance.
- Instance-based authorization is used, and distinct is not specified. Even though work item information is not retrieved, multiple rows may be returned if instance-based authorization is used.

If the entity-based query table API is used:
- Entity-based queries are always run with the SQL distinct operator.
- Entity-based queries return a result which allows array values for work-item-related information.

Query table API parameters
You use query table API methods to retrieve content when you run queries against a query table in Business Process Choreographer.

The following input parameters are passed to the methods of the query table API:

Table 50. Parameters of the query table API

Parameter	Optional	Type and description
Query table name	No	java.lang.String
		The unique name of the query table.
Filter options	Yes	com.ibm.bpe.api.FilterOptions if the Business Flow Manager Enterprise JavaBeans is used or com.ibm.task.api.FilterOptions if the Human Task Manager Enterprise JavaBeans is used.
		Options which can be used to define the query. For example, a query threshold is set on this parameter to limit the number of results returned.
Authorization options	Yes	com.ibm.bpe.api.AuthorizationOptions or com.ibm.bpe.api.AdminAuthorizationOptions if the Business Flow Manager Enterprise JavaBeans is used. com.ibm.task.api.AuthorizationOptions or com.ibm.task.api.AdminAuthorizationOptions if the Human Task Manager Enterprise JavaBeans is used.
		Authorization can be further constrained if instance-based authorization is used. For query tables which require role-based authorization, an instance of AdminAuthorizationOptions must be passed.
Parameters	Yes	A java.util.List of com.ibm.bpe.api.Parameter if the Business Flow Manager Enterprise JavaBeans is used or com.ibm.task.api.Parameter if the Human Task Manager Enterprise JavaBeans is used.
		This parameter is used to pass user parameters, which are specified in a filter or selection criterion on a composite query table.

A query is run on one specific query table only. The relationship between multiple query tables is defined with composite query tables. In terms of the query API (as distinct from the query table API), this corresponds to database views.

You specify filters and selection criteria in expressions during query table development using the Query Table Builder. For more information, refer to the information center topic about composite query tables and the topic about filter and search criteria of query tables. For information about the Query Table Builder, see the WebSphere Business Process Management SupportPacs site. Look for PA71 WebSphere Process Server - Query Table Builder. To access the link, see the related references section of this topic.

Query table name:

When you run a query on a query table in Business Process Choreographer, the query table name is passed as an input parameter to the methods of the query table API.

The query table name is the name of the query table on which the query is run.
- For predefined query tables, this is the name of the predefined query table.
- For composite and supplemental query tables, this is the name of the respective query table that is specified while modeling the query table. The name of a composite or supplemental query table follows the *prefix.name* naming convention, and *prefix* may not be 'IBM'.

Both the query table name and prefix must be in uppercase. The maximum length of the query table name is 28 characters.

Filter options for query tables:

When you run a query on a query table in Business Process Choreographer, filter options can be passed as input parameters to the methods of the query table API.

An instance of the com.ibm.bpe.api.FilterOptions class if the Business Flow Manager Enterprise JavaBeans is used, or an instance of the com.ibm.task.api.FilterOptions if the Human Task Manager Enterprise JavaBeans is used, can be passed to the query table API. The filter options allow a configuration of the query using:
- A threshold and offset (skipCount)
- Sort attributes (similar to the ORDER BY clause in an SQL query)
- A user-provided query filter
- The set of attributes returned, including work item information
- Other

The result set that can be obtained from a query table is specified by the definition of the query table. However, you might want to specify additional options when the query is run. The following table describes the options that can be specified as filter options using the FilterOptions object.

Table 51. Query table API parameters: Filter options

Option	Type	Description
Selected attributes	java.lang.String	• A comma separated list of attributes of the query table that must be returned in the result set. • If instance-based authorization is used, work item information can be retrieved by specifying attributes of the WORK_ITEM query table, prefixed with `WI.`, for example, `WI.REASON`. • If null is specified, all attributes of the query table are returned, without work item information.
Sort attributes	java.lang.String	A comma separated list of attributes of the query table, optionally followed by ASC or DESC, for ascending or descending, respectively. This list is similar to the SQL ORDER BY clause: *sortAttributes* ::= *attribute* [ASC | DESC] [, *sortAttributes*]. If ASC or DESC is not specified, ASC is assumed. Sorting occurs in the sequence of the sort attributes. This example sorts tasks in query table TASK in descending order by state, and within the groups of the same STATE by NAME, in ascending order: `"STATE DESC, NAME ASC"`.
Threshold	java.lang.Integer	Defines the maximum: • Number of rows returned if queryRows is used. • Number of entities returned if queryEntities is used. The actual number of available entities in the respective query table may exceed the threshold number of entities for the query even if the entity result set does not contain as many entities as the threshold number. This is due to technical reasons if work item information is selected. • Count returned if queryRowCount or queryEntityCount is used. The default is null which means that no threshold is set.

Table 51. Query table API parameters: Filter options (continued)

Option	Type	Description
Skip count	java.lang.Integer	Defines the number of rows (row-based queries) or the number of entities (entity-based queries) that are skipped. As with the threshold parameter, skipCount may not be accurate for entity-based queries. Skip count is used to allow paging over a large result set. The default is null which means that no skipCount is set.
Time zone	java.util.TimeZone	The time zone that is used when converting timestamps. An example is CREATED on the predefined query table TASK. If not specified (null), the time zone on the server is used.
Locale	java.util.Locale	The locale which is used to calculate the value of the $LOCALE system parameter. An example usage of $LOCALE in a selection criterion is: 'LOCALE=$LOCALE'.
Distinct rows	java.lang.Boolean	Used for row-based queries only. If set to true, row-based queries return distinct rows. This does not imply that unique rows are returned due to the possible multiplicity of work item information.
Query condition	java.lang.String	This option performs additional filtering on the result set. All of the attributes that are defined for the query table can be referenced. If authorization is required for the query table, columns that are defined for the WORK_ITEM query table can also be referenced using the WI prefix, for example, WI.REASON=REASON_POTENTIAL_OWNER.

Authorization options for the query table API:

When you run a query on a query table in Business Process Choreographer, authorization options can be passed as input parameters to the methods of the query table API.

Use an instance of the com.ibm.bpe.api.AuthorizationOptions class or the com.ibm.bpe.api.AdminAuthorizationOptions if the Business Flow Manager EJB is used, or an instance of the com.ibm.task.api.AuthorizationOptions class or the com.ibm.task.api.AdminAuthorizationOptions class if the Human Task Manager EJB is used, to specify additional authorization options when the query is run.

If instance-based authorization is used, instances of the AuthorizationOptions class allow the specification of the type of work items used to identify eligible instances that are returned by the query.

An instance of the AuthorizationOptions class can be passed to the query table API if the query is run on a predefined query table that contains instance data. It can also be passed if the query is run on a composite query table with a primary query table that contains instance data and instance-based authorization is configured to be used. If the query is run on a predefined query table with template data or a composite query table with role-based authorization configured, an com.ibm.bpe.api.EngineNotAuthorizedException exception is thrown if the Business Flow Manager EJB is used or com.ibm.task.api.NotAuthorizedException is thrown if the Human Task Manager EJB is used. In all other cases, the authorization options passed to the query table API are ignored.

Composite query tables can restrict the types of work items that are considered when identifying objects (or entities) that are contained in it. For example, if the authorization options that are passed to the query table API are configured to use

everybody work items, this is only taken into account if everybody work items are defined for use on the definition of the composite query table. As a simple rule, a work item type that is not specified to be considered on the query table definition cannot be overwritten to be considered by the query table API, but a work item type that is specified to be considered on the query table definition can be overwritten not to be used. Also, the authorization type of a composite or predefined query table cannot be overwritten by the query table API.

Depending on the type of query table that is queried, different authorization option defaults apply if the authorization object is not specified or if the related attributes (everybody, individual, group, or inherited) are set to null, which is the default.

The following table shows the authorization option defaults for instance-based authorization for the query table type and work item type used.

Table 52. Query table API parameters: Authorization option defaults for instance-based authorization

Query table type	Everybody work item	Individual work item	Group work item	Inherited work item
Predefined with instance data	TRUE	TRUE	TRUE	FALSE
Predefined with template data	N/A	N/A	N/A	N/A
Composite with a primary query table with instance data	TRUE	TRUE	TRUE	TRUE
Composite with a primary query table with template data	N/A	N/A	N/A	N/A
Supplemental	N/A	N/A	N/A	N/A

N/A means that instance-based authorization is not used and, therefore, any setting on the authorization object with respect to work items is ignored.

If TRUE is specified, the resulting query will only consider the specific work item type if the query table is defined to use this type of work item. This is true for all predefined query tables with instance data, but might not be true for a composite query table. For the group work item, the latter must also be enabled on the human task container. An example of the inherited work item set to TRUE is that the administrator of a process instance may see participating human task instances that are created for that process instance.

Specify an instance of the AdminAuthorizationOptions class instead of an instance of the AuthorizationOptions class if:
- A query is run on a query table with role-based authorization. Predefined query tables with template data require role-based authorization, and composite query tables with a primary query table with template data can be configured to require role-based authorization.
- A query is run on a query table with instance data or on a composite query table with a primary query table that contains instance data. It should return the content of that query table, regardless of restrictions due to authorization for a

particular user. This behavior is equivalent to using the queryAll method on the query API (as distinct from the query table API).

- A query should be executed on behalf of another user.

The following table describes how the various behaviors above are accomplished:

Table 53. Query table API parameters: AdminAuthorizationOptions

Situation	Description
onBehalfUser set to null	• If the query is run on a query table with role-based authorization, all contents of that query table are returned. • If the query is run on a query table which uses instance-based authorization, the particular objects contained in the query table are not checked for work items for a particular user. All objects that are contained in the query table are returned.
onBehalfUser set to a particular user	The query is run with the authority of the specified user, and the objects in the query table are checked against the work items for this user, if instance-based authorization is used.

If you specify AdminAuthorizationOptions, the caller must be in the BPESystemAdministrator or BPESystemMonitor Java EE role if the Business Flow Manager EJB is used, or in the TaskSystemAdministrator or TaskSystemMonitor Java EE role if the Human Task Manager EJB is used.

Related concepts

"Authorization for query tables" on page 489
You can use instance-based authorization, role-based authorization, or no authorization when you run queries on query tables.

Parameters:

When you run a query on a query table in Business Process Choreographer, you can pass user parameters as input parameters to the methods of the query table API. In query table definitions, you can specify parameters in filters on the primary query table, on the authorization, and on the query table. Parameters can also be specified in selection criteria on attached query tables.

The system parameters, $USER and $LOCALE, are replaced at runtime in filters and selection criteria, and are not required to be passed into the query table API. The input value for the calculation of the $LOCALE system parameter is provided by setting the locale in the filter options.

User parameters must be passed into the query table API when the query is run. This is accomplished by passing a list of instances of the com.ibm.bpe.api.Parameter class if the Business Flow Manager EJB is used, or an instance of the com.ibm.task.api.Parameter class if the Human Task Manager EJB is used.

The following properties must be specified on a parameter object:

Table 54. User parameters for the query table API

Property	Description
Name	The name of the parameter as used in the query table definition. The name is case sensitive.
Value	The value of the parameter. The type of the parameter must be compatible with the type of the left-hand operand of all filters and selection criteria where this parameter is used. Constants that are defined on some attributes of predefined query tables can be passed as a string, for example `STATE_READY`.

Example

```
// execute a query against a composite query
// table CUST.CPM with the primary query table filter
// set to 'STATE=PARAM(theState)'
EntityResultSet ers = null;
List parameterList = new ArrayList();
parameterList.add(new Parameter
("theState", new Integer(2)));

// run the query;
// the business flow manager EJB or the
// human task manager EJB can be used to access query tables
ers = bfm.queryEntities
("CUST.CPM", null, null, parameterList);

// work on the result set
// ...
```

Results of query table queries

You use query table API methods when you run queries on a query table in Business Process Choreographer. The result of a queryEntityCount method or queryRowCount method query is a number. The queryEntities and the queryRows methods return result sets.

EntityResultSet

An instance of the com.ibm.bpe.api.EntityResultSet class is returned by the method queryEntities if the Business Flow Manager Enterprise JavaBeans is used. An instance of the com.ibm.task.api.EntityResultSet class is returned by the method queryEntities if the Human Task Manager Enterprise JavaBeans is used. An entity result set has the following properties:

Table 55. Entity result set properties of a query table API entity

Property	Description
queryTableName	Name of the query table on which the query was run.
entityTypeName	• If the query was run on a composite query table, this is the name of the primary query table. • If the query was run on a predefined query table or on a supplemental query table, this is the name of the query table, that is, the same value as the *queryTableName* property.

Table 55. Entity result set properties of a query table API entity (continued)

Property	Description
EntityInfo	This property contains the meta information of the entities that are contained in the entity result set. A java.util.List list of com.ibm.bpe.api.AttributeInfo objects if the Business Flow Manager EJB is used, or a list of com.ibm.task.api.AttributeInfo objects if the Human Task Manager EJB is used, can be retrieved on this object. This list contains the attribute names and attribute types of the information contained in the entities of this result set. Meta information about the attributes which constitute the key for these entities is also contained.
entities	A java.util.List list of com.ibm.bpe.api.Entity objects if the Business Flow Manager EJB is used, or a list of com.ibm.task.api.Entity objects if the Human Task Manager is used.
locale	The locale that is calculated for the $LOCALE system parameter.

Instances of the Entity class contain the information that is retrieved from the query table query. An entity represents a uniquely identifiable object such as a task, a process instance, an activity, or an escalation. The following properties are available for entities:

Table 56. Entity properties of a query table API entity

Property	Description
EntityInfo	The EntityInfo object which is also contained in the entity result set. A java.util.List list of com.ibm.bpe.api.AttributeInfo objects if the Business Flow Manager EJB is used, or a list of com.ibm.task.api.AttributeInfo objects if the Human Task Manager EJB is used, can be retrieved on this object. This list contains the attribute names and attribute types of the information contained in the entities of this result set. Meta information about the attributes which constitute the key for these entities is also contained.
attributeValue (*attributeName*)	The value of the specified attribute that is retrieved for this entity. The type is contained in the related AttributeInfo object of this attribute.
attributeValuesOfArray (*attributeName*)	An array of values. Use this property if the attribute info property *array* is set to true which is currently the case only if the attribute refers to work item information.

The number of entities in the entity result set is retrieved using the size method on the list of entities.

Example: Entity-based query table API:

```
...
// the following example shows a query against
// predefined query table TASK, using the entity-based API

...
// run the query
// service is a (Local)BusinessFlowManager object or a
// (Local)HumanTaskManager object
EntityResultSet rs = service.queryEntities("TASK", null, null, null);
```

```
        // get the entities meta information
        EntityInfo ei = rs.getEntityInfo();
        List atts = ei.getAttributeInfo();

        // get the entities and iterate over it
        Iterator entitiesIter = rs.getEntities().iterator();
        while (entitiesIter.hasNext()) {

                // work on a particular entity
                Entity en = (Entity) entitiesIter.next();

                for (int i = 0; i < atts.size(); i++) {
                        AttributeInfo ai = (AttributeInfo) atts.get(i);
                        Serializable value = en.getAttributeValue(ai.getName()) ;

                        // process...
                }
        }
        ...
```

RowResultSet

An instance of the com.ibm.bpe.api.RowResultSet class is returned by the method queryRows if the Business Flow Manager Enterprise JavaBeans is used. An instance of the com.ibm.task.api.RowResultSet class is returned by the method queryRows if the Human Task Manager Enterprise JavaBeans is used. This type of result set is similar to a JDBC result set. A row result set has the following properties:

Table 57. Row result set properties of a query table API row

Property	Description
primaryQueryTableName	• If the query was run on a composite query table, this is the name of the primary query table. • If the query was run on a predefined query table or on a supplemental query table, this is the name of the query table, that is, the same value as property *queryTableName*.
attributeInfo	This property contains a list of com.ibm.bpe.api.AttributeInfo objects if the Business Flow Manager Enterprise JavaBeans is used or a list of com.ibm.task.api.AttributeInfo objects if the Human Task Manager Enterprise JavaBeans is used. They describe the meta information for this result set. AttributeInfo objects contain the attribute names and attribute types of the information. Meta data about keys is not contained because row result sets do not have a key.
attributeValue	The value of the specified attribute that was retrieved for this row. The type is contained in the related AttributeInfo object of this attribute.
next, first, last, previous	The row result set is navigated using these methods. Compare its usage to iterators, enumerations, or JDBC result sets.

The number of rows in the row result set is retrieved using the size method on the list of rows.

Example: Row-based query table API

```
...
// the following example shows a query against
// predefined query table TASK, using the entity-based API
...
// run the query
// service is a (Local)BusinessFlowManager object or a
// (Local)HumanTaskManager object
RowResultSet rs = service.queryRows("TASK", null, null, null);

// get the entities meta information
List atts = rs.getAttributeInfo();

// get the entities and iterate over it
while (rs.next()) {

    // work on a particular row
    for (int i = 0; i < atts.size(); i++) {
        AttributeInfo ai = (AttributeInfo) atts.get(i);
        Serializable value = rs.getAttributeValue(ai.getName()) ;

        // process...
    }
}
...
```

Query table queries for meta data retrieval

Queries are run on query tables in Business Process Choreographer using the query table API. Methods are available to retrieve meta data from query tables.

The following methods are provided to retrieve meta data when you run queries on query tables in Business Process Choreographer using the query table API:

Table 58. Methods for meta data retrieval on query tables

Purpose	Method
Return the meta data of a specific query table	getQueryTableMetaData
Return a list of query table meta data with specific properties	findQueryTableMetaData
Return contents of a query table, based on entities, and a subset of the meta data for the selected attributes	queryEntities
Return contents of a query table, based on rows, and a subset of the meta data for the selected attributes	queryRows

Meta data of query tables consists of data that relates to structure and data that relates to internationalization.

The following table shows the meta data that is related to the structure of a query table.

Table 59. Meta data related to query table structure

Meta data	Description	Returned by getQuery-TableMetaData	Returned by findQuery-TableMetaData	Returned by queryEntities	Returned by queryRows
Query table name	The name of the query table	Yes	Yes	Yes	Yes
Primary query table name	For supplemental and predefined query tables, name of the query table; for composite query tables the name of the primary query table	Yes	Yes	Yes	Yes
Kind	The type of query table: predefined, composite, or supplemental	Yes	Yes	No	No
Authorization	The authorization that is defined on the query table: • Use of work items • Instance-based, role-based, or no authorization	Yes	Yes	No	No
Defined attributes	Meta data of the attributes that are defined on the query table	Yes	Yes	No. Meta data of the selected attributes is returned.	No. Meta data of the selected attributes is returned.
Key attributes	Key attributes of the query table	Yes	Yes	Yes	No. Not applicable to row-based queries.

The following table shows the meta data that is related to the internationalization of a query table.

Table 60. Meta data related to query table internationalization

Meta data	Description	Returned by getQuery-TableMetaData	Returned by findQuery-TableMetaData	Returned by queryEntities	Returned by queryRows
locales[]	Locales for which display names and descriptions of the query table and attributes are defined.	Yes	Yes	No	No
Locale	Value of the $LOCALE system parameter which results from the locale that is passed to the API.	Yes	Yes	Yes	Yes
Display name and description of the query table	Display names and descriptions for the query table, which are provided for all defined locales.	Yes	Yes	No	No
Display names and descriptions of the attributes	Display names and descriptions for the attributes, which are provided for all defined locales.	Yes	Yes	No	No

All EJB query table API methods which return query table meta data accept a locale parameter, such as `FilterOptions.setLocale` and `MetaDataOptions.setLocale`. This parameter should be set to the Java locale that the client uses to present information to the user. This locale parameter is used to calculate the value of the `$LOCALE` system parameter, which can be used in filters and selection criteria. The locale that is returned contains the actual Java locale that is used for `$LOCALE`.

If the display names and descriptions of a specific query table are retrieved, pass getLocale to the related methods to get the display names and descriptions in the same locale as the descriptions of the tasks. For example, these descriptions are attached using a selection criterion of `'LOCALE=$LOCALE'`.

Example

```
// the following example shows how meta data for a particular
// composite query table can be retrieved

...
// run the query
MetaDataOptions mdo = new MetaDataOptions("TASK", null, false, new Locale("en_US"));
List list = bfm.findQueryTableMetaData(mdo);

// to get the meta data of a specific query table
// use bfm.getQueryTableMetaData(...)

// iterate through the list of query tables that have TASK as primary query table
// => at least one query table is returned: the predefined query table TASK

Iterator iter = list.iterator();
while (iter.hasNext()) {
 QueryTableMetaData md = (QueryTableMetaData) iter.next();
 Locale effectiveLocale = md.getLocale();
 String queryTableDisplayName = md.getDisplayName(effectiveLocale);
 System.out.println("found query table: " + queryTableDisplayName);
 List attributesList = md.getAttributeMetaData();
 Iterator attrIter = attributesList.iterator();
 while (attrIter.hasNext()) {
  AttributeMetaData amd = (AttributeMetaData) attrIter.next();
  String attributeDisplayName = amd.getDisplayName(effectiveLocale);
  System.out.println("\tattribute:" + attributeDisplayName);
 }
}
```

Best match locale

When specifying the conditions on an attached query table, using the value `$LOCALE` can return unexpected results if the specified locale does not match the metadata exactly. For example if you pass a locale en_US with a query against a query table that has metadata specifying the language as en, the result returned will be `null`.

To avoid such cases, you can use `LOCALE=$LOCALE_BEST_MATCH`, which applies a best-match algorithm to calculate the actual locale used in the query. For example, a query with locale en_US against a query table in language en is performed using en.

You cannot specify any other logical or comparison operators in the condition `LOCALE=$LOCALE_BEST_MATCH`. You can only use the best-match locale condition on attached query tables, specifying it as a condition on other queries results in an error.

Internationalization for query table meta data

Internationalization is supported for query table meta data.

Display names and descriptions can be provided for composite query tables in different locales. For example, a composite query table can define a display name for the query table in the en_US locale, the de locale, and in the default locale. This is done when the query table is developed using the Query Table Builder. To deploy query tables with localized display names and descriptions, the -deploy jarFile option must be used when the query table is deployed on the Business Process Choreographer container.

In terms of locale handling, the behavior of the query table API methods, queryEntities and queryRows, and the meta data methods of the query table API, getQueryTableMetaData and findQueryTableMetaData, is similar to that provided by Java resource bundles.

To make the display names and descriptions of the query table meta data consistent with the contents of the query table, the value of the $LOCALE system parameter depends on the locales for which display names and descriptions are specified on the query table.

Example

Consider the following scenario of a client which displays task lists or process lists and creates a request to query a query table.

- The client did not specify the locale it uses to present information to the user. It is likely that the application is not enabled for different languages.
  - A default locale is specified on the query table for display names and descriptions. This is the case for all composite and supplemental query tables that are built with the current version of the Query Table Builder. Therefore, the value of $LOCALE is set to default.
  - The query table does not specify display names or descriptions on the query table for the default locale. This is the case for all predefined query tables and for all query tables that are deployed using the -deploy qtdFile option. The value of $LOCALE is based on the Java resource bundle method.
- The client specified the locale to use to present information to the user. For example, this is the case when the REST API for query tables is used.
  - Display names and descriptions are specified on the query table. The Java resource bundle method is used to calculate the value of $LOCALE, based on the locale that is passed in by the client.
  - Display names and descriptions are not specified on the query table. The value of $LOCALE is set to the value that is passed in by the client.

Best match locale

When specifying the conditions on an attached query table, using the value $LOCALE can return unexpected results if the specified locale does not match the metadata exactly. For example if you pass a locale en_US with a query against a query table that has metadata specifying the language as en, the result returned will be null.

To avoid such cases, you can use LOCALE=$LOCALE_BEST_MATCH, which applies a best-match algorithm to calculate the actual locale used in the query. For example, a query with locale en_US against a query table in language en is performed using en.

You cannot specify any other logical or comparison operators in the condition LOCALE=$LOCALE_BEST_MATCH. You can only use the best-match locale condition on attached query tables, specifying it as a condition on other queries results in an error.

Related tasks

"Creating query tables for Business Process Choreographer Explorer" on page 518 You can use query tables instead of the EJB query API to improve the performance of Business Process Choreographer Explorer. To create the query tables, use the Query Table Builder.

Query tables and query performance

Query tables introduce a clean programming model for developing client applications that retrieve lists of human tasks and business processes in Business Process Choreographer. Using query tables improves the performance. Information is provided about the query table API parameters and other factors that affect the performance.

Query response times on query tables depend mainly on the authorization options, filters, and selection criteria that are used. The following are some general performance tips to consider.

- Authorization options have considerable performance impact. Enable authorization using as few options as is possible, such as individual and group work items. Avoid using inherited work items. The authorization options can be further restricted when the query is run. Also, if not needed, specify that authorization using work items is not required.
- If authorization using work items is required, specify an authorization filter. For example, to allow only objects in the query table with a potential owner work item, use WI.REASON=REASON_POTENTIAL_OWNER.
- Filtering on the primary query table is efficient, for example, to allow only tasks in the ready state in the query table where TASK is the primary query table.
- Filters on the query table, as well as query filters, which are filters that are passed when the query is run, are less efficient as primary filters in terms of performance.
- Avoid, where possible, using parameters in filters and selection criteria.
- Avoid using LIKE operators in filters and selection criteria.

Composite query table definition

The following table provides information about the query performance impact of options that are defined on composite query tables. It also provides information other topics related to composite query table definitions. The impact given in column Performance Impact is an average performance impact, actual impact observations may vary.

Table 61. Query performance impact of composite query table options

Object or topic	Performance impact	Description
Query table filter	Negative	Filters on query tables are the filters with the highest negative impact on query performance. These filters typically cannot use any defined indexes in the database.
Primary query table filter	Positive	A filter on the primary query table provides high performance filtering at a very early stage of the query result set calculation. It is suggested to restrict the contents of the query table using a primary query table filter.
Authorization filter	Positive	A filter on authorization can improve the performance of the query, such as how the primary query table filter improves it. If possible, an authorization filter should be applied. For example, if reader work items should not be considered, specify WI.REASON=REASON_READER.
Selection criteria	None	Some primary query table to attached query table relationships require the definition of a selection criterion in order to meet the one-to-one or one-to-zero relationship. A selection criterion typically has low performance impact because it is evaluated for a small numbers of rows only.
Parameters	None	Currently, using parameters in query tables has no negative performance impact. Nevertheless, parameters should be used only if needed.
Instance-based authorization	Negative	If instance-based authorization is used, each object in the query table must be checked against the existence of a work item. Work items are represented as entries in the WORK_ITEM query table. This verification affects performance.
Instance-based authorization: • everybody • individuals • groups • inherited	Negative	Each type of work item that is specified for use in the query table has a performance impact. Applications with high volume queries should only use individual and group work items, or only one of those. Inherited work items are usually not required, in particular when defining task lists that return human tasks representing to-dos. They should be used only when it is clear that they are needed, for example, to return lists of tasks that belong to a business process where a person might have read access based on the authorization for the enclosing business process.
Role-based authorization or no authorization	None	If role-based authorization or no authorization is used, checks against work items are not made.
Number of defined attributes	Currently none	Currently, the number of attributes contained in a query table has no impact on performance. Nevertheless, only those attributes that are needed should be part of a query table.

Query table API

The following table provides information about the query performance impact of options that are specified on the query table API. The impact given in the Performance impact column is an average performance impact; actual impact observations may vary.

Table 62. Query performance impact of query table API options

Option	Performance impact	Description
Selected attributes	Negative (less is better)	The number of attributes that are selected when a query is run on a query table impacts on the number that need to be processed both by the database and by the Business Process Choreographer query table runtime. Also, for composite query tables, information from attached query tables need be retrieved only if those are either specified by the selected attributes or referenced by the query table filter or by the query filter.
Query filter	Negative	If specified, the query filter currently has the same performance impact as the query table filter. However, it is a good practice if filters are specified on query tables rather than passed into the query table API.
Sort attributes	Negative	The sorting of query result sets is an expensive operation, and database optimizations are restricted if sorting is used. If not needed, sorting should be avoided. Most applications require sorting, however.
Threshold	Positive	The specification of a threshold can greatly improve the performance of queries. It is a best practice to always specify a threshold.
Skip count	Negative	Skipping a particular number of objects in the query result set is expensive and should be done only if required, for example when paging over a query result.
Time zone	None	The time zone setting has no performance impact.
Locale	None	The locale setting has no performance impact.
Distinct rows	Negative	Using distinct in queries has some performance impact but might be necessary in order to retrieve non-duplicate rows. This option impacts only on row based queries and is ignored otherwise.
Count queries	Positive	If only the total number of entities or the number of rows for a particular query is needed, that is, the contents are not needed for all entries of the query table, the method queryEntityCount or queryRowCount should be used. The Business Process Choreographer runtime can apply optimizations that are valid only for count queries.

Other considerations

Other factors to consider with regard to performance are:

Table 63. Query table performance: Other considerations

Item	Description
Number of query tables on the system	The number of query tables which are deployed on a Business Process Choreographer container does not influence the performance of query table queries. Also, currently, it does not influence the navigation of business process instances, nor does it have impact on claim or complete operations on human tasks. Due to maintainability, keep the number of query tables at a reasonable level. Typically, one query table represents one task list or process list which is displayed on the user interface.

Table 63. Query table performance: Other considerations (continued)

Item	Description
Database tuning	Although optimized SQL is used to access the contents of a query table, database tuning needs to be implemented on a Business Process Choreographer database:
	• Database memory should be set to a maximum, taking into account other processes that are running on the database server, as well as hardware constraints.
	• Statistics on the database must be up-to-date, and should be updated on a regular basis. Typically, those procedures are already implemented in large topologies. For example, collect database statistics for the optimizer once per week in order to reflect changes of the data in the database.
	• Database systems provide tools to reorganize (or defragment) the data containers. The physical layout of the data in a database can also influence query performance and access paths of queries.
	• Optimal indexes are the key for good query performance. Business Process Choreographer comes with predefined indexes which are optimized for both process navigation and query performance of typical scenarios. In customized environments, additional indexes may be necessary in order to support high volume task or process list queries. Use tools provided by the database in order to support the queries which are run on a query table.

Creating query tables for Business Space

In the Query Table Builder, you can use the composite query table definition that has predefined properties to create query tables for Business Space.

Before you begin

The Query Table Builder is available as an Eclipse plug-in and can be downloaded from the WebSphere Business Process Management SupportPacs site. Look for PA71 WebSphere Process Server - Query Table Builder. To access the link, see the related references section of this topic.

Procedure

1. In the Query Table Builder, right click your project, then select **New →
   Composite Query Definition for Business Space**. Follow the wizard
   instructions to create a query table definition. The new query table definition
   consists of the predefined properties. If required, add more properties to the
   query table definition and deploy the query table definition file to the
   WebSphere Process Server.

   Note: The names you give the properties in the Query Table Builder are used
   as the names for the task properties in Business Space for Choreographer.

2. After the query table definition files have been created and deployed you can
   configure them in Business Space. For example, if you have deployed a query
   table definition file for the Tasks List widget:

   a. Open the widget menu and select **Configure**, then the **Content** tab.

b. On the **Content** tab, open the **Select task list to display** drop down list to display the lists that you can make available to the user of the widget. Select **Add task lists**. The query table definition you deployed should be available in this list for selection.

If the query table definition is not available you need to go back to the Query Table Builder and check whether the definition file was correctly defined and deployed.

Creating query tables for Business Process Choreographer Explorer

You can use query tables instead of the EJB query API to improve the performance of Business Process Choreographer Explorer. To create the query tables, use the Query Table Builder.

Before you begin

The Query Table Builder is available as an Eclipse plug-in and can be downloaded from the WebSphere Business Process Management SupportPacs site. Look for PA71 WebSphere Process Server - Query Table Builder. To access the link, see the related references section of this topic.

Procedure

1. In the Query Table Builder, right click your project, then select **New → Composite Query Definition for Business Space**. This option ensures that all of the columns that are required for Business Process Choreographer Explorer are preselected.

2. Follow the wizard instructions to create a query table definition. If required, add more properties to the query table definition. Consider the following aspects when you define your query table:

   Filter criteria
   When you create views in Business Process Choreographer Explorer based on query tables, you cannot specify additional filter or variables for your search criteria. You must specify these filter criteria and the parameters for the variables when you create the query table.

   You can use a query table for more than one view in Business Process Choreographer Explorer by using parameters in the query table definition. For more flexibility, you can also specify whether the default values of the parameters can be overwritten when the query for the custom view is run.

   Authorization
   When you create views in Business Process Choreographer Explorer based on query tables, you cannot filter your search criteria based on the user role. You must set the filter criteria for user roles when you define the query table. For primary query tables based on template information, use instance-based authorization as the authorization type and not role-based authorization. For primary query tables based on instance information, specify the appropriate instance-based authorization filter.

   Internationalization
   When you define properties in the Query Table Builder, you can also specify the names and descriptions of these properties in different languages. When the query for the customized view is run, Business

Process Choreographer Explorer uses the translation that is appropriate for the language setting of your browser.

Display name and description for the query table definition
In the Query Table Builder, you can provide a display name and description for all of the languages that are supported by the view.

Display name and description for the columns
At runtime, Business Process Choreographer Explorer retrieves the appropriate internationalized column names that are displayed in a result list. For columns that come from your primary query table, such as PIID, Business Process Choreographer Explorer uses the translations that are already available for all of the supported languages.

For columns that come from an attached query table, such as QUERY_PROPERTY, you need to provide display names and descriptions in the Query Table Builder in all of the languages that are supported by your business.

Task names and description
If you have internationalized task names and descriptions in WebSphere Integration Developer, they are displayed in Business Process Choreographer Explorer according to the language and country settings of your browser. If your browser settings do not match the settings that are defined in the process model, the translation of the default language is used.

Sort criteria
When you define sort criteria for a query table definition, be aware that several properties, for example, process state, are stored as integer values, while Business Process Choreographer Explorer displays them as translated strings in the resulting list. This might produce unexpected sorting results in some languages.

The new query table definition consists of the predefined properties and any additional properties that you define.

What to do next

Deploy and test the query definition in the Query Table Builder on an application server. If this is the server to which Business Process Choreographer Explorer is connected, you can now use the query table when you customize Business Process Choreographer Explorer for your own use, or for different user groups. If Business Process Choreographer Explorer is connected to a different server, you must deploy the query table on the appropriate server before you can use it to create customized views.

Business Process Choreographer EJB query API

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 Java EE 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 Java EE 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 64. Query syntax for different object types

Object	Syntax

Process template	`ProcessTemplateData[] queryProcessTemplates`
	`                              (java.lang.String whereClause,`
	`                               java.lang.String orderByClause,`
	`                               java.lang.Integer threshold,`
	`                               java.util.TimeZone timezone);`

Task template	`TaskTemplate[]        queryTaskTemplates`
	`                              (java.lang.String whereClause,`
	`                               java.lang.String orderByClause,`
	`                               java.lang.Integer threshold,`
	`                               java.util.TimeZone timezone);`

Business-process and task-related data	`QueryResultSet query (java.lang.String selectClause,`
	`                              java.lang.String whereClause,`
	`                              java.lang.String orderByClause,`
	`                              java.lang.Integer skipTuples`
	`                              java.lang.Integer threshold,`
	`                              java.util.TimeZone timezone);`

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.

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, specify a value for the threshold parameter. If you set the threshold parameter accordingly, the database query is faster and less data needs to transfer from the server to the client. 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 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.

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.

Query results

A query result set contains the results of a Business Process Choreographer API 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 getString getTimestampAsLong
Integer	getInteger getShort getLong getString getBoolean
Boolean	getBoolean getShort getInteger getLong 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.

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.

Related concepts

"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.

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.

Related tasks

"Managing stored queries" on page 532
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).

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), null);
```

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 use the Human Task Manager API to retrieve information about a stored query, use StoredQuery for the returned object 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.

   ```
   StoredQueryData 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();
   ```

   If you use the Human Task Manager API to retrieve information about a stored query, use StoredQuery for the returned object instead of StoredQueryData.

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.

   ```
   StoredQueryData storedQuery = process.getStoredQuery
       ("CustomerOrdersStartingWithA");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();
   String owner = storedQuery.getOwner();
   ```

   If you use the Human Task Manager API to retrieve information about a stored query, use StoredQuery for the returned object instead of StoredQueryData.

5. Delete a private stored query.

   The following code snippet shows how to delete a private stored query.

   ```
   process.deleteStoredQuery("CustomerOrdersStartingWithA");
   ```

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.

537

- Work with the data.

Related concepts

Alternate administration authorization mode
If your process template was modeled to include administration tasks, this administration mode will deactivate them and related actions such as escalations and monitoring. Running in this mode improves performance by disabling instance-based administration, and restricting process administration and monitoring to users who are system administrators and system monitors.

Related reference

"BusinessFlowManagerService interface" on page 564
The BusinessFlowManagerService interface exposes business-process functions that can be called by a client application.

"HumanTaskManagerService interface" on page 582
The HumanTaskManagerService interface exposes task-related functions that can be called by a local or a remote client.

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 for business processes or human tasks 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 Platform, Enterprise Edition (Java EE) 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. Decide on how you are going to provide definitions of business objects.

To work with business objects in a remote client application, you need to have access to the corresponding schemas for the business objects (XSD or WSDL files) that are used to interact with a process or task. Access to these files can be provided in one of the following ways:

- If the client application does not run in a Java EE managed environment, package the files with the client application's EAR file.
- If the client application is a Web application or an EJB client in a managed Java EE environment, either package the files with the client application's EAR file or leverage remote artifact loading.

    a. Use the Business Process Choreographer EJB API createMessage and the ClientObjectWrapper.getObject methods to load the remote business object definitions from the corresponding application on the server transparently.

    b. Use the Service Data Object Programming API to create or read a business object as part of an already instantiated business object. Do this by using the commonj.sdo.DataObject.createDataObject or getDataObject methods on the DataObject interface.

    c. When you want to create a business object as the value for a business object's property that is typed using the XML schema any or anyType, use the Business Object services to create or read your business object. To do this, you must set the remote artifact loader context to point to the application that the schemas will be loaded from. Then you can use the appropriate Business Object services.

    For example, create a business object, where "ApplicationName" is the name of the application that contains your business object definitions.

```
BOFactory bofactory = (BOFactory) new
    ServiceManager().locateService("com/ibm/websphere/bo/BOFactory");

com.ibm.wsspi.al.ALContext.setContext
    ("RALTemplateName", "ApplicationName");
try {
    DataObject dataObject = bofactory.create("uriName", "typeName" );
    } finally {
    com.ibm.wsspi.al.ALContext.unset();
    }
```

    For example, read XML input, where "ApplicationName" is the name of the application that contains your business object definitions.

```
BOXMLSerializer serializerService =
    (BOXMLSerializer) new ServiceManager().locateService
        ("com/ibm/websphere/bo/BOXMLSerializer");
ByteArrayInputStream input = new ByteArrayInputStream(<?xml?>..);

com.ibm.wsspi.al.ALContext.setContext
```

```
                         ("RALTemplateName", "ApplicationName");
          try {
             BOXMLDocument document = serializerService.readXMLDocument(input);
             DataObject dataObject = document.getDataObject();
          } finally {
             com.ibm.wsspi.al.ALContext.unset();
          }
```

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 Java EE roles. The context is used to check the caller's authorization for each call, even when administrative security is not set. If administrative 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("java:comp/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("java:comp/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 for business processes or human tasks 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 Platform, Enterprise Edition (Java EE) 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 Java EE roles. The context is used to check the caller's authorization for each call, even when administrative security is not set. If administrative 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("java:comp/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("java:comp/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);
```

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

Note: If process administration is restricted to system administrators, then instance-based administration is disabled. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. In addition, reading, viewing, and monitoring a process instance or parts of it can only be performed by users in the BPESystemAdministrator or BPESystemMonitor roles. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

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

Action	Caller's principal role				
	Reader	Editor	Potential owner	Owner	Administrator
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
transferWorkItem				x To potential owners or administrators only	x

Note: If process administration is restricted to system administrators, then instance-based administration is disabled. This means that administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. In addition, reading, viewing, and monitoring a process instance or parts of it can only be performed by users in the BPESystemAdministrator or BPESystemMonitor roles. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

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.EXECUTION_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.EXECUTION_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

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.

To suspend a process instance, it must be in the running or failing state. The caller must be a process administrator or a system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only callers in the BPESystemAdministrator role can perform this action.

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. In this state, activities that are started can still be finished but no new activities are activated. Subprocesses with the autonomy attribute set to `child` are also suspended if they are in the running, failing, terminating, or compensating state. Inline tasks and stand-alone tasks that are associated with this process instance are not suspended.

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.

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.

The caller must be a process administrator or a system administrator. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only callers in the BPESystemAdministrator role can perform this action.

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, a top-level process instance that in an unrecoverable state must be
terminated.

About this task

To perform this action, the caller must be a process administrator or a system
administrator. However, if Business Flow Manager is using the alternate process
administration authorization mode, which restricts process administration to
system administrators, then only callers in the BPESystemAdministrator role can
perform this action.

Because a process instance terminates immediately, without waiting for any
outstanding subprocesses or activities, you should only take this action 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, myFault,(String)faultNames.get(0) );
```

   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.

Related concepts

"Continue-on-error behavior of activities and business processes" on page 37
When you define a business process, you can specify what should happen if an unexpected fault is raised and a fault handler is not defined for that fault. You can use the **Continue On Error** setting when you define your process to specify that it is to stop where the fault occurs.

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 initiateAndClaimFirst and completeAndClaimSuccessor APIs support the processing of this type of workflow. This example shows the implementation of a single person workflow using a client-side page flow.

About this task

A single person workflow is also referred to as a *page flow* or a *screen flow*. There are two kinds of page flows:

- Client-side page flows, where the navigation between the different pages is realized using client-side technology, such as a multi-page Lotus® Forms form.
- Server-side page flows are realized using a business process and a set of human tasks that are modeled so that subsequent tasks are assigned to the same person.

Server-side page flows are more powerful than client-side page flows, but they consume more server resources to process them. Therefore, consider using this type of workflow primarily in the following situations:

- You need to invoke services between steps carried out in a user interface, for example, to retrieve or update data.
- You have auditing requirements that require CEI events to be written after a user interface interaction completes.

A typical example of a single person workflow is the ordering process in an online bookstore, where 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 starting an order process, and claiming the next human task activity.

The initiateAndClaimFirst API starts the page flow, that is, it starts the specified process and claims the first human task activity in the sequence of activities. It returns information about the claimed activity, including the input message to be worked on.

The completeAndClaimSuccessor API then completes the 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. Start the book ordering process and claim the first activity in the sequence of activities. 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.

   a. Retrieve the process template to create an input message of the appropriate type.

   ```
   ProcessTemplateData template = process.getProcessTemplate("CustomerOrder");
   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");
   }
   ```

   b. Start the process and claim the first human task activity.

   ```
   InitiateAndClaimFirstResult result =
     process.initiateAndClaimFirst("CustomerOrder", "MyOrderProcess", input);
   AIID aiid = result.getAIID();
   ClientObjectWrapper input = result.getInputMessage();
   ```

   When the first activity is claimed, the input message and the ID of the claimed 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.

Related tasks

"Processing a single person workflow that includes human tasks" on page 587
Some workflows are performed by only one person, for example, ordering books
from an online bookstore. This example shows how to implement a single person
workflow using a server-side page flow. Both the Business Flow Manager and the
Human Task Manager APIs are used to process the workflow.

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

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.

About this task

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.

Related concepts

"Continue-on-error behavior of activities and business processes" on page 37
When you define a business process, you can specify what should happen if an
unexpected fault is raised and a fault handler is not defined for that fault. You can
use the **Continue On Error** setting when you define your process to specify that it
is to stop where the fault occurs.

Retrying the execution of a stopped activity

If an activity in a long-running process encounters an uncaught fault in the
enclosing scope and if the associated activity template specifies that the activity
stops when an error occurs, the activity is put into the stopped state so that it can
be repaired. You can retry the execution of the activity.

About this task

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.

When you define a business process, you can specify what should happen if an unexpected fault is raised and a fault handler is not defined for that fault. You can use the **Continue On Error** setting when you define your process to specify that it is to stop where the fault occurs.

Repairing activities that stopped because a join, loop, or counter evaluation failed

Activities can stop because an exception occurred when a join or loop condition, or a forEach counter value was evaluated. The administrator decides not to retry the execution of the activity, for example, because the evaluation might fail again. In such cases, the correct values for the expression can be supplied using the Business Process Choreographer EJB API so that the navigation of the process can continue.

About this task

You can set the value of a join condition for any type of activity, the value of a loop condition of a while or repeat-until activity. You can also set the values of start and final counters, and the maximum number of completed branches for a forEach activity. The value that you set for the completed branches depends on the definition of the forEach activity in the process model. If an early exit condition is specified in the model, set a value for the maximum completed branches. If an early exit condition is not specified, set the value of the maximum completed branches to null.

The following sample shows how to set the value of a loop condition.

Procedure

1. List the activities that stopped because the evaluation of a loop condition failed.

   ```
   QueryResultSet result = process.query(
     "DISTINCT ACTIVITY.AIID",
     "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
      ACTIVITY.STOP_REASON = ACTIVITY.STOP_REASON.STOP_REASON_IMPLEMENTATION_FAILED AND
      (ACTIVITY.KIND = ACTIVITY.KIND.KIND_WHILE OR
          ACTIVITY.KIND = ACTIVITY.KIND.KIND_REPEAT_UNTIL) AND
      PROCESS_INSTANCE.NAME='CustomerOrder'",
     (String)null, (Integer)null, (TimeZone)null );
   ```

   Similarly, you can list the activities that stopped because the evaluation of a join condition or a forEach counter failed.

   - For a failed join condition, use the following expression:

     ```
     ACTIVITY.STOP_REASON.STOP_REASON_ACTIVATION_FAILED
     ```

   - For a failed forEach counter, use the following expression:

   ```
   ACTIVITY.STOP_REASON.STOP_REASON_IMPLEMENTATION_FAILED AND
   (ACTIVITY.KIND = ACTIVITY.KIND.KIND_FOR_EACH_SERIAL OR
       ACTIVITY.KIND = ACTIVITY.KIND.KIND_FOR_EACH_PARALLEL)
   ```

   This action returns the activities for the CustomerOrder process instance that stopped because the evaluation of a loop condition failed.

2. Provide the value of the loop condition, for example, true.

   ```
   if (result.size() > 0)
   {
     result.first();
     AIID aiid = (AIID) result.getOID(1);

     process.forceLoopCondition(aiid, true);
   }
   ```

This action sets the value of the loop condition for the activity to true and the navigation of the process instance continues.

Similarly, you can set the value of a join condition (process.forceJoinCondition(aiid, true);) or the values of forEach activity counters (process.forceForEachCounterValues(aiid, 1, 5, new Integer(2));).

Updating correlation sets associated with stopped activities

Correlation sets are used to support stateful collaboration between Web services. In such cases, the correct values for the expression can be supplied using the Business Process Choreographer EJB API so that the navigation of the process can continue.

About this task

An activity that is in a stopped state can require an update of its associated correlation set for one of the following reasons:

- An exception occurred when the correlation set was evaluated. The correlation set is to be initialized but it is already initialized.
- An exception occurred when the correlation set was evaluated. The correlation set is to not be initialized but its values are not set. This can occur, for example, because an initializing activity was skipped.
- The activity needs to be retried. If the correlation set is initialized by the activity, then it can be unitialized or changed before the forceRetry method is called.
- The activity needs to be completed. If the correlation set is initialized by the activity, then it can be unitialized or changed before the forceComplete method is called.

You can retrieve the correlation set instances of a process or activity instance. The following example shows how to initialize or unitialize correlation set instances.

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. Retrieve the correlation set instances that are defined for the activity.

   ```
   AIID aiid = null;

   List correlationSet = null;

   if (result.size() > 0)
   {
      result.first();
      AIID aiid = (AIID) result.getOID(1);

      ActivityInstanceData activity = process.getActivityInstance(aiid);

      correlationSet = process.getCorrelationSetInstances
                   (aiid, activity.getInputMessageTypeName());
   }
   ```

3. Unitialize the correlation set, for example, MyCorrelationSet.

   ```
   for ( int i=0; i<correlationSet.size(); i++ )
   {
     CorrelationSetInstanceData correlationSetInstance =
               (CorrelationSetInstanceData)correlationSet.get(i);
   ```

Developing EJB client applications **563**

```
    if ( correlationSetInstance.isInitialized() &&
        correlationSetInstance.getCorrelationSetName().equals("MyCorrelationSet") )
    {
      process.uninitializeCorrelationSet
        ( activity.getProcessInstanceID(), correlSetInstance.getCorrelationSetName() );
    }
  }
```

This action uninitializes the correlation set MyCorrelationSet.

4. Initialize the correlation set, for example, MyCorrelationSet. In this example, a string-valued property of the correlation set is set.

```
for ( int i=0; i<correlationSet.size(); i++ )
{
  CorrelationSetInstanceData correlationSetInstance =
            (CorrelationSetInstanceData)correlationSet.get(i);

  if ( correlationSetInstance.getCorrelationSetName().equals("MyCorrelationSet") )
  {
    List correlationSetProperties =
                  correlationSetInstance.getCorrelationSetProperties();
    for ( int j=0; j<correlationSetProperties.size(); j++ )
    {
      CorrelationPropertyInstanceData property =
              (CorrelationPropertyInstanceData)correlationSetProperties.get(j);

      if ( property.getPropertyName().equals("MyProperty") )
      {
        property.setValue("NewValue");

        process.initializeCorrelationSet
          ( activity.getProcessInstanceID(), correlationSetInstance );
      }
    }
  }
}
```

This action initializes the string-valued property MyProperty in the correlation set MyCorrelationSet.

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 65. API methods for process templates

Method	Description
getProcessTemplate	Retrieves the specified process template.

Table 65. API methods for process templates (continued)

Method	Description
queryProcessTemplates	Retrieves process templates that are stored in the database.

Process instances

The following API methods are related to starting process instances.

Table 66. API methods that 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.
initiateAndSuspend	Creates a process instance but immediately suspends the further processing of the process instance.
initiateAndClaimFirst	Creates a process instance and claims the first inline human task.
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 67. 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.

Method	Description
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.
queryEntities	Uses query tables to retrieve the properties from the database that match the search criteria.
getWaitingActivities	Returns information about the activities that are waiting for a message so that the processing of these activities can continue.
migrate	Migrates a process instance to the specified newer version of its process model.

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 68. 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 the following: • An activity instance that is in the running or stopped state. • A human task activity that is in the state ready or claimed. • A wait activity in state waiting.

Method	Description
forceRetry	Forces the repetition of the following: • An activity instance that is in the running or stopped state. • A human task activity that is in the state ready or claimed.
forceNavigate, forceForEach, forceLoop, forceJoin	These methods force the navigation of a stopped activity.
skip	Skips processing of the activity.
jump	Jumps from one activity to the other.
query	Retrieves the properties from the database that match the search criteria.
queryEntities	Uses query tables to retrieve 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 69. 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 task template ID (TKTID) of the collaboration task you want to run.

   This step is optional if you already know the task template ID.

```
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.

Related concepts

"State transition diagrams for collaboration tasks" on page 90
Collaboration tasks support people when they perform work for other people. During the life cycle of a collaboration task, certain interactions are possible only in certain task states, and these interactions, in turn, influence the state of the task.

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.

Procedure

- 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");
}
```

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.

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 70. API methods for task templates

Method	Description
getTaskTemplate	Retrieves the specified task template.
createTask	Creates a task instance from 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.
createAndStartTaskAsSubtask	Creates and starts a task instance as a subtask of the specified task.
createInputMessage	Creates an input message for the specified task template. For example, create a message that can be used to start a task.

Table 70. API methods for task templates (continued)

Method	Description
queryTaskTemplates	Retrieves task templates that are stored in the database.

Task instances

The following methods are available to work with task instances.

Table 71. 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.
startTaskAsSubtask	Starts a task as a subtask of the task instance.
suspend	Suspends the collaboration or to-do task.
resume	Resumes the collaboration or to-do task.
restart	Restarts the task instance.
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.
completeWithFollowOnTask	Completes the task instance and starts a follow-on task.
cancelClaim	Releases a claimed task instance so that it can be worked on by another potential owner.
createWorkItem	Creates a work item for the task instance.
transferWorkItem	Transfers the work item to a specified owner.
deleteWorkItem	Deletes the work item.

Escalations

The following methods are available to work with escalations.

Table 72. API methods for working with escalations

Method	Description
getEscalation	Retrieves the specified escalation instance.
triggerEscalation	Manually triggers an escalation.

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 73. 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.

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. For example, 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. However, if Business Flow Manager is using the alternate process administration authorization mode, which restricts process administration to system administrators, then only users in the BPESystemAdministrator role can perform this action. Therefore, to get a complete list of templates, you also need to 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 a single person workflow using a server-side page flow. Both the Business Flow Manager and the Human Task Manager APIs are used to process the workflow.

About this task

A single person workflow is also referred to as a *page flow* or a *screen flow*. There are two kinds of page flows:

- Client-side page flows, where the navigation between the different pages is realized using client-side technology, such as a multi-page Lotus Forms form.
- Server-side page flows are realized using a business process and a set of human tasks that are modeled so that subsequent tasks are assigned to the same person.

Server-side page flows are more powerful than client-side page flows, but they consume more server resources to process them. Therefore, consider using this type of workflow primarily in the following situations:

- You need to invoke services between steps carried out in a user interface, for example, to retrieve or update data.
- You have auditing requirements that require CEI events to be written after a user interface interaction completes.

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.

Related tasks

"Processing a single person workflow" on page 555
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 initiateAndClaimFirst and completeAndClaimSuccessor APIs support the processing of this type of workflow. This example shows the implementation of a single person workflow using a client-side page flow.

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.

Related concepts

"Fault handling in business processes" on page 34
When a fault occurs in a process, the navigation continues with the fault handler or fault link.

Handling Business Process Choreographer EJB API exceptions

If a method in the BusinessFlowManagerService interface or the HumanTaskManagerService interface does not complete successfully, an exception is thrown that denotes the cause of the error. You can handle this exception specifically to provide guidance to the caller.

About this task

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. 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 a human task activity

When a human task activity is processed, it can complete successfully. In this case, you can pass an output message. If the human task activity does not complete successfully, you can pass a fault message.

About this task

You can read the fault message to determine the cause of the error.

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

In a well-designed process, exceptions and faults are usually handled by fault handlers. You can retrieve information about the exception or fault that occurred for an invoke activity from the activity instance.

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();
  }
}
```

Checking which unhandled exception or fault occurred for a failed process instance

In a well-designed process, exceptions and faults are usually handled by a fault handler. If the process implements a two-way operation, you can retrieve information about a fault or handled exception from the fault name property of the process instance object. For faults, you can also retrieve the corresponding fault message using the getFaultMessage API.

About this task

If a process instance fails because of an exception that is not handled by any fault handler, you can retrieve information about the unhandled exception from the process instance object. By contrast, if a fault is caught by a fault handler, then information about the fault is not available. You can, however, retrieve the fault name and message and return to the caller by using a FaultReplyException exception.

Procedure

1. List the process instances that are in the failed state.

```
QueryResultSet result =
     process.query("PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                       PROCESS_INSTANCE.STATE.STATE_FAILED",
                   (String)null, (Integer)null, (TimeZone)null);
```

This action returns a query result set that contains the failed process instances.

2. Read the information for the unhandled exception.

```
if (result.size() > 0)
{
  result.first();
  PIID piid = (PIID) result.getOID(1);
  ProcessInstanceData pInstance = process.getProcessInstance(piid);

  ProcessException excp = pInstance.getUnhandledException();
  if ( excp instanceof RuntimeFaultException )
  {
   RuntimeFaultException xcp = (RuntimeFaultException)excp;
   Throwable cause = xcp.getRootCause();
  }
  else if ( excp instanceof StandardFaultException )
  {
   StandardFaultException xcp = (StandardFaultException)excp;
   String faultName = xcp.getFaultName();
  }
  else if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException xcp = (ApplicationFaultException)excp;
   String faultName = xcp.getFaultName();
  }
}
```

Results

Use this information to look up the fault name or the root cause of the problem.

Developing Web services API client applications for business processes and human tasks

You can develop client applications that access business process applications and human task applications through the Business Process Choreographer Web services APIs. The client application development process consists of a number of mandatory and optional steps, including generating a Web service proxy and adding security and transaction policies to the client application.

About this task

Beginning with Version 7, the JAX-WS-based Web services API replaces the JAX-RPC-based Business Process Choreographer Web services API in Version 6 (first published in release 6.0.2). The JAX-RPC-based Business Process Choreographer Web services API will be deprecated, such that new Web service client applications should be implemented using the JAX-WS-based API.

Note: The Business Process Choreographer Java Message Service (JMS) API is still using the WSDL and XML Schema definitions for Version 6.

You can develop client applications in any Web services client environment. The following steps provide an overview of the actions you need to take to develop such an application.

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.
3. In your client application development environment, generate a *Web service proxy* using the exported artifacts.
4. Develop the code for your client application.
5. Add any necessary security or transaction policies to your client application.

Web service components and sequence of control

In Web services applications, 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 Web service proxy (also provided by the user, but which can be automatically generated using client-side utilities) wraps the service request in a SOAP request envelope and 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 Web service proxy extracts the data from the SOAP response and passes it to the client application.

c. The client application processes the returned data as necessary.

Example

The following is a possible outline for a client application that accesses the Human Task Manager Web services API to process a to-do task:

1. The client application issues a query Web service call to the WebSphere Process Server requesting a list of to-do tasks to be worked on by a user.

2. The WebSphere Process Server returns the list of to-do tasks.

3. The client application then issues a claim Web service call to claim one of the to-do tasks.

4. The WebSphere Process Server returns the input message for the task.

5. The client application issues a complete Web service call to complete the task with an output or fault message.

Web service API 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:

- 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 Web Services (JAX-WS 2.0) specification. This is the default style for all business processes and human tasks developed with WebSphere Integration Developer.

- Do not use the maxOccurs attribute in parameter elements of the operations, or ensure that the value of this attribute is set to the default value, `maxOccurs="1"`.

- Fault messages that are 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"/>
```

JAX-WS-based Business Process Choreographer Web services APIs

Beginning with Version 7, the JAX-WS-based Web services API replaces the JAX-RPC-based Business Process Choreographer Web services API in Version 6 (first published in release 6.0.2). Two Business Process Choreographer Web services interfaces are provided, one for business processes and one for human tasks, each with their own file artifacts and XML definition namespaces.

The following table provides an overview of the file artifacts and XML definition namespaces for the JAX-WS-based Web services.

Table 74. File artifacts and XML definition namespaces for the JAX-WS-based Web services

Business Process Choreographer Web services interface	JAX-WS Web services file artifact	JAX-WS Web services XML namespace
Business Flow Manager Web Service	BFMJAXWSService.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0/Binding
Business Flow Manager Web Service Interface	BFMJAXWSInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0
Business Flow Manager Web Service Data Types	BFMDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/business-process/types/7.0
Business Flow Manager Callback Web Service	BFMJAXWSCallbackService.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/callback-services/7.0/Binding
Business Flow Manager Callback Web Service Interface	BFMJAXWSCallbackInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/callback-services/7.0
Human Task Manager Web Service	HTMJAXWSService.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/services/7.0/Binding
Human Task Manager Web Service Interface	HTMJAXWSInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/services/7.0
Human Task Manager Web Service Data Types	HTMDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/human-task/types/7.0
Human Task Manager Callback Web Service	HTMJAXWSCallbackService.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/callback-services/7.0/Binding
Human Task Manager Callback Web Service Interface	HTMJAXWSCallbackInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/callback-services/7.0
Common Business Process Choreographer Data Types	BPCDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/7.0

Business Process Choreographer Web services API: Standards

Use the following links to find relevant supplemental information about the standards that apply to Web applications. The information resides on non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to IBM WebSphere Process Server, but is useful for understanding Web services in general.

* Java API for XML-based Web Services (JAX-WS 2.0) (JSR-224; Java Community Process)
* Java Architecture for XML Binding (JAXB) 2.0 (JSR-222; Java Community Process)
* Web Services Description Language (WSDL) 1.1 (W3C)

- XML Schema Part 0: Primer Second Edition (W3C)
- XML Schema Part 1: Structures Second Edition (W3C)
- XML Schema Part 2: Datatypes Second Edition (W3C)
- Simple Object Access Protocol (SOAP) 1.1 (W3C)
- Web Services Policy Framework (WS-Policy) 1.5 (W3C)
- WS-Security 1.1 (OASIS)
- WS-Security UserName Token Profile 1.1 (OASIS)
- WS-AtomicTransaction 1.2 (OASIS)
- WS-Interoperability Basic Profile 1.1 (WS-Interoperability Organization)

Publishing and exporting artifacts from the server environment for Web services client applications

Before you can develop client applications to access the Business Process Choreographer 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 Web service endpoint, the port types and operations that make up the Business Process Choreographer Web services API (always required for the Web service proxy generation).
- XML Schema Definition (XSD) files containing definitions of data types referenced by services in the Business Process Choreographer WSDL files (always required for the Web service proxy generation).
- Your own WSDL and XSD files describing interfaces and data types for your 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 your business processes or human tasks through the Web services APIs. They are not necessary if your client application is only going to invoke operations that can be fulfilled by Business Process Choreographer without direct interaction with your process or task instances, such as issuing queries.
- Web Service Policy (WS-Policy) files describing the quality of service attributes for the Web services API. They may be exported in order to serve as a base for creating client-side Web service policies.

  WS-Security
  The request message must contain either a UserName token or an LPTA token.

  WS-Transaction
  The request message can contain a WS-AtomicTransaction context. If this context is present, the request is processed in the transaction scope of the caller.

After these artifacts are published, you need to copy them to your client programming environment, where they are used to generate a Web service proxy and helper classes.

Publishing Business Process Choreographer 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. These files are used to generate a Web service proxy for your application.

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 must publish these WSDL files, and any XSD files referenced by the WSDL files. You can then copy them from the WebSphere environment to your development environment, where they are used to generate a Web service proxy and helper classes. You only need to publish the Business Process Choreographer WSDL files once.

Publishing the business process WSDL file for Web services applications

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. Click **Applications** → **SCA modules**.

   Note: You can also click **Applications** → **Application Types** → **WebSphere 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 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_nodename_servername_WSDLFiles.zip`. The .zip file contains a WSDL file that describes the Web services, and any XSD files that are referenced by the WSDL file.

Note: The exported .zip file contains WSDL and XSD artifacts of both the JAX-WS Web service introduced in Version 7 and the JAX-RPC Web service used in Version 6. When you generate a the Web service proxy using the wsimport tool, you select the JAX-WS Web service artifacts and the JAX-RPC artifacts are ignored.

Publishing the human task WSDL file for Web services applications

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. Click **Applications** → **SCA modules**.

   Note: You can also click **Applications** → **Application Types** → **WebSphere 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 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_nodename_servername_WSDLFiles.zip`. The .zip file contains a WSDL file that describes the Web services, and any XSD files that are referenced by the WSDL file.

Note: The exported .zip file contains WSDL and XSD artifacts of both the JAX-WS Web service introduced in Version 7 and the JAX-RPC Web service used in Version 6. When you generate a the Web service proxy using wsimport tool, you select the JAX-WS Web service artifacts and the JAX-RPC artifacts are ignored.

Exporting WSDL and XSD files for business process and human task Web services applications

Business processes and human tasks have well-defined interfaces that allow them to be accessed externally as Web services. You need to export the WSDL interface definitions and the XML Schema data type definitions to your client programming environment.

About this task

This procedure must be repeated for each business process or human task that your client application needs to interact with.

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, and importing the data type definitions into your client programming environment.

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 `UserData` parameters in all *generic* operations (call, sendMessage, and so on) performed by the client application.

The customer-defined elements are created, serialized and deserialized on the client application side using classes generated using the exported XSD files. The generation of these classes is part of the Web service proxy generation where the exported WSDL and XSD files are included.

The generic operations of the Web service interface propagate the document wrapper element to and from the operation that is implemented by the business process or human task. For the sample operation in the previous example, a Web service SOAP message might look as follows:

```
<soapenv:Envelope xmlns:soapenv="..." ...>
  <soapenv:Header>
    ...
  </soapenv:Header>
  <soapenv:Body>
    <bfm:sendMessage
        xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0">
      <processTemplateName>customerProcessTemplate</processTemplateName>
      <portType xmlns:cns="http://example.com/customerProcess">cns:customerProcessPortType</portType>
      <operation>updateCustomer</operation>
      <input>
        <cns:updateCustomer xmlns:cns="http://example.com/customerProcess">
          <street>1600 Pennsylvania Avenue Northwest</street>
          <city>Washington, DC 20006</city>
        </cns:updateCustomer>
      </input>
    </bfm:sendMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

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 Business Process Choreographer Web services APIs.

Generating a Web service proxy (Java Web services)

Java Web services client applications use a *Web service proxy* to interact with the Business Process Choreographer Web services APIs.

About this task

A Web service proxy for Java Web services contains a number of JavaBeans classes that the client application calls to perform Web service requests. The Web service proxy 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 Web service proxy allows a client application to call a Web service as if it were a local function.

Note: You only need to generate a Web service proxy once. All client applications accessing the same Web services API can then use the same Web service proxy.

In the IBM Web services environment, you can generate a Web service proxy in one of the following ways.
- Use Rational® Application Developer or WebSphere Integration Developer integrated development environments.
- Use the wsimport command-line tool.

Other Java Web services development environments usually include either the wsimport tool or proprietary client application generation facilities.

Using Rational Application Developer to generate a Web service proxy for a Web services application

You can use the Rational Application Developer integrated development environment to generate a Web service proxy for your Web services client application. The following sequence of steps applies to Rational Application Developer Version 7.5.3.

Before you begin

Before generating a Web service proxy, you must have previously exported the WSDL and XSD files that describe the business process or human task Web services interfaces from the WebSphere environment 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. Do not change the contents of this directory, and be aware that only the following WSDL and XSD files are used for the Web service proxy generation for interactions with business processes:
        - BFMJAXWSService.wsdl
        - BFMJAXWSInterface.wsdl
        - BFMJAXWSCallbackService.wsdl
        - BFMJAXWSCallbackInterface.wsdl
        - BFMDataTypes.xsd
        - BPCDataTypes.xsd
        - wsa.xsd
     b. Import the subdirectory `META-INF` from the unzipped directory `BPEContainer_nodename_servername.ear/bfmjaxws.jar`.
   - For human tasks:
     a. Unzip the exported file `TaskContainer_nodename_servername_WSDLFiles.zip` to a temporary directory. Do not change the contents of this directory, and be aware that only the following WSDL and XSD files are used for the Web service proxy generation for interactions with human tasks:
        - HTMJAXWSService.wsdl
        - HTMJAXWSInterface.wsdl
        - HTMJAXWSCallbackService.wsdl
        - HTMJAXWSCallbackInterface.wsdl
        - HTMDataTypes.xsd
        - BPCDataTypes.xsd
        - wsa.xsd
     b. Import the subdirectory `META-INF` from the unzipped directory `TaskContainer_nodename_servername.ear/htmjaxws.jar`.

   A new `wsdl` directory and subdirectory structure are created in your project.
2. Select the `BFMJAXWSService.wsdl` file located in the newly-created `wsdl` directory.
3. Right-click and choose **Web services** ▸ **Generate client**.

Before continuing with the remaining steps, ensure that the server has started.

4. On the Web Services window, click **Next** to accept all defaults.

5. On the Web Service JAX-WS Web Service Client Configuration window, change the Version of JAX-WS code to be generated to 2.0 and click **Finish** to accept all other defaults

6. Redo steps 2 to 5 of this procedure with `HTMJAXWSService.wsdl` and overwrite all files if you are prompted.

Results

A Web service proxy, made up of a number of proxy, locator, and JAXB classes, is generated and added to your project.

Using the wsimport command-line tool to generate a Web service proxy for a Web services application

You can use the wsimport command-line tool to generate a Web service proxy for a Web services application.

Before you begin

Before generating a Web service proxy, you must have previously exported the WSDL files that describe the business process or human task Web services APIs from the WebSphere environment, and copied them to your client programming environment.

Procedure

1. Generate a Web service proxy for the Business Process Choreographer Web services API:

   Note: For a detailed description of the wsimport command-line tool for JAX-WS applications, see the WebSphere Application Server wsimport command-line tool documentation.

   ```
   wsimport.bat BFMJAXWSService.wsdl myService1.wsdl myService2.wsdl
    –d proxy-bfm
    -wsdllocation <bfm_location>

   wsimport.bat HTMJAXWSService.wsdl myService1.wsdl myService2.wsdl
    –d proxy-htm
    -wsdllocation <htm_location>
   ```

   In this example, `myService1.wsdl` and `myService2.wsdl` contain interface definitions of custom business processes, or human tasks, or both. In addition, *<bfm_location>* and *<htm_location>* can be obtained from the WSDL *<port>* element in `BFMJAXWSService.wsdl` and `HTMJAXWSService.wsdl`, respectively.

   You can merge both proxies into one common directory (for example, proxy-bpc) and overwrite existing files if you are prompted.

2. Include the generated class files in your project.

Related tasks

"Creating a client application for business processes and human tasks (Java Web services)"
A client application sends requests to and receives responses from the Business Process Choreographer Web services APIs. By using a Web service proxy 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 a client application for business processes and human tasks (Java Web services)

A client application sends requests to and receives responses from the Business Process Choreographer Web services APIs. By using a Web service proxy 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 Web service proxy.

About this task

You can develop client applications using any Web services-compatible development tool, for example IBM Rational Application Developer. 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 Web service proxy.
3. Code your client application.
4. Build the project.
5. Run the client application.

Example

The following example shows how to use the Business Flow Manager Web service API.

```
try {
   // create bfm proxy
   BFMJAXWSPortType bfm = new BFMJAXWSService().getBFMJAXWSPort();

   // call getProcessTemplate
   ProcessTemplateType ptt =
     bfm.getProcessTemplate("MY_PROCESS_TEMPLATE_NAME");

   // handle return value
   System.out.println("Process template '" + ptt.getName() +
       "' found, details following:");
   System.out.println("Execution mode: " +
       ptt.getExecutionMode());
   System.out.println("Schema version: " +
       ptt.getSchemaVersion());
 } catch (Exception e) {
   if ( e instanceof ProcessFaultMsg )
   {
     ProcessFaultMsg pfm = (ProcessFaultMsg) e;
     List<FaultStackType> list =
         ( pfm.getFaultInfo() ).getFaultStack();
     FaultStackType fault = list.get( 0 );
     System.out.println( "ProcessFaultMessage: " +
         fault.getMessage() );
   }
```

```
        else
        {
            e.printStackTrace( System.out );
        }
    }
```

Related tasks

"Using the wsimport command-line tool to generate a Web service proxy for a Web
services application" on page 602
You can use the wsimport command-line tool to generate a Web service proxy for a
Web services application.

"Generating a Web service proxy (Java Web services)" on page 600
Java Web services client applications use a *Web service proxy* to interact with the
Business Process Choreographer Web services APIs.

Adding security

The Business Process Choreographer Web service requires that you configure your
client application for an authentication mechanism.

About this task

By default, Business Process Choreographer supports the following authentication
mechanisms:

Username Token
>   A web service consumer supplies a Username token as a means of
>   identifying the requestor by "username", and optionally using a password
>   to authenticate that identity to the web service provider.

Binary Security Token – Lightweight Third-Party Authentication (LTPA) Token
>   A web service consumer supplies an LTPA token as a means of
>   authenticating the requestor to the web service provider

You can replace the Business Process Choreographer Web service security policy by
an alternative authentication mechanism. However, it is not possible to invoke
Business Process Choreographer Web service operations as an unauthenticated
user, so one authentication mechanism is always required.

Adding transaction support

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

Business Process Choreographer runs each Web service operation request as a
separate global transaction. Client applications can be configured to use transaction
support in one of the following ways:

*   Propagate the client's transaction context. Server-side request processing is
    performed within the client application transaction context and therefore
    committed (or backed out) together with the client's transaction. Conversely, if
    the server encounters a problem while the Web service operation is running and
    requests a rollback, the client application's transaction is also rolled back.

- Not use transaction support. Business Process Choreographer creates a new global transaction in which to run the request, but server-side request processing is not performed with the client application transaction context

The Web service policy attached to the Business Process Choreographer Web service allows that each request message may contain a WS-AT transaction context as described above. If you choose to invoke the Web service operations without passing a client transaction context, it is safe to ignore the provider-side transaction policy and configure the Web service client without a transaction policy.

Developing client applications using the Business Process Choreographer JMS API

You can develop client applications that access business process applications asynchronously through the Java Messaging Service (JMS) API.

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.

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?

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>
```

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).

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-acknowledgment.
   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

Operations exposed by business processes or human tasks must comply with the document/literal wrapper style. The JMS message body is a String containing an XML document that represents the document/literal wrapper element of the operation. The generic operations of the JMS message interface propagate the document-wrapper element to and from the operation that is implemented by the business process or human task.

The following example shows a simple valid request message body:

```
<bfm:queryProcessTemplates
    xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0">
  <whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</bfm:queryProcessTemplates>
```

The following example shows a more complex, valid request message body. The client application has a sendMessage API operation for submitting a message to a specific process. The process input message is one of the API parameters; this message is the input message of a business operation exposed by a customer process. The process contains a receive activity that consumes the message.

The `bfm:sendMessage` element is the document wrapper element of the JMS API operation. It includes the `cns:updateCustomer` element, which is the document wrapper element for the operation that is implemented by the process. This process has, for example, a `bpel:receive` activity that references the `cns:customerProcessPortType` WSDL port type, and the `updateCustomer` WSDL operation.

```
<bfm:sendMessage
    xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0">
  <processTemplateName>customerProcessTemplate</processTemplateName>
  <portType xmlns:cns="http://example.com/customerProcess">cns:customerProcessPortType</portType>
  <operation>updateCustomer</operation>
  <cns:updateCustomer xmlns:cns="http://example.com/customerProcess">
    <street>1600 Pennsylvania Avenue Northwest</street>
    <city>Washington, DC 20006</city>
  </cns:updateCustomer>
</bfm:sendMessage>
```

Related tasks

"Checking the response message for business exceptions" on page 612
JMS client applications must check the message header of all response messages for business exceptions.

Copying artifacts for JMS client applications

A number of artifacts can be copied from the WebSphere Process Server environment to help in the creation of JMS client applications.

About this task

These artifacts are mandatory only if you use the BOXMLSerializer to create the JMS message body. For the JMS API, these artifacts are:

    BFMIF.wsdl

    BFMIF.xsd

    BPCGen.xsd

    wsa.xsd

You must publish and export these files from the WebSphere Process Server environment to your development environment.

Publishing the business process WSDL file for JMS applications

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. Click **Applications** → **SCA modules**.

   Note: You can also click **Applications** → **Application Types** → **WebSphere 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 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, and any XSD files that are referenced by the WSDL file.

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:

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
}
```

The header and body of each JMS message must have a predefined structure.

Example: executing a long running process using the Business Process Choreographer JMS API

This example shows how to create a generic client application that uses the JMS API to work with long-running processes.

Procedure

1. Set up the JMS environment, as described in "Accessing the JMS interface" on page 608.
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 might require the use of predeployed, process-specific artifacts.
5. Create a process instance.
   - Issue a `sendMessage`.

   With the JMS API, you can 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. Optional: Obtain output messages from the process instances by repeating the following steps:
   a. Issue `query` to obtain the finished state of the process instance.
   b. Issue `getOutputMessage` .
7. Optional: Work with additional operations exposed by the process:
   a. Issue `getWaitingActivities` or `getActiveEventHandlers` to obtain a list of InboundOperationTemplate objects.
   b. Create input messages.
   c. Send messages with `sendMessage`.
8. Optional: Get and set custom properties that are defined on the process or contained activities with `getCustomProperties` and `setCustomProperties`.
9. Finish working with a process instance:
   a. Send `delete` and `terminate` to finish working with the long-running process.

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. For applications that include human tasks, you can generate a JSF custom client. For more information on generating a JSF client, go to the information center for WebSphere Integration Developer.

You can also develop your Web client using the JSF components provided by Business Process Choreographer.

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

   In WebSphere Process Server, all of these files are in the following directory:

   - **Windows** On Windows platforms: *install_root*`\ProcessChoreographer\`
     `client`

   - **Linux** **UNIX** On Linuxand UNIX platforms: *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. Ensure that you use `BPCError` for the name of the error bean and `error` for the name of the navigation target of the error page.

```
<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 75. 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.

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.
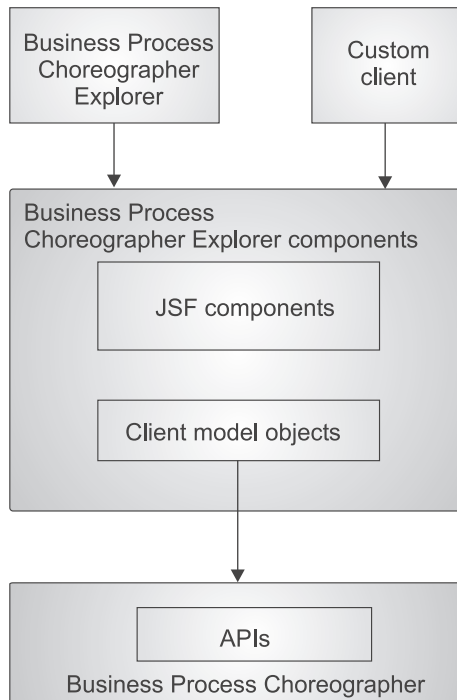
JSF components

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();
}
```

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 76. 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,

Developing Web applications using JSF components   **621**

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.
    //
    // Add the database columns for all of the properties you want to show
    // in your list to the select statement
    //
    QueryResultSet queryResult = taskService.query(
     "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
      + "TASK.STARTER, TASK.OWNER, 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.

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 run

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 619
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 77. 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.
styleClass	no	The CSS style class for rendering the overall table containing titles, rows, and paging buttons.

Table 78. 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().

Table 78. bpe:column attributes (continued)

Attribute	Required	Description
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.

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 79. 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.

Table 79. bpe:details attributes (continued)

Attribute	Required	Description
styleClass	no	The CSS class that is used for rendering the HTML element.

Table 80. 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 JavaServer Faces (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 use neither the BPCListHandler class nor the BPCDetailsHandler class for the model, you must refer to another object that has a class that implements the ItemProvider interface.

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. Optional: 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. Optional: 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.

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 81. 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 82. 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.
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.

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 83. bpe:form attributes

Attribute	Required	Description
id	no	The JavaServer Faces ID of the component.

Table 83. bpe:form attributes (continued)

Attribute	Required	Description
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.
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.

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>
```

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 post processing of people query results.

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.

Before you can use the plug-ins, you must install and register them. You can register the plug-in to post process people query results with the TaskContainer application. The plug-in is then available for all tasks.

Creating API event handlers for Business Process Choreographer

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 APIEventHandlerPlugin5 interface or extends the APIEventHandler implementation class. This class can invoke the methods of other classes.

   - If you use the APIEventHandlerPlugin5 interface, you must implement all of the methods of the APIEventHandlerPlugin5 interface and the APIEventHandlerPlugin interface.
   - If you extend the APIEventHandler implementation class, overwrite the methods that you need.

   This class runs in the context of a Java Platform, Enterprise Edition (Java EE) Enterprise application. Ensure that this class and its helper classes follow the EJB specification.

   Note: 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 might result in inconsistent task data in the database.

2. Assemble the plug-in class and its helper classes into a JAR file.

   You can make the JAR file available in one of the following ways:

   - As a utility JAR file in the application EAR file.
   - As a shared library that is installed with the application EAR file.

- As a shared library that is installed with the TaskContainer application. In this case, the plug-in is available for all tasks.

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 EE 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.APIEventHandlerPlugin5 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 (a line that starts with a number sign (#)) 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.

Notes: 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.

To make a change to an implementation effective, replace the JAR file in the shared library, deploy the associated EAR file again, and restart the server.

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.

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 and prevent processing from continuing.

To implement API event handlers, you can implement either the APIEventHandlerPlugin3 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 for Business Process Choreographer

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 implement 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 Platform, Enterprise Edition (Java EE) Enterprise 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.

   Note: 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 might result in inconsistent task data in the database.

2. Assemble the plug-in class and its helper classes into a JAR file.

   You can make the JAR file available in one of the following ways:
   - As a utility JAR file in the application EAR file.
   - As a shared library that is installed with the application EAR file.
   - As a shared library that is installed with the TaskContainer application. In this case, the plug-in is available for all tasks.

3. Assemble the plug-in class and its helper classes into a JAR file.

   If the helper classes are used by several Java EE applications, you can package these classes in a separate JAR file that you register as a shared library.

4. 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 EE 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 (a line that starts with a number sign (#)) 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.

Notes: 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.

To make a change to an implementation effective, replace the JAR file in the shared library, deploy the associated EAR file again, and restart the server.

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.

Installing API event handler and notification event handler plug-ins for human tasks

To use API event handler or notification event handler plug-ins, you must install the plug-in so that it can be accessed by the human 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 Platform, Enterprise Edition (Java EE) application, or several applications.

Complete one of the following steps to install a plug-in.

Procedure

- Install a plug-in for use by a single Java EE 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 Java EE application of the main enterprise JavaBeans (EJB) module.

- Install a plug-in for use by several Java EE 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, manually distribute the JAR file on each node that hosts a server or cluster member on which any of your applications is deployed. You can use the deployment target scope of your applications, that is the server or cluster on which the applications are deployed, or the cell scope. Be aware that the plug-in classes are then visible throughout the selected deployment scope.

What to do next

You can now register the plug-in.

Registering API event handler and notification event handler plug-ins with task templates, task models, and tasks

You can register plug-ins for API event handlers and notification event handlers with tasks, task templates, and task models at various times: when you create an ad hoc task, update an existing task, create an ad hoc task model, or define a task template.

About this task

You can register plug-ins for API event handlers and notification event handlers with tasks on the following levels:

Task template
All of the tasks that are created using the template use the same handlers

Ad hoc task model
The tasks that are created using the model use the same handlers

Ad hoc task
The task that is created uses the specified handlers

Existing task
The task uses the specified handlers

You can register a plug-in in one of the following ways.

Procedure

- For task templates modeled in WebSphere Integration Developer, specify the plug-in in the task model.
- For ad hoc tasks or ad hoc task models, specify the plug-in when you create the task or task model.

  Use the setEventHandlerName method of the TTask class to register the name of the event handler.
- Change the 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.

Using a plug-in to post-process people query results

People resolution in Business Process Choreographer returns a list of the users that are assigned to a specific role, for example, potential owners of a task. You can create a plug-in that changes the results of people queries that are returned by people resolution. For example, to improve workload balancing, you could remove users from the query result who already have a high workload.

About this task

To modify the results that are returned by people assignment and people substitution, you must write a class that implements the plug-in interface, assemble a JAR file for the plug-in, then install and activate it.

Complete the following steps to create a plug-in to post-process people query results.

Procedure

1. Implement your people query result post-processing plug-in. Write a class that implements either the StaffQueryResultPostProcessorPlugin interface or the StaffQueryResultPostProcessorPlugin2 interface.
2. Create an installable JAR file.
   a. Assemble your plug-in class and its helper classes into a JAR file.
   b. 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 must conform to the Java EE service provider interface specification.
      1) In a text editor, create a service provider configuration file with the name `com.ibm.task.spi.`*plug-in_name*`StaffQueryResultPostProcessorPlugin`, where *plug-in_name* is the name of the plug-in. The name of the configuration file does not depend on the name of the interface that you implemented. For example, if your plug-in is called `MyHandler` and it implements the com.ibm.task.spi.StaffQueryResultPostProcessorPlugin2 interface, the name of the configuration file is `com.ibm.task.spi.`**MyHandler**`StaffQueryResultPostProcessorPlugin`.
      2) In the first line of the file that is neither a comment line (a line that starts with a number sign (#)) 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.
3. Install the JAR file in a shared library in the application server and associate it with the Human Task Manager application.
   a. Define a WebSphere Application Server shared library for the plug-in on the scope of the server or cluster where Business Process Choreographer is configured.
   b. Associate the shared library with the `TaskContainer` application.
   c. Make the plug-in JAR file available to each affected WebSphere Process Server installation that hosts a server or a cluster member.
4. Configure the Human Task Manager to use the plug-in.
   a. In the administrative console, go to the Custom Properties page of the Human Task Manager.

      Click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**. Under **Additional Properties**, select **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, for example, `MyHandler`.

   The plug-in is now available for post processing people query results.

5. Restart the server to activate the plug-in. The post processing plug-in is invoked after both the people assignment and people substitution have run.

Note: If you modify the plug-in, you must replace the JAR file in the shared library, and restart the server.

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 and Human Task Manager 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.

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.

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 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.

Deployment of business processes and human tasks

Use WebSphere Integration Developer or serviceDeploy to package process components or task components in an enterprise application (EAR) file. 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. The versions are differentiated by their valid-from dates. 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.

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 conditions must apply:

- 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 must have access to the Business Process Choreographer database.
- If the application is installed on a managed server, the deployment manager and the managed server must be running. The server must have access to the Business Process Choreographer database.
- Either there are no instances of business process or human task templates present in any state, or you have a stand-alone server that is running in development mode.
- If a process instance was migrated to a newer version of the process but it is waiting for a service invocation to reply, the application that contains the previous version cannot be uninstalled until the reply is received. In all other cases, instances that have been migrated are considered to be instances of the new version, and the application that contains the older version of the process can be uninstalled.

About this task

To uninstall an enterprise application that contains business processes or human tasks, perform the following actions:

Procedure

1. In the administrative console, click **Applications** → **Application Types** → **WebSphere enterprise applications**.
2. 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. You can either use the Business Process Choreographer Explorer to delete the instances, or the **-force** option of the bpcTamplates.jacl administrative script to stop and delete these instances before the application is uninstalled.
3. Select the application that you want to uninstall, and click **Uninstall**.
4. Click **Save** to save your changes.

Results

The application is uninstalled.

Related tasks

"Uninstalling business process and human task applications, using an administrative command"
Using the bpcTemplates.jacl script provides an alternative to the administrative console for uninstalling applications that contain business processes or human tasks.

Uninstalling business process and human task applications, using an administrative command

Using the bpcTemplates.jacl script provides 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 conditions must apply:

- 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 must have access to the Business Process Choreographer database.
- If the application is installed on a managed server, the deployment manager and the managed server must be running. The server must have access to the Business Process Choreographer database.
- Ensure that the server process to which the administrative 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.
- If WebSphere administrative security is enabled, and your user ID does not have operator or administrator authority, include the wsadmin `-user` and `-password`

options to specify a user ID that has operator or administrator authority. The -uninstall option requires operator authority and the -force option requires administrator authority.

- One or more of the following is true:
  - There are no instances of business process or human task templates present in any state.
  - You intend to use the **-force** option.
  - You have a stand-alone server that is running in development mode.
- If a process instance was migrated to a newer version of the process but it is waiting for a service invocation to reply, the application that contains the previous version cannot be uninstalled until the reply is received. In all other cases, instances that have been migrated are considered to be instances of the new version, and the application that contains the older version of the process can be uninstalled.

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.

Procedure

1. If there are still process instances or task instances associated with the templates in the application that you want to uninstall perform one or both of the following:
   - Use the Business Process Choreographer Explorer to delete the instances.
   - In cases where you are sure that no other business processes depend on the process templates that are defined in the application you want to uninstall, you can use the **-force** option.

     CAUTION:

     If you use the script with this option, it deletes any instances that are associated with the templates, all of the data that is associated with any running instances, stops the templates, and uninstalls the application in one step. Use this option with extreme care.

2. Change to the Business Process Choreographer subdirectory where the administrative scripts are located.

   `Linux` `UNIX` On Linux and UNIX platforms, enter the following command:

   ```
   cd install_root/ProcessChoreographer/admin
   ```

   `Windows` On Windows platforms, enter the following command:

   ```
   cd install_root\ProcessChoreographer\admin
   ```

3. Stop the templates and uninstall the corresponding application.

   `Windows` On Windows platforms, enter:

   ```
   install_root\bin\wsadmin -f bpcTemplates.jacl
                              -uninstall application_name
                              [-force]
   ```

   `Linux` `UNIX` On Linux and UNIX platforms, enter:

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                              -uninstall application_name
                              [-force]
   ```

   Where:

-uninstall *application_name*
>   This specifies the name of the application to be uninstalled.

-force
>   This option causes any running instances to be stopped and deleted before the application is uninstalled. Use this option with care because it also deletes all of the data associated with the running instances.

Results

The application is uninstalled.

Related tasks

"Uninstalling business process and human task applications, using the administrative console" on page 650
You can use the administrative console to uninstall applications that contain business processes or human tasks.

Part 5. Monitoring business processes and tasks

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.

Business process events overview

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 events can have one of the following formats:

WebSphere Business Monitor 6.1, 6.2 or 7.0 format (XML with schema support)
Events are produced in this format if there are processes modeled in WebSphere Integration Developer 6.1, or later, and this format is selected.

The object-specific content for these events is written as XML elements in the `xs:any` slot in the eventPointData part of the Common Base Event (CBE), and the payload message is written to the *applicationData* section. The structure of the XML is defined in the schema definition file `install_root\ProcessChoreographer\client\BFMEvents.xsd`. To parse and validate the CBE information, use the schema definition in `install_root\ProcessChoreographer\client\WBIEvent.xsd`

WebSphere Business Monitor 6.0.2 format (legacy XML)
Events are produced in this format if there are processes modeled in WebSphere Integration Developer 6.0.2, or if the WebSphere Business Monitor 6.0.2 format is selected in WebSphere Integration Developer 6.1 or later. If not specified otherwise, the object-specific content for these events is written as *extendedDataElement* XML elements of the type string.

Legacy hexBinary
Events are produced in this format if selected in WebSphere Integration Developer.

Related reference

"Business process events" on page 677
Common Base Events are emitted for business processes if monitoring is requested for the business process elements in WebSphere Integration Developer. A process can cause process events, activity events, activity scope events, link events, and variable events to be emitted.

"Common Base Events for business processes" on page 678
Common Base Events are emitted for business processes if monitoring is requested for the business process in WebSphere Integration Developer. A list of all the events that can be emitted by a business process can be found here. These events are also written to the audit log.

"Common Base Events for activities" on page 682
Common Base Events are emitted for activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity can be found here. These events are also written to the audit log.

"Common Base Events for scope activities" on page 692
Common Base Events are emitted for scope activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity scope can be found here. These events are also written to the audit log.

"Common Base Events for links in flow activities" on page 695
Common Base Events for links are emitted if monitoring is requested in WebSphere Integration Developer for the flow activity on which the link is defined. A list of all the events that can be emitted by a link can be found here. These events are also written to the audit log.

"Common Base Events for process variables" on page 696
Common Base Events are emitted for process variables if monitoring is requested for the business process elements in WebSphere Integration Developer. A list of all the events that can be emitted by variables can be found here. These events are also written to the audit log.

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 (CBE). 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.ACTIVITY
- "BPC.BFM.ACTIVITY.BASE" on page 661
- "BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING" on page 663
- "BPC.BFM.ACTIVITY.CLAIM" on page 664
- "BPC.BFM.ACTIVITY.CONDITION" on page 664
- "BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET" on page 664

BPC.BFM.ACTIVITY.BASE

BPC.BFM.ACTIVITY.BASE inherits the XML elements from "BPC.BFM.BASE" on page 669.

Table 84. 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: 3 - KIND_EMPTY 21 - KIND_INVOKE 23 - KIND_RECEIVE 24 - KIND_REPLY 25 - KIND_THROW 26 - KIND_TERMINATE 27 - KIND_WAIT 29 - KIND_COMPENSATE 30 - KIND_SEQUENCE 32 - KIND_SWITCH 34 - KIND_WHILE 36 - KIND_PICK 38 - KIND_FLOW 40 - KIND_SCOPE 42 - KIND_SCRIPT 43 - KIND_STAFF 44 - KIND_ASSIGN 45 - KIND_CUSTOM 46 - KIND_RETHROW 47 - KIND_FOR_EACH_SERIAL 49 - KIND_FOR_EACH_PARALLEL 52 - KIND_REPEAT_UNTIL 1000 - SQLSnippet 1001 - RetrieveSet 1002 - InvokeInformationService 1003 - AtomicSQLSnippetSequence

Table 84. XML elements for BPC.BFM.ACTIVITY.BASE (continued)

XML element	Description
state	The current state of the activity instance in the format: *state code-state name*. For activities, this attribute can have one of the following values: 1 - STATE_INACTIVE 2 - STATE_READY 3 - STATE_RUNNING 4 - STATE_SKIPPED 5 - STATE_FINISHED 6 - STATE_FAILED 7 - STATE_TERMINATED 8 - STATE_CLAIMED 11 - STATE_WAITING 12 - STATE_EXPIRED 13 - STATE_STOPPED For scope activities, this attribute can have one of the following values: 1 - STATE_READY 2 - STATE_RUNNING 3 - STATE_FINISHED 4 - STATE_COMPENSATING 5 - STATE_FAILED 6 - STATE_TERMINATED 7 - STATE_COMPENSATED 8 - STATE_COMPENSATION_FAILED 9 - STATE_FAILING 10 - STATE_SKIPPED 11 - STATE_COMPENSATION_FAILING 12 - STATE_FAULTHANDLER_FAILING 13 - STATE_FINISHING 14 - STATE_STOPPED
bpelId	The wpc:id attribute of the activity in the BPEL file. It is unique for activities in a process model.
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 on whose behalf the current action is being performed.
taskInstanceId	The ID of the associated human task instance. This is only included for staff activity events.
processTemplateId	The ID of the process template.

BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING

BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 85. XML elements for BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING

XML element	Description
subState	The substate of the activity. The substate can be one of the following strings: SUB_STATE_NONE SUB_STATE_EXPIRING SUB_STATE_SKIPPING SUB_STATE_RESTARTING SUB_STATE_FINISHING SUB_STATE_FAILING
childProcessInstanceID	The ProcessInstanceID of the child process.

BPC.BFM.ACTIVITY.CLAIM

BPC.BFM.ACTIVITY.CLAIM inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 86. XML elements for BPC.BFM.ACTIVITY.CLAIM

XML element	Description
username	The name of the user for whom the task has been claimed.

BPC.BFM.ACTIVITY.CONDITION

BPC.BFM.ACTIVITY.CONDITION inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 87. XML elements for BPC.BFM.ACTIVITY.CONDITION

XML element	Description
branchBpelId	This is set to the value of the wpc:id attribute of the related case element, as specified in the BPEL file. This information is provided only for processes that are installed with version 6.1.2 or later.
condition	This specifies the condition as a string for XPath conditions. (This property is not present for otherwise or Java conditions.)
isForced	This specifies whether the event is triggered through the forceNavigate APIs (=true), or in any other way (=false).
isOtherwise	This specifies whether the otherwise branch is entered (=true) or a case branch is entered (=false).

BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET

BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 88. XML elements for BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET

XML element	Description
propertyName	The name of the custom property.
propertyValue	The value of the custom property.
associatedObjecID	The ID of the associated object that is the activity instance ID.
associatedObjectName	The name of the associated object that is the activity template name.
query	If isBinary is true, this element specifies the query string for the binary property. Otherwise, this element is not present.
type	If isBinary is true, this element specifies the type of the binary property. Otherwise, this element is not present.
isBinary	Set to false for string custom properties, and to true for binary custom properties. The payload type for binary custom properties is restricted to Empty. The property propertyValue is omitted for binary custom properties.

BPC.BFM.ACTIVITY.ESCALATED

BPC.BFM.ACTIVITY.ESCALATED inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 89. XML elements for BPC.BFM.ACTIVITY.ESCALATED

XML element	Description
escalationName	The name of the escalation.
operation	This is 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.BASE" on page 661.

Table 90. XML elements for BPC.BFM.ACTIVITY.EVENT

XML element	Description
operation	The name of the operation for the received event.

BPC.BFM.ACTIVITY.FAILURE

BPC.BFM.ACTIVITY.FAILURE inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 91. XML elements for BPC.BFM.ACTIVITY.FAILURE

XML element	Description
activityFailedException	The exception that caused the activity to fail.
faultNamespace	The namespace URI of the fault.
faultName	The local part of the fault.

BPC.BFM.ACTIVITY.FOREACH

BPC.BFM.ACTIVITY.FOREACH inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 92. XML elements for BPC.BFM.ACTIVITY.FOREACH

XML element	Description
parallelBranchesStarted	The number of branches started.

BPC.BFM.ACTIVITY.JUMPED

BPC.BFM.ACTIVITY.JUMPED inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 93. XML elements for BPC.BFM.ACTIVITY.JUMPED

XML element	Description
targetName	Contains the activity template name of the target activity for the jump. The aiid contained in the ECSCurrentId of the event refers to the source activity of the jump.

BPC.BFM.ACTIVITY.MESSAGE

BPC.BFM.ACTIVITY.MESSAGE inherits the XML elements from "BPC.BFM.ACTIVITY.BASE" on page 661.

Table 94. 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. 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.SKIP_ON_EXIT_CONDITION_TRUE

BPC.BFM.ACTIVITY.SKIP_ON_EXIT_CONDITION_TRUE inherits the XML
elements from "BPC.BFM.ACTIVITY.BASE" on page 661. No further specific
properties are defined for
BPC.BFM.ACTIVITY.SKIP_ON_EXIT_CONDITION_TRUE beyond the inherited
properties

BPC.BFM.ACTIVITY.SKIP_REQUESTED

BPC.BFM.ACTIVITY.SKIP_REQUESTED inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 661.

Table 95. XML elements for BPC.BFM.ACTIVITY.SKIP_REQUESTED

XML element	Description
cancel	Cancel specifies whether the activity is skipped or not to distinguish between a skip (=false) and a cancelSkipRequest (=true) call.

BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST

BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 661. No further specific properties are
defined for this BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST beyond the
inherited properties

BPC.BFM.ACTIVITY.STATUS

BPC.BFM.ACTIVITY.STATUS inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 661.

Table 96. XML elements for BPC.BFM.ACTIVITY.STATUS

XML element	Description
reason	The stop reason code. The stop reason code is only relevant if the activity is in the stopped state. It indicates the reason why the activity stopped. This attribute can have one of the following values: 1 - STOP_REASON_UNSPECIFIED 2 - STOP_REASON_ACTIVATION_FAILED 3 - STOP_REASON_IMPLEMENTATION_FAILED 4 - STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED 5 - STOP_REASON_EXIT_CONDITION_FALSE

Note: A payload is available for the event nature FRETRIED for activities that
provide payload for the ENTRY event nature, and similarly for the event nature
FCOMPLETED corresponding to the EXIT event nature:

- FRETRIED and ENTRY for the element kinds invoke and staff (see CREATED).
- FCOMPLETED and EXIT for the element kinds: pick, receive, and reply.

The payload is provided in the application data section of the event, but only for
event version 6.1.

BPC.BFM.ACTIVITY.TIMER_RESCHEDULED

BPC.BFM.ACTIVITY.TIMER_RESCHEDULED inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 661.

Table 97. XML elements for BPC.BFM.ACTIVITY.TIMER_RESCHEDULED

XML element	Description
timestamp	The date and time are expressed in Coordinated Universal Time (UTC), in the format *yyyy-MM-dd*[T*hh:mm:ss*], which represents year, month, day, T, hours, minutes, and seconds, respectively.

BPC.BFM.ACTIVITY.WISTATUS

BPC.BFM.ACTIVITY.WISTATUS inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 661.

Table 98. XML elements for BPC.BFM.ACTIVITY.WISTATUS

XML element	Description
username	The names of the users who are associated with the work item.
reason	The reason for the assignment of the work item. Possible integer values have the following meanings: 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_ CREATOR

BPC.BFM.ACTIVITY.WITRANSFER

BPC.BFM.ACTIVITY.WITRANSFER inherits the XML elements from
"BPC.BFM.ACTIVITY.BASE" on page 661.

Table 99. 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.

Table 99. XML elements for BPC.BFM.ACTIVITY.WITRANSFER (continued)

XML element	Description
reason	The reason for the assignment of the work item. Possible integer values have the following meanings: 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_ CREATOR

BPC.BFM.BASE

BPC.BFM.BASE inherits the XML elements from WBIMonitoringEvent.

Table 100. 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.
eventProgressCounter	The event progress counter is used to indicate the position of the current navigation step in the execution order of all navigation steps of the same process instance. The event progress counter is required for long-running processes, and it can be used together with the event local counter to recreate the (possibly incomplete) order of the events belonging to the same process instance. In microflows, the event progress counter is set to zero.
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.
processInstanceName	The process instance name, as provided by an API invocation, is only present if it is different from the process instance ID.
processInstanceId	The ID of the process instance.

BPC.BFM.LINK.STATUS

BPC.BFM.LINK.STATUS inherits the XML elements from "BPC.BFM.BASE" on page 669.

Table 101. 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.
sourceBpelId	The wpc:id attribute of the source activity corresponding to the navigated link.
targetBpelId	The wpc:id attribute of the target activity corresponding to the navigated link.
isForced	This specifies whether the event is triggered through the forceNavigate APIs (=true), or in any other way (=false).
processTemplateId	The ID of the process template.

BPC.BFM.PROCESS.BASE

BPC.BFM.PROCESS.BASE inherits the XML elements from "BPC.BFM.BASE" on page 669.

Table 102. 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: 1 - STATE_READY 2 - STATE_RUNNING 3 - STATE_FINISHED 4 - STATE_COMPENSATING 5 - STATE_FAILED 6 - STATE_TERMINATED 7 - STATE_COMPENSATED 8 - STATE_TERMINATING 9 - STATE_FAILING 11 - STATE_SUSPENDED 12 - STATE_COMPENSATION_FAILED
processTemplateId	The ID of the process template.
processInstanceDescription	The description of the process instance.
principal	The name of the user who is associated with this event.

BPC.BFM.PROCESS.CORREL

BPC.BFM.PROCESS.CORREL inherits the XML elements from "BPC.BFM.PROCESS.BASE."

Table 103. XML elements for BPC.BFM.PROCESS.CORREL

XML element	Description

correlationSet	This is a hexBinary string. After converting it to a string, it has the following format:
	
	```<?xml version="1.0"?>```
	```<correlationSet name="correlation_set_name">```
	``` <property name="property_name"```
	``` value="property_value"/>*```
	```</correlationSet>```

action	Contains one of the following strings: **init** This indicates that the correlation set property *correlationSet* was initialized. **set** This indicates that the value of the correlation set property *correlationSet* was set using the API. **unset** This indicates that the correlation set property *correlationSet* was deleted or unset using the API, causing the property to contain no value.

BPC.BFM.PROCESS.CUSTOMPROPERTYSET

BPC.BFM.PROCESS.CUSTOMPROPERTYSET inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 104. XML elements for BPC.BFM.PROCESS.CUSTOMPROPERTYSET

XML element	Description
propertyName	The name of the custom property.
propertyValue	The value of the custom property.
associatedObjectID	The ID of the associated object that is the process instance ID.
associatedObjectName	The name of the associated object that is the process template name.
query	If isBinary is true, this element specifies the query string for the binary property. Otherwise, this element is not present.
type	If isBinary is true, this element specifies the type of the binary property. Otherwise, this element is not present.
isBinary	Set to false for string custom properties, and to true for binary custom properties. The payload type for binary custom properties is restricted to Empty. The property propertyValue is omitted for binary custom properties.

BPC.BFM.PROCESS.ESCALATED

BPC.BFM.PROCESS.ESCALATED inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 105. XML elements for BPC.BFM.PROCESS.ESCALATED

XML element	Description
escalationName	The name of the escalation.
operation	This is 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.BASE" on page 670.

Table 106. 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. 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.FAILURE

BPC.BFM.PROCESS.FAILURE inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 107. XML elements for BPC.BFM.PROCESS.FAILURE

XML element	Description
processFailedException	The exception message that lead to the failure of the process.
faultNamespace	The namespace URI of the fault.
faultName	The local part of the fault.

BPC.BFM.PROCESS.MIGRATED

BPC.BFM.PROCESS.MIGRATED inherits the XML elements from
"BPC.BFM.PROCESS.BASE" on page 670.

Table 108. XML elements for BPC.BFM.PROCESS.MIGRATED

XML element	Description
migratedFromPTID	The ID of the process template that is being migrated from.
migratedFromValidFrom	The validFrom date for the process template that is being migrated from.
After the migration, information about the activity instances within the process is provided in a business object (BO) in the application data section of the Common Base Event (CBE). The business object is defined by in the *install_root*/ProcessChoreographer/client/ BFMEvent_Data_V7.xsd file. On Windows platforms, it is in *install_root*\ ProcessChoreographer\client\BFMEvent_Data_V7.xsd. The business object contains the following information for each activity that is has been migrated.	
activityInstanceID	The ID of the activity instance.
activityState	The current execution state of the activity in the following format: *state_code-state_name*
activitySubState	The current execution substate of the activity in the following format: *substate_code-substate_name*
activityStopReason	The stop reason code. The stop reason code is only relevant if the activity is in the stopped state. It indicates the reason why the activity stopped. This attribute can have one of the following values: 1 - STOP_REASON_UNSPECIFIED 2 - STOP_REASON_ACTIVATION_FAILED 3 - STOP_REASON_IMPLEMENTATION_ FAILED 4 - STOP_REASON_FOLLOW_ON_ NAVIGATION_FAILED 5 - STOP_REASON_EXIT_ CONDITION_FALSE
bpelId	The BPEL ID of the activity.
activityTemplateName	The name of the activity template.
activityTemplateId	The ID of the activity template.

BPC.BFM.PROCESS.MIGRATIONTRIGGERED

BPC.BFM.PROCESS.MIGRATIONTRIGGERED inherits the XML elements from
"BPC.BFM.PROCESS.BASE" on page 670.

Table 109. XML elements for BPC.BFM.PROCESS.MIGRATIONTRIGGERED

XML element	Description
migrateToPTID	The ID of the process template to migrate to.
migrateToValidFrom	The validFrom date for the process template to migrate to.

XML element	Description
Before the migration, information about the activity instances within the process is provided in a business object (BO) in the application data section of the Common Base Event (CBE). The business object is defined in the *install_root*/ProcessChoreographer/client/BFMEvent_Data_V7.xsd file. On Windows platforms, it is in *install_root*\ProcessChoreographer\client\BFMEvent_Data_V7.xsd. The business object contains the following information for each activity that will be migrated.	
activityInstanceID	The ID of the activity instance.
activityState	The current execution state of the activity in the following format: *state_code-state_name*
activitySubState	The current execution substate of the activity in the following format: *substate_code-substate_name*
activityStopReason	The stop reason code. The stop reason code is only relevant if the activity is in the stopped state. It indicates the reason why the activity stopped. This attribute can have one of the following values: 1 - STOP_REASON_UNSPECIFIED 2 - STOP_REASON_ACTIVATION_FAILED 3 - STOP_REASON_IMPLEMENTATION_ FAILED 4 - STOP_REASON_FOLLOW_ON_ NAVIGATION_FAILED 5 - STOP_REASON_EXIT_ CONDITION_FALSE
bpelId	The BPEL ID of the activity.
activityTemplateName	The name of the activity template.
activityTemplateId	The ID of the activity template.

BPC.BFM.PROCESS.OWNERTRANSFER

BPC.BFM.PROCESS.OWNERTRANSFER inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 110. XML elements for BPC.BFM.PROCESS.OWNERTRANSFER

XML element	Description
current	The user name of the current owner of the process. This is the user whose process is transferred to someone else.
target	The user name of the new owner of the process.

BPC.BFM.PROCESS.PARTNER

BPC.BFM.PROCESS.PARTNER inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 111. XML elements for BPC.BFM.PROCESS.PARTNER

XML element	Description
partnerLinkName	The name of the partner link.

Note: The endpoint reference for a BPC.BFM.PROCESS.PARTNER event is only written to the application data section of the CBE for event version 6.1. The payload is the Web Services Business Process Execution Language (WS-BPEL) ServiceRefType wrapper element that contains the Web Services Addressing (WS-Addressing) EndpointReferenceType element. The ServiceRefType artefact (schema) must be available in the context of the application, which is the case in typical scenarios. However, if you dynamically assign an endpoint from one statically defined partner link to another, the schema is not available, and the endpoint reference is not included.

BPC.BFM.PROCESS.START

BPC.BFM.PROCESS.START inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 112. XML elements for BPC.BFM.PROCESS.START

XML element	Description
username	The name of the user who requested the start or restart of the process.

BPC.BFM.PROCESS.STATUS

BPC.BFM.PROCESS.STATUS inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

BPC.BFM.PROCESS.WISTATUS

BPC.BFM.PROCESS.WISTATUS inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 113. XML elements for BPC.BFM.PROCESS.WISTATUS

XML element	Description
username	The names of the users with work items that were created or deleted.
reason	The reason for the assignment of the work item. Possible integer values have the following meanings: 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_ CREATOR

BPC.BFM.PROCESS.WITRANSFER

BPC.BFM.PROCESS.WITRANSFER inherits the XML elements from "BPC.BFM.PROCESS.BASE" on page 670.

Table 114. 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.
reason	The reason for the assignment of the work item. Possible integer values have the following meanings: 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_ CREATOR

BPC.BFM.VARIABLE.STATUS

BPC.BFM.VARIABLE.STATUS inherits the XML elements from "BPC.BFM.BASE" on page 669.

Table 115. 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. 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 on whose behalf the current action is being performed.
processTemplateId	The ID of the process template.

Related reference

"Business process events"
Common Base Events are emitted for business processes if monitoring is requested for the business process elements in WebSphere Integration Developer. A process can cause process events, activity events, activity scope events, link events, and variable events to be emitted.

"Common Base Events for business processes" on page 678
Common Base Events are emitted for business processes if monitoring is requested for the business process in WebSphere Integration Developer. A list of all the events that can be emitted by a business process can be found here. These events are also written to the audit log.

"Common Base Events for activities" on page 682
Common Base Events are emitted for activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity can be found here. These events are also written to the audit log.

"Common Base Events for scope activities" on page 692
Common Base Events are emitted for scope activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity scope can be found here. These events are also written to the audit log.

"Common Base Events for links in flow activities" on page 695
Common Base Events for links are emitted if monitoring is requested in WebSphere Integration Developer for the flow activity on which the link is defined. A list of all the events that can be emitted by a link can be found here. These events are also written to the audit log.

"Common Base Events for process variables" on page 696
Common Base Events are emitted for process variables if monitoring is requested for the business process elements in WebSphere Integration Developer. A list of all the events that can be emitted by variables can be found here. These events are also written to the audit log.

Business process events

Common Base Events are emitted for business processes if monitoring is requested for the business process elements in WebSphere Integration Developer. A process can cause process events, activity events, activity scope events, link events, and variable events to be emitted.

All business process events can be emitted in both the CEI and the audit trail, with the exception of the process template events. The process template events PROCESS_INSTALLED and PROCESS_UNINSTALLED can only be emitted in the audit trail.

Note: A human task activity has an associated inline human task. When you define your business process, you can specify that both the activity and its associated inline human task emit events.

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.

In business processes, events relate to processes, activities, scopes, links, and variables.
Business process events can be emitted in different situations. The data for these situations is described in situation elements.
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.

Common Base Events for business processes

Common Base Events are emitted for business processes if monitoring is requested for the business process in WebSphere Integration Developer. A list of all the events that can be emitted by a business process can be found here. These events are also written to the audit log.

State transitions and process events

The following diagram shows the state transitions that can occur for a business process and the events that are emitted when these state changes take place. The link between each state indicates the nature of the event and the event code of the event that is emitted for the state transitions.
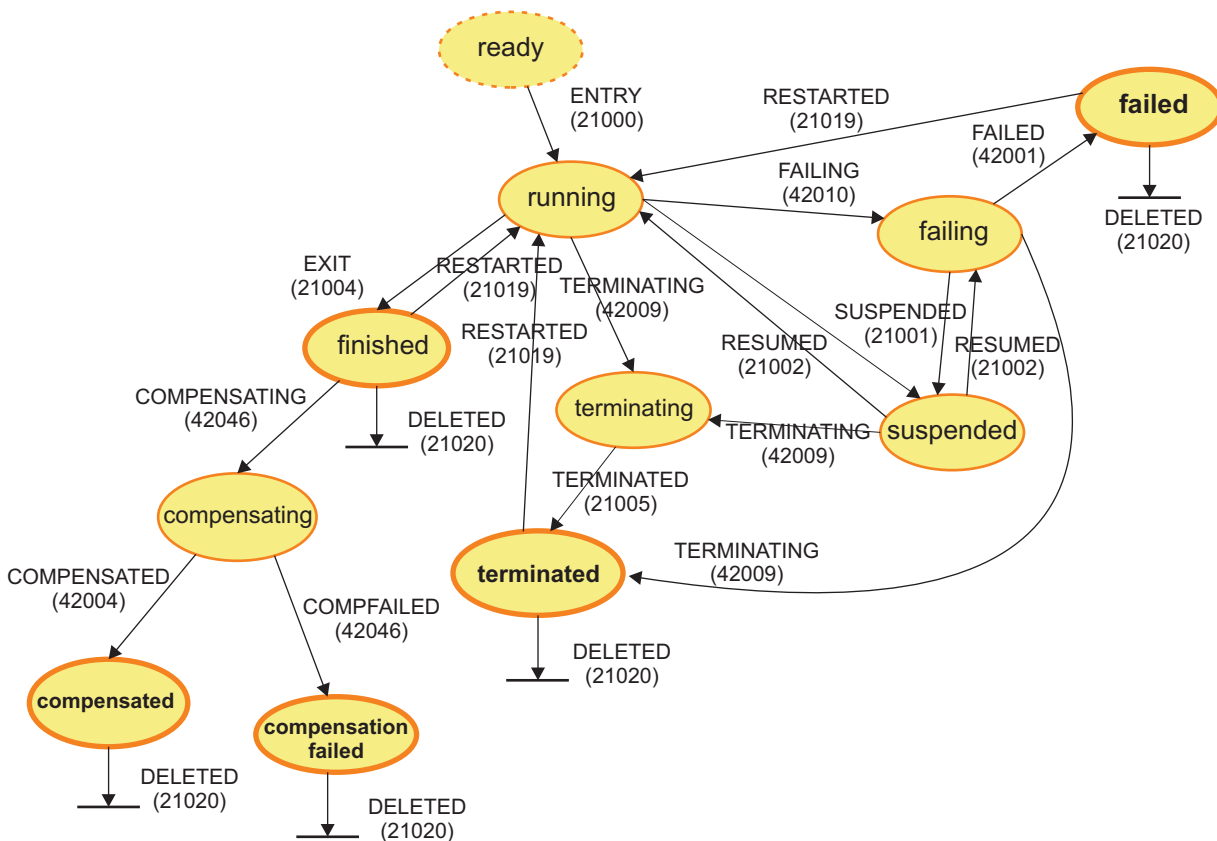
Figure 10. State transitions and process events

Process events

The columns in the following table 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.

Event name and extension name
> This column contains two values. The name of the event and the value that is set in the *extensionName* attribute of the Common Base Event. The extension name identifies which event specific information is contained in the Common Base Event, and it is also the name of the XML element that provides additional data about the event.

Situation
> Refers to the situation name of the business process event.

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.

Some process events are emitted without a state change. The following table describes all process events.

Table 116. Process events

Code	Event name and extension name	Situation	Event nature	Description
21000	PROCESS_STARTED BPC.BFM.PROCESS.START	Start	ENTRY	Process started
21001	PROCESS_SUSPENDED BPC.BFM.PROCESS.STATUS	Report	SUSPENDED	Process suspended. To suspend process instances, use Business Process Choreographer Explorer.
21002	PROCESS_RESUMED BPC.BFM.PROCESS.STATUS	Report	RESUMED	Process resumed. Only suspended processes can be resumed. To resume process instances, use Business Process Choreographer Explorer.
21004	PROCESS_COMPLETED BPC.BFM.PROCESS.STATUS	Stop	EXIT	Process completed
21005	PROCESS_TERMINATED BPC.BFM.PROCESS.STATUS	Stop	TERMINATED	Process terminated. To terminate process instances, use Business Process Choreographer Explorer.
21019	PROCESS_RESTARTED BPC.BFM.PROCESS.START	Report	RESTARTED	Process restarted. A process is restarted on request, for example, by using Business Process Choreographer Explorer.

Table 116. Process events (continued)

Code	Event name and extension name	Situation	Event nature	Description
21020	PROCESS_DELETED BPC.BFM.PROCESS.STATUS	Destroy	DELETED	Process deleted
42001	PROCESS_FAILED BPC.BFM.PROCESS. FAILURE	Fail	FAILED	Process failed
42003	PROCESS_COMPENSATING BPC.BFM.PROCESS.STATUS	Report	COMPENSATING	Process compensating. Only child processes can be compensated. The compensation of a child process is triggered by a fault handler or compensation handler associated with the parent process.
42004	PROCESS_COMPENSATED BPC.BFM.PROCESS.STATUS	Stop	COMPENSATED	Process compensated
42006	PROCESS_INSTALLED	Report	INSTALLED	These are process instance events, which are only emitted in the audit trail. They are not emitted as common base events, and are included here for completeness.
42007	PROCESS_UNINSTALLED	Report	UNINSTALLED	
42009	PROCESS_TERMINATING BPC.BFM.PROCESS.STATUS	Report	TERMINATING	Process terminating
42010	PROCESS_FAILING BPC.BFM.PROCESS.STATUS	Report	FAILING	Process failing
42027	PROCESS_CORRELATION_ SET_INITIALIZED BPC.BFM.PROCESS.CORREL	Report	CORRELATION	This event is emitted when a new correlation set for the process instance is initialized, for example, when a receive activity with an initiating correlation set receives a message. This event is not associated with a state change.
42041	PROCESS_WORKITEM_ DELETED BPC.BFM.PROCESS. WISTATUS	Report	WI_DELETED	Process work item deleted. This event is emitted only when a work item is explicitly deleted by an API request. If the work item is deleted because the corresponding process instance is deleted, an event is not emitted. This event is not associated with a state change.

Table 116. Process events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42042	PROCESS_WORKITEM_ CREATED BPC.BFM.PROCESS. WISTATUS	Report	WI_CREATED	Process work item created. This event is emitted when an additional work item is created for the process, for example, by an API request. This event is not associated with a state change.
42046	PROCESS_COMPENSATION_FAILED BPC.BFM.PROCESS.STATUS	Fail	COMPFAILED	Process compensation failed
42047	PROCESS_EVENT_RECEIVED BPC.BFM.PROCESS.EVENT	Report	EV_RECEIVED	Process event received. The event is emitted when an event handler that is associated with a process is activated. This event is not associated with a state change.
42049	PROCESS_EVENT_ESCALATED BPC.BFM.PROCESS.ESCALATED	Report	EV_ESCALATED	Process event escalated. This event is emitted when an inline invocation task is escalated that is associated with an onEvent event handler for the process. This event is not associated with a state change.
42056	PROCESS_WORKITEM_ TRANSFERRED BPC.BFM.PROCESS. WITRANSFER	Report	WI_ TRANSFERRED	Process work item transferred. This event is not associated with a state change.
42058	PROCESS_PARTNER_CHANGED BPC.BFM.PROCESS.PARTNER	Report	PA_CHANGE	Process partner changed. This event is emitted when a new endpoint reference is assigned to a partner link. This event is not associated with a state change.
42059	PROCESS_CUSTOMPROPERTY_SET BPC.BFM.PROCESS. CUSTOMPROPERTYSET	Report	CP_SET	Process custom property set. This event is emitted when a custom property of a process instance is changed. This event is not associated with a state change.
42071	PROCESS_OWNER_TRANSFERRED BPC.BFM.PROCESS.OWNERTRANSFER	Report	OWNER_ TRANSFERRED	This event is emitted when the ownership of a process is transferred from one user to another. This event is not associated with a state change.

Table 116. Process events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42077	PROCESS_CORRELATION_SET_SET BPC.BFM.PROCESS.CORREL	Report	CORRELATION	This event is emitted when the value of a correlation set for the process instance is set. This event is not associated with a state change.
42078	PROCESS_CORRELATION_SET_UNSET BPC.BFM.PROCESS.CORREL	Report	CORRELATION	This event is emitted when the value of a correlation set for the process instance is deleted or unset. This event is not associated with a state change.
42079	PROCESS_MIGRATED BPC.BFM.PROCESS.MIGRATED	Report	MIGRATED	This event is emitted when a process is migrated to use a new template. This event is not associated with a state change.
42080	PROCESS_MIGRATION_TRIGGERED BPC.BFM.PROCESS. MIGRATIONTRIGGERED	Report	MIGRATION_ TRIGGERED	This event is emitted when the migration of process instances to use a new template is started. This event is not associated with a state change.

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.

Related reference

"Event data specific to business processes" on page 659
In business processes, events relate to processes, activities, scopes, links, and variables.

"Situations in business process events" on page 697
Business process events can be emitted in different situations. The data for these situations is described in situation elements.

"Extension names for business process events" on page 660
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.

Common Base Events for activities

Common Base Events are emitted for activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity can be found here. These events are also written to the audit log.

State transitions and activity events

The state changes and the events that are emitted depend on the type of activity:

- Invoke, assign, empty, reply, rethrow, throw, terminate, and Java snippet activities

Figure 11. State transitions and events for invoke activities and short-lived activities

- Pick (receive choice), wait, and receive activities

Figure 12. State transitions and events for wait, and receive activities

- Human task activities
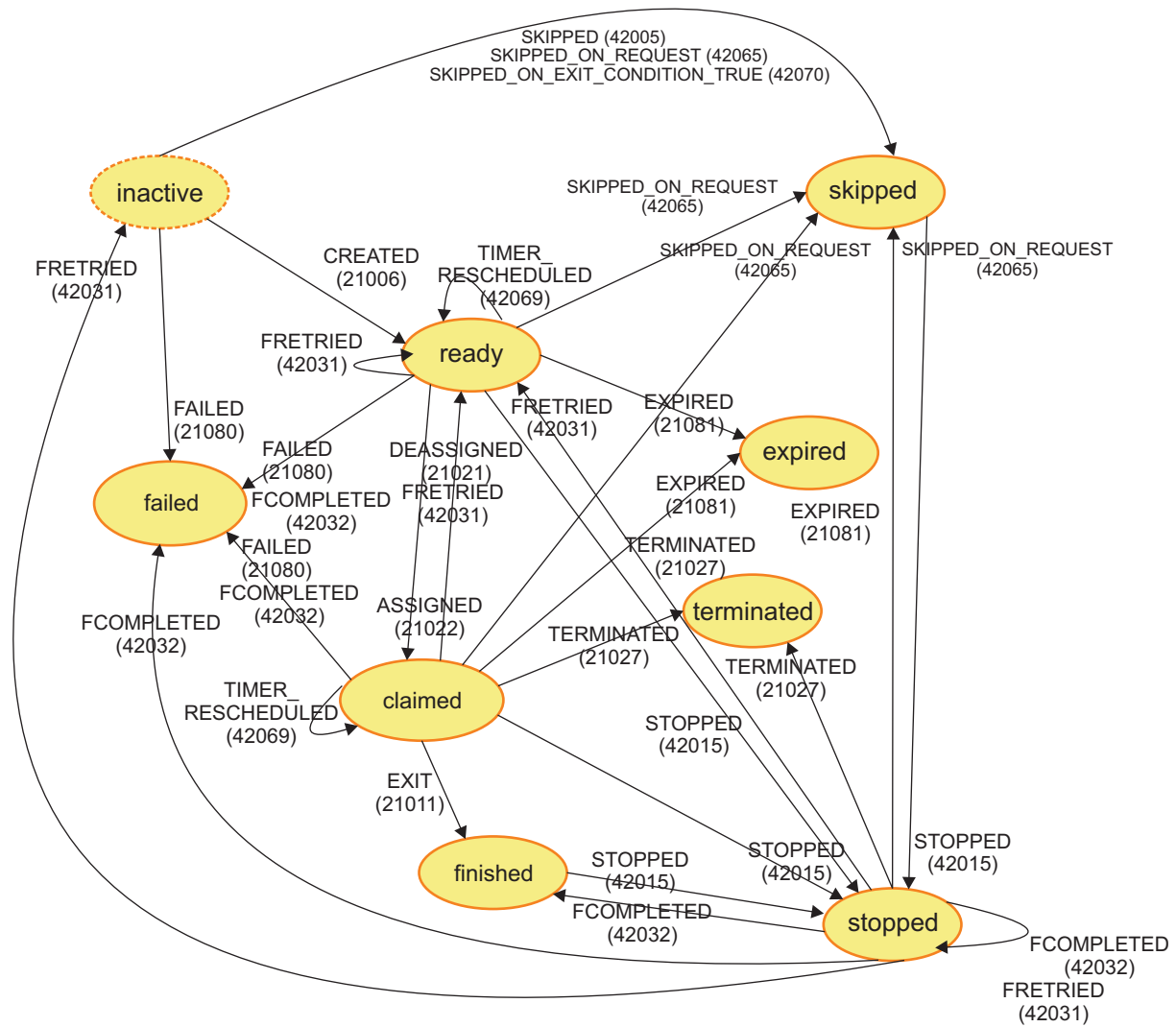
Figure 13. State transitions and events for human task activities

- Structured activities, such as flow or sequence activities
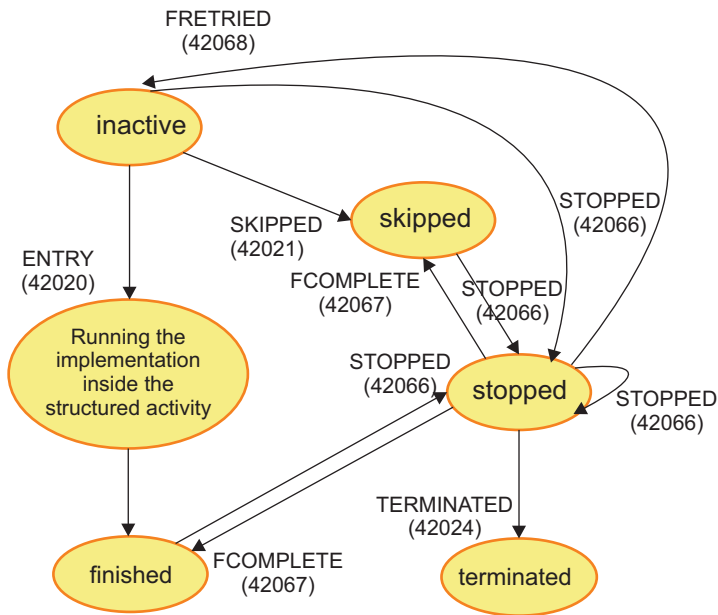
Figure 14. State transitions and events for structured activities

Activity events

The columns in the following table 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.

Event name and extension name
This column contains two values. The name of the event and the value that is set in the *extensionName* attribute of the Common Base Event. The extension name identifies which event specific information is contained in the Common Base Event, and it is also the name of the XML element that provides additional data about the event.

Situation
Refers to the situation name of the business process event.

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.

The following table describes all activity events.

Table 117. Activity events

Code	Event name and extension name	Situation	Event nature	Description
21006	ACTIVITY_READY BPC.BFM.ACTIVITY. MESSAGE	Start	CREATED	Activity ready. This event is emitted when a human task activity is started.

Table 117. Activity events (continued)

Code	Event name and extension name	Situation	Event nature	Description
21007	ACTIVITY_STARTED For invoke activities: BPC.BFM.ACTIVITY. MESSAGE For all other activity types: BPC.BFM.ACTIVITY. STATUS	Start	ENTRY	Activity started. For invoke activities, a business object payload is available.
21011	ACTIVITY_COMPLETED 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, a business object payload is available.
21021	ACTIVITY_CLAIM_ CANCELED BPC.BFM.ACTIVITY. STATUS	Report	DEASSIGNED	Claim canceled. This event is emitted when the claim for a human task activity is canceled.
21022	ACTIVITY_CLAIMED BPC.BFM.ACTIVITY. CLAIM	Report	ASSIGNED	Activity claimed. This event is emitted when a human task activity is claimed.
21027	ACTIVITY_ TERMINATED 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	ACTIVITY_FAILED BPC.BFM.ACTIVITY. FAILURE	Failed	FAILED	Activity failed. This event is emitted if a fault occurs when the activity runs and the fault is propagated to the fault handlers that are defined for the enclosing scopes or process.
21081	ACTIVITY_EXPIRED BPC.BFM.ACTIVITY. STATUS	Report	EXPIRED	Activity expired. This event applies to invoke and human task activities only.

Table 117. Activity events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42005	ACTIVITY_SKIPPED BPC.BFM.ACTIVITY. STATUS	Report	SKIPPED	Activity skipped. This event applies only 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	ACTIVITY_OUTPUT_ MESSAGE_SET BPC.BFM.ACTIVITY. MESSAGE	Report	OUTPUTSET	Activity output message set. A business object payload is available. This event is emitted when the output message for a claimed human task activity is set without completing the activity, for example, to store intermediate results. The state of the activity does not change. This event is not emitted when a human task activity is completed.
42013	ACTIVITY_FAULT_ MESSAGE_SET BPC.BFM.ACTIVITY. MESSAGE	Report	FAULTSET	Activity fault message set. Business object payload is available. This event is emitted when a fault message for a claimed human task activity is set without completing the activity. This event is not emitted when a human task activity is completed with a fault.
42015	ACTIVITY_STOPPED BPC.BFM.ACTIVITY. STATUS	Stop	STOPPED	Activity stopped. An activity can be stopped if an unhandled fault occurs when the activity runs.
42031	ACTIVITY_FORCE_ RETRIED BPC.BFM.ACTIVITY. STATUS	Report	FRETRIED	Activity forcibly retried. To force activities to retry, use Business Process Choreographer Explorer.
42032	ACTIVITY_FORCE_ COMPLETED BPC.BFM.ACTIVITY. STATUS	Stop	FCOMPLETED	Activity forcibly completed. To force activities to complete use Business Process Choreographer Explorer.
42036	ACTIVITY_MESSAGE_ RECEIVED BPC.BFM.ACTIVITY. MESSAGE	Report	EXIT	A pick (receive choice) activity has received a message

Table 117. Activity events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42037	ACTIVITY_LOOP_ CONDITION_TRUE BPC.BFM.ACTIVITY. STATUS	Report	CONDTRUE	Loop condition true
42038	ACTIVITY_LOOP_ CONDITION_FALSE BPC.BFM.ACTIVITY. STATUS	Report	CONDFALSE	Loop condition false
42039	ACTIVITY_WORKITEM_ DELETED BPC.BFM.ACTIVITY. WISTATUS	Report	WI_DELETED	Work item deleted. This event applies to pick, human tasks, and receive events only. This event is emitted only when a work item is explicitly deleted by an API request. If the work item is deleted because the corresponding process instance is deleted, an event is not emitted.
42040	ACTIVITY_WORKITEM_ CREATED BPC.BFM.ACTIVITY. WISTATUS	Report	WI_CREATED	Work items created. This event applies only to pick, human tasks, and receive events.
42050	ACTIVITY_ESCALATED BPC.BFM.ACTIVITY. ESCALATED	Report	ESCALATED	Activity escalated. This event applies only to pick, human tasks, and receive events when the escalation associated with the human task activity is raised.
42054	ACTIVITY_WORKITEM_ REFRESHED BPC.BFM.ACTIVITY. WISTATUS	Report	WI_REFRESHED	Activity work items refreshed. This event applies only to pick, human tasks, and receive events.
42055	ACTIVITY_WORKITEM_ TRANSFERRED BPC.BFM.ACTIVITY. WITRANSFER	Report	WI_TRANSFERRED	Work item transferred. This event applies only to pick, human tasks, and receive events.
42057	ACTIVITY_PARALLEL_ BRANCHES_STARTED BPC.BFM.ACTIVITY. FOREACH	Report	BRANCHES_ STARTED	This event is emitted when branches are started for a forEach activity.
42060	ACTIVITY_ CUSTOMPROPERTY_SET BPC.BFM.ACTIVITY. CUSTOMPROPERTYSET	Report	CP_SET	This event is emitted when a custom property of an activity instance is changed.

Table 117. Activity events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42061	ACTIVITY_BRANCH_ CONDITION_TRUE BPC.BFM.ACTIVITY. CONDITION	Report	CONDTRUE	This event is emitted when the case condition of a choice activity evaluates to true. There is, at most, one event with the case element condition set to true for each navigated choice activity instance. That is, non-entered case elements are not honored by an event, and otherwise elements provoke the same event as condition case elements.
42062	ACTIVITY_ALL_BRANCH_ CONDITIONS_FALSE BPC.BFM.ACTIVITY. STATUS	Report	ALLCONDFALSE	This event is emitted when no case element was used and no otherwise element exists. In this case, the navigation continues at the end of the choice construct.
42063	ACTIVITY_JUMPED BPC.BFM.ACTIVITY. JUMPED	Report	JUMPED	This event is emitted after the final activity event of the source activity of the jump action and before the first event of the target activity.
42064	ACTIVITY_SKIP_ REQUESTED BPC.BFM.ACTIVITY. SKIP_REQUESTED	Report	SKIP_REQUESTED	Skip activity requested. This event is emitted if the corresponding activity is not in an active state and a skip or cancelSkipRequest API is called. In this case, the request has no immediate effect on the navigation. The event contains a flag to distinguish between a skip and a cancelSkipRequest call. The ECSCurrentID for the event to be skipped is not set to the AIID of the associated activity.
42065	ACTIVITY_SKIPPED_ ON_REQUEST BPC.BFM.ACTIVITY. SKIPPED_ON_REQUEST	Report	SKIPPED_ON_ REQUEST	Event skipped on request. This event is emitted when the navigation after an activity that is marked for skipping is continued.
42069	ACTIVITY_TIMER_ RESCHEDULED BPC.BFM.ACTIVITY. TIMER_RESCHEDULED	Report	TIMER_ RESCHEDULED	This event is emitted when a rescheduleTimer request is processed. This event can be produced for wait, human task, invoke, and pick activities.

Table 117. Activity events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42070	ACTIVITY_SKIPPED_ ON_EXIT_ CONDITION BPC.BFM.ACTIVITY.SKIP_ ON_EXIT_CONDITION_TRUE	Report	SKIPPED_ON_ EXIT_ CONDITION_ TRUE	This event is emitted when an exit condition of the onEntry type evaluates to true, and the activity is skipped for this reason.
42072	ACTIVITY_CHILD_PROCESS_ TERMINATING BPC.BFM.ACTIVITY.CHILD_ PROCESS_TERMINATING	Report	CHILD_ PROCESS_ TERMINATING	This event is emitted if the corresponding activity is an invoke activity in the running state, has a child process, and the forceRetry, forceComplete, or skip API is called or the activity expires.
42073	ACTIVITY_CONDITION_ FORCED BPC.BFM.ACTIVITY. CONDITIONFORCED	Report	ACTIVITY_ CONDITION_ FORCED	An activity that has stopped because of an error evaluating a join condition was forced to continue navigating.
42074	ACTIVITY_LOOP_ CONDITION_FORCED BPC.BFM.ACTIVITY. CONDITIONFORCED	Report	ACTIVITY_LOOP_ CONDITION_ FORCED	An activity that has stopped because of an error evaluating a repeat-until or while loop condition was forced to continue navigating.
42075	ACTIVITY_FOR_EACH_ COUNTERS_FORCED BPC.BFM.ACTIVITY. COUNTERSFORCED	Report	ACTIVITY_ FOR_EACH_ COUNTERS_ FORCED	An activity that has stopped because of an error evaluating a for-each loop condition was forced to continue navigating.

For most 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.

For the custom property set event, the event correlation sphere identifiers indicate the context in which the custom property was set. If, for example, the custom property is set using an API request, the event correlation sphere identifiers are set as for a process event. If the custom property is set in a Java snippet, the *ECSCurrentID* is set to the activity instance ID of the Java snippet and the *ECSParentID* is set to the process instance ID.

In business processes, events relate to processes, activities, scopes, links, and variables.
Business process events can be emitted in different situations. The data for these situations is described in situation elements.
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.

Common Base Events for scope activities

Common Base Events are emitted for scope activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity scope can be found here. These events are also written to the audit log.

State transitions and events for scope activities

The following diagram shows the state transitions that can occur for a scope activity and the events that are emitted when these state changes take place.
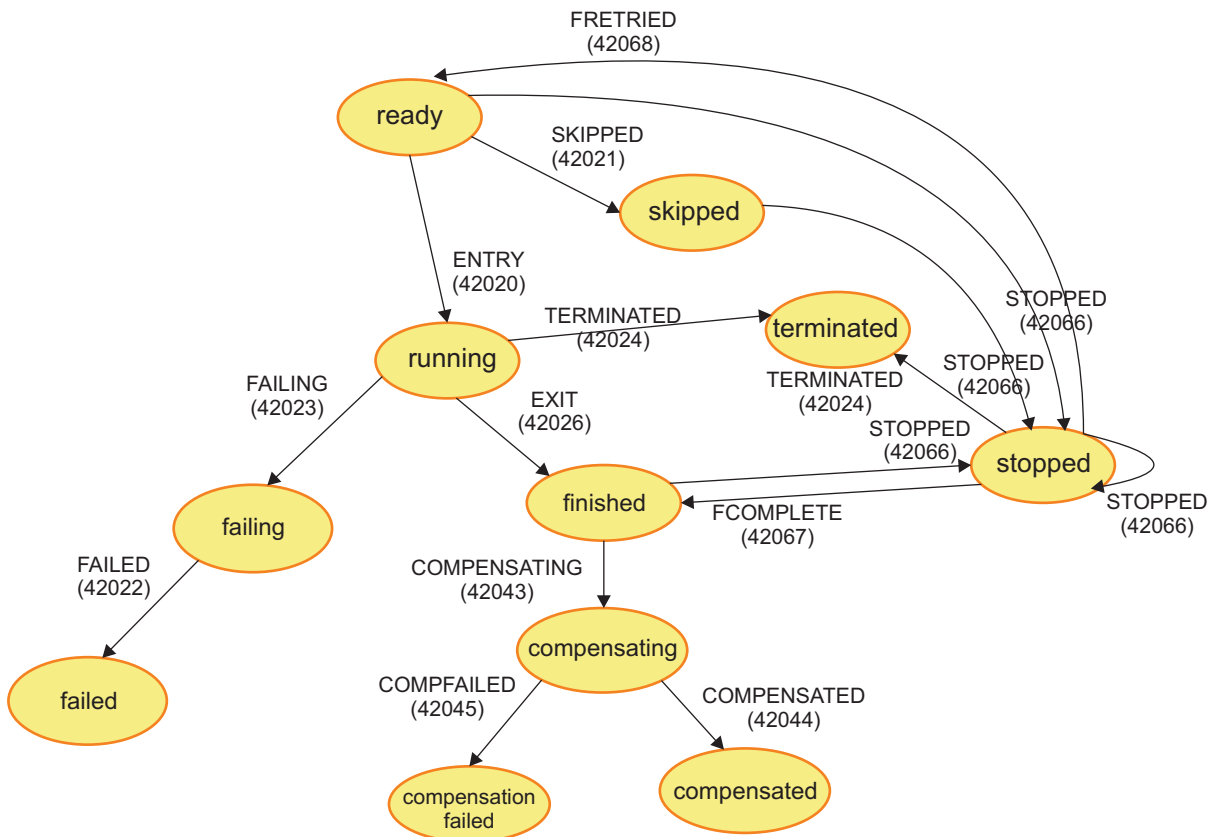
Figure 15. State transitions and events for scope activities

Scope activity events

The columns in the following table 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.

Event name and extension name

This column contains two values. The name of the event and the value that is set in the *extensionName* attribute of the Common Base Event. The extension name identifies which event specific information is contained in the Common Base Event, and it is also the name of the XML element that provides additional data about the event.

Situation

Refers to the situation name of the business process event.

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.

The following table describes all scope activity events.

Table 118. Scope activity events

Code	Event name and extension name	Situation	Event nature	Description
42020	SCOPE_STARTED BPC.BFM.ACTIVITY. STATUS	Start	ENTRY	Scope started. This event is emitted when the navigation enters the scope instance.
42021	SCOPE_SKIPPED BPC.BFM.ACTIVITY. STATUS	Report	SKIPPED	Scope skipped. The event applies only to scope activities that have join behavior defined. The event is emitted when the join condition of the scope evaluates to false. The navigation of the process continues at the end of the scope with dead-path elimination.
42022	SCOPE_FAILED BPC.BFM.ACTIVITY. FAILURE	Fail	FAILED	Scope failed. This event is emitted when the process navigation leaves the fault handler of the scope.
42023	SCOPE_FAILING BPC.BFM.ACTIVITY. STATUS	Report	FAILING	Scope failing. This event is emitted when the process navigation enters the fault handling path of the scope.
42024	SCOPE_TERMINATED BPC.BFM.ACTIVITY. STATUS	Stop	TERMINATED	Scope terminated. A scope activity can be terminated if the associated process is terminated, for example, by a terminate activity in a branch that is parallel to the scope activity.

Table 118. Scope activity events (continued)

Code	Event name and extension name	Situation	Event nature	Description
42026	SCOPE_COMPLETED BPC.BFM.ACTIVITY. STATUS	Stop	EXIT	Scope completed. This event is emitted when the normal navigation path of the scope and all of the activated event handler paths are completed.
42043	SCOPE_COMPENSATING BPC.BFM.ACTIVITY. STATUS	Report	COMPENSATING	Scope compensating. This event is emitted when the process navigation enters the compensation handler, including the default compensation handler, of the scope.
42044	SCOPE_COMPENSATED BPC.BFM.ACTIVITY. STATUS	Stop	COMPENSATED	Scope compensated. This event is emitted when the compensation handler, including default compensation handler, of the scope completes.
42045	SCOPE_COMPENSATION_ FAILED BPC.BFM.ACTIVITY. STATUS	Fail	COMPFAILED	Scope compensation failed. This event is emitted if a fault occurs when the compensation handler for the scope runs.
42048	SCOPE_EVENT_ RECEIVED BPC.BFM.ACTIVITY. EVENT	Report	EV_RECEIVED	This event is emitted when a new event handler instance is started for the scope.
42051	SCOPE_EVENT_ ESCALATED BPC.BFM.ACTIVITY. ESCALATED	Report	EV_ESCALATED	Scope event escalated. This event is emitted when the escalation is started that is associated with the inline human task of an active event handler for the scope.
42066	SCOPE_STOPPED BPC.BFM.ACTIVITY.STATUS	Stop	STOPPED	Scope is stopped. A scope instance can stop if an unhandled fault occurs during the activation or the follow-on navigation of a scope.
42067	SCOPE_FORCE_ COMPLETED BPC.BFM.ACTIVITY. STATUS	Report	FCOMPLETED	Scope is force completed
42068	SCOPE_FORCE_RETRIED BPC.BFM.ACTIVITY. STATUS	Report	FRETRIED	Scope has been force retried
42076	SCOPE_CONDITION_ FORCED BPC.BFM.ACTIVITY. CONDITIONFORCED	Report	SCOPE_ CONDITION_ FORCED	An activity that has stopped because of an error evaluating a join condition was forced to continue navigating.

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.

Related reference

"Event data specific to business processes" on page 659
In business processes, events relate to processes, activities, scopes, links, and variables.

"Situations in business process events" on page 697
Business process events can be emitted in different situations. The data for these situations is described in situation elements.

"Extension names for business process events" on page 660
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.

Common Base Events for links in flow activities

Common Base Events for links are emitted if monitoring is requested in WebSphere Integration Developer for the flow activity on which the link is defined. A list of all the events that can be emitted by a link can be found here. These events are also written to the audit log.

The following types of events can be caused by links in flow activities.

Link events

The columns in the following table 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.

Event name and extension name
This column contains two values. The name of the event and the value that is set in the *extensionName* attribute of the Common Base Event. The extension name identifies which event specific information is contained in the Common Base Event, and it is also the name of the XML element that provides additional data about the event.

Situation
Refers to the situation name of the business process event.

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.

The following table describes all link events.

Table 119. Link events

Code	Event name and extension name	Situation	Event nature	Description
21034	LINK_EVALUATED_ TO_TRUE BPC.BFM.LINK.STATUS	Report	CONDTRUE	Link evaluated true
42000	LINK_EVALUATED_ TO_FALSE 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.

Related reference

"Event data specific to business processes" on page 659
In business processes, events relate to processes, activities, scopes, links, and variables.

"Situations in business process events" on page 697
Business process events can be emitted in different situations. The data for these situations is described in situation elements.

"Extension names for business process events" on page 660
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.

Common Base Events for process variables

Common Base Events are emitted for process variables if monitoring is requested for the business process elements in WebSphere Integration Developer. A list of all the events that can be emitted by variables can be found here. These events are also written to the audit log.
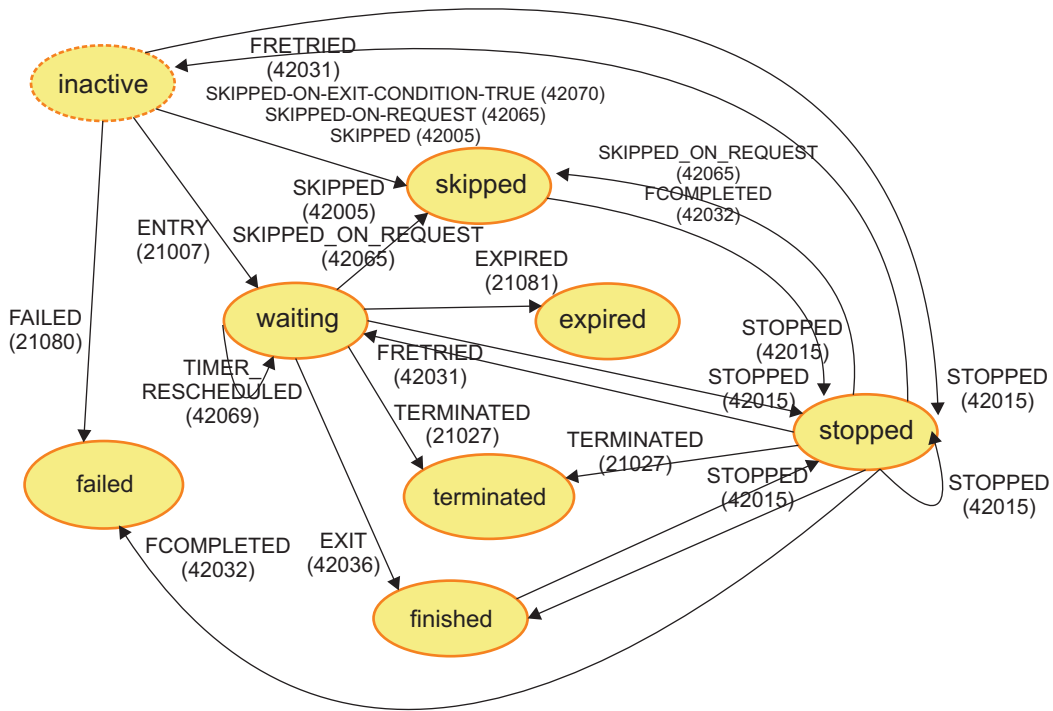
The following types of events can be caused by process variables.

Variable events

The columns in the following table 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.

Event name and extension name
This column contains two values. The name of the event and the value that is set in the *extensionName* attribute of the Common Base Event. The extension name identifies which event specific information is contained in the Common Base Event, and it is also the name of the XML element that provides additional data about the event.

Situation
Refers to the situation name of the business process event.

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.

The following table describes the variable events.

Table 120. Variable events

Code	Event name and extension name	Situation	Event nature	Description
21090	VARIABLE_UPDATED BPC.BFM.VARIABLE. STATUS	Report	CHANGED	Variable update. A 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.

Related reference

"Event data specific to business processes" on page 659
In business processes, events relate to processes, activities, scopes, links, and variables.

"Situations in business process events"
Business process events can be emitted in different situations. The data for these situations is described in situation elements.

"Extension names for business process events" on page 660
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.

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`

Situation name	Content of the Common Base Event	
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

Related reference

"Business process events" on page 677
Common Base Events are emitted for business processes if monitoring is requested for the business process elements in WebSphere Integration Developer. A process can cause process events, activity events, activity scope events, link events, and variable events to be emitted.

"Common Base Events for business processes" on page 678
Common Base Events are emitted for business processes if monitoring is requested for the business process in WebSphere Integration Developer. A list of all the events that can be emitted by a business process can be found here. These events are also written to the audit log.

"Common Base Events for activities" on page 682
Common Base Events are emitted for activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity can be found here. These events are also written to the audit log.

"Common Base Events for scope activities" on page 692
Common Base Events are emitted for scope activities if monitoring is requested for these activities in WebSphere Integration Developer. A list of all the events that can be emitted by an activity scope can be found here. These events are also written to the audit log.

"Common Base Events for links in flow activities" on page 695
Common Base Events for links are emitted if monitoring is requested in WebSphere Integration Developer for the flow activity on which the link is defined. A list of all the events that can be emitted by a link can be found here. These events are also written to the audit log.

"Common Base Events for process variables" on page 696
Common Base Events are emitted for process variables if monitoring is requested for the business process elements in WebSphere Integration Developer. A list of all the events that can be emitted by variables can be found here. These events are also written to the audit log.

Human task events overview

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 events can have one of the following formats:

WebSphere Business Monitor 6.0.2 format
WebSphere Business Monitor 6.0.2 format events occur when there are tasks modeled in WebSphere Integration Developer 6.0.2, or if the WebSphere Business Monitor 6.0.2 format (legacy XML) is enabled in WebSphere Integration Developer 6.1, or later. 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 in WebSphere Integration Developer 6.1, or later, and the WebSphere Business Monitor 6.1 format (XML schema support) 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.

Related reference

"Human task events" on page 710
Human task events are sent if monitoring is requested for the elements of the task in WebSphere Integration Developer. Use the information provided here for a detailed description of all of the events, that is, task events and escalation events, that can be emitted by human tasks.

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

- "BPC.HTM.BASE"

BPC.HTM.ESCALATION

- "BPC.HTM.ESCALATION.BASE" on page 703
- "BPC.HTM.ESCALATION.CUSTOMPROPERTYSET" on page 704
- "BPC.HTM.ESCALATION.STATUS" on page 704
- BPC.HTM.ESCALATION.UPDATED
- "BPC.HTM.ESCALATION.WISTATUS" on page 704
- "BPC.HTM.ESCALATION.WITRANSFER" on page 705

BPC.HTM.TASK

- "BPC.HTM.TASK.BASE" on page 705
- "BPC.HTM.TASK.CUSTOMPROPERTYSET" on page 706
- "BPC.HTM.TASK.FAILURE" on page 706
- "BPC.HTM.TASK.FOLLOW" on page 707
- "BPC.HTM.TASK.INTERACT" on page 707
- "BPC.HTM.TASK.MESSAGE" on page 707
- "BPC.HTM.TASK.STATUS" on page 707
- "BPC.HTM.TASK.UPDATED" on page 707
- "BPC.HTM.TASK.WISTATUS" on page 709
- "BPC.HTM.TASK.WITRANSFER" on page 709

BPC.HTM.BASE

BPC.HTM.BASE inherits the XML elements from WBIMonitoringEvent.

Table 121. 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.
activityInstanceId	The ID of the activity instance.
displayName	The display name of the task instance or escalation instance.
expirationDate	The expiration date of the task in Coordinated Universal Time (UTC) ISO 8601 format yyyyMMdd HHmmssZ.
isAdHoc	This has the value true if the task was created at runtime.
isEscalated	This has the value true if the task is escalated.
isFollowOn	This has the value true for a follow-on task.
isSubTask	This has the value true for a subtask.
isSuspended	This has the value true if the task is suspended.
isWaitingForSubTask	This has the value true if the task is waiting for subtask.

Table 121. XML elements for BPC.HTM.BASE (continued)

XML element	Description
kind	This contains one of the following values, which indicate the kind of task: 101 for a human task. 105 for a participating task. 106 for an administrative task.
parentTaskId	The ID of the parent task. If there is no parent task, this is left empty.
principal	The name of the user associated with this event.
processInstanceId	The ID of the process instance.
processTemplateId	The ID of the process template.
state	This contains one of the following values, which indicate the current state of the task instance. 1 - INACTIVE 2 - READY 3 - RUNNING 5 - FINISHED 6 - FAILED 7 - TERMINATE 8 - CLAIMED 12 - EXPIRED 101 - FORWARDED
taskInstanceId	The ID of the task instance.
taskTemplateId	The ID of the template.
taskTemplateName	The name of the task template, including the namespace. This can differ from the display name. For a subtask of a parallel routing task, this value is the name of the parent task template with the string $Child appended to it.
taskTemplateValidFrom	The date and time from when the task template is valid.

BPC.HTM.ESCALATION.BASE

BPC.HTM.ESCALATION.BASE inherits the XML elements from "BPC.HTM.BASE" on page 702.

Table 122. XML elements for BPC.HTM.ESCALATION.BASE

XML element	Description
escalationName	The name of the escalation.
escalationInstanceDescription	The description of the escalation.
escalationTemplateId	The template ID of the escalation.

BPC.HTM.ESCALATION.CUSTOMPROPERTYSET

BPC.HTM.ESCALATION.CUSTOMPROPERTYSET inherits the XML elements from
"BPC.HTM.ESCALATION.BASE" on page 703.

Table 123. XML elements for BPC.HTM.ESCALATION.CUSTOMPROPERTYSET

XML element	Description
username	The name of the user who set the custom property.
propertyName	The name of the custom property.
propertyValue	The value of the custom property.
associatedObjectID	The ID of the associated object that is the escalation instance ID.
query	If isBinary is true, this element specifies the query string for the binary property. Otherwise, this element is not present.
type	If isBinary is true, this element specifies the type of the binary property. Otherwise, this element is not present.
isBinary	Set to false for string custom properties, and to true for binary custom properties. The payload type for binary custom properties is restricted to Empty. The property propertyValue is omitted for binary custom properties.

BPC.HTM.ESCALATION.STATUS

BPC.HTM.ESCALATION.STATUS inherits the XML elements from
"BPC.HTM.ESCALATION.BASE" on page 703. No further specific properties are
defined for BPC.HTM.ESCALATION.STATUS beyond the inherited properties.

BPC.HTM.ESCALATION.UPDATED

BPC.HTM.ESCALATION.UPDATED inherits the XML elements from
"BPC.HTM.ESCALATION.BASE" on page 703.

Table 124. XML elements for BPC.HTM.ESCALATION.UPDATED

XML element	Description
durationUntilEscalated	A calendar-specific duration, after which, the task state is checked and depending on it, the escalation occurs or is superfluous.
durationUntilRepeated	A calendar-specific duration, after which the escalation action is performed again.
escalationTime	The time when this escalation will fire.
name	Name of the escalation.

BPC.HTM.ESCALATION.WISTATUS

BPC.HTM.ESCALATION.WISTATUS inherits the XML elements from
"BPC.HTM.ESCALATION.BASE" on page 703.

Table 125. XML elements for BPC.HTM.ESCALATION.WISTATUS

XML element	Description
username	The names of the users who have work items that are escalated.
reason	The reason that the work item was assigned to the user. This integer value indicates one of the following meanings: REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

BPC.HTM.ESCALATION.WITRANSFER

BPC.HTM.ESCALATION.WITRANSFER inherits the XML elements from "BPC.HTM.ESCALATION.BASE" on page 703.

Table 126. XML elements for BPC.HTM.ESCALATION.WITRANSFER

XML element	Description
current	The name of the current user. This is the user whose work item was transferred to someone else.
target	The name of the user of the work item receiver.
reason	The reason that the work item was transferred. This integer value indicates one of the following meanings: REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

BPC.HTM.TASK.BASE

BPC.HTM.TASK.BASE inherits the XML elements from "BPC.HTM.BASE" on page 702.

Table 127. XML elements for BPC.HTM.TASK.BASE

XML element	Description
taskInstanceDescription	The description of the task.
subTaskLevel	The hierarchy level of a sub-task. The value is 1 for a first level sub-task, 2 for a second level sub-task, and so on.
taskInstanceName	The name of the task instance.

For inline tasks, it has a prefix consisting of the process template name and the dollar symbol.

For a subtask of a parallel routing task, this value is constructed by concatenating the name of the parent task instance with the string $p and an integer that identifies the subtask, for example, *parentTaskName*$p5 for the fifth subtask. |

BPC.HTM.TASK.CUSTOMPROPERTYSET

BPC.HTM.TASK.CUSTOMPROPERTYSET inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 128. XML elements for BPC.HTM.TASK.CUSTOMPROPERTYSET

XML element	Description
username	The name of the user who set the custom property.
propertyName	The name of the custom property.
propertyValue	The value of the custom property.
associatedObjectID	The ID of the associated object that is the task instance ID.
query	If isBinary is true, this element specifies the query string for the binary property. Otherwise, this element is not present.
type	If isBinary is true, this element specifies the type of the binary property. Otherwise, this element is not present.
isBinary	Set to false for string custom properties, and to true for binary custom properties. The payload type for binary custom properties is restricted to Empty. The property propertyValue is omitted for binary custom properties.

BPC.HTM.TASK.FAILURE

BPC.HTM.TASK.FAILURE inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 129. XML elements for BPC.HTM.TASK.FAILURE

XML element	Description
taskFailedException	A string containing the *faultNameSpace* and *faultName* separated by a semicolon (;).
faultName	The name of the fault.

BPC.HTM.TASK.FOLLOW

BPC.HTM.TASK.FOLLOW inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 130. 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.INTERACT

BPC.HTM.TASK.INTERACT inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 131. XML elements for BPC.HTM.TASK.INTERACT

XML element	Description
username	The name of the user that is associated with the task.

BPC.HTM.TASK.MESSAGE

BPC.HTM.TASK.MESSAGE inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 132. XML elements for BPC.HTM.TASK.MESSAGE

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.

BPC.HTM.TASK.STATUS

BPC.HTM.TASK.STATUS inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705. No further specific properties are defined for BPC.HTM.TASK.STATUS beyond the inherited properties.

BPC.HTM.TASK.UPDATED

BPC.HTM.TASK.UPDATED inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 133. XML elements for BPC.HTM.TASK.UPDATED

XML element	Description
businessRelevant	Allows you to distinguish between business relevant and "auxiliary" tasks.
contextAuthorizationOfOwner	Possible values are: • 0 = AUTH_NONE: Indicates that no operations can be performed on the associated context. • 3 = AUTH_READER: Indicates that operations that require Reader authority can be performed on the associated context object, for example, reading the properties of a process instance.
name	The name of the task.
namespace	The namespace used to categorize the task.
description	The description of the task.
displayName	The display name of the task instance.
priority	The priority of the task.
type	The type used to categorize the task.
eventHandlerName	A Java object that handles vetoable events sent to the application component.
durationUntilDeleted	The time period after the task instance reaches an end state, that the instance will be deleted.
deletionTime	Either the scheduled deletion time, or null if no deletion is scheduled.
durationUntilDue	A calendar-specific duration, for how long this task is expected to take.
dueTime	The time when the task is expected to be finished.
durationUntilExpires	A calendar-specific duration, after which the task will expire.
expirationTime	The actual date when this task will expire.
escalated	Indicated whether an escalation occurred for this task.
parentContextID	The parent context for this task. This is the ID the task is dependant on. • For top-level tasks (either the root of a sub-task tree or the root of a follow-on task chain) this is set by the task that creates the application component at creation time and provides a key to the corresponding context in the calling application component. For example, for Business Flow Manager, this can be the PIID, EIID, SIID or AIID. • For sub-tasks this is the ID of the next higher level task instance. • For non-inline tasks this is the ACOID.

Table 133. XML elements for BPC.HTM.TASK.UPDATED (continued)

XML element	Description
supportsClaimIfSuspended	Indicates whether suspended tasks can be claimed.
supportsDelegation	Indicates whether this task can be delegated.
supportsFollowOnTasks	Indicates whether following tasks are supported.
supportsSubTasks	Indicates whether sub-tasks can be invoked for this task.

BPC.HTM.TASK.WISTATUS

BPC.HTM.TASK.WISTATUS inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 134. XML elements for BPC.HTM.TASK.WISTATUS

XML element	Description
username	The names of the users who have work items that were created or deleted.
reason	The reason that the work item was assigned to the user. This integer value indicates one of the following meanings: REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

BPC.HTM.TASK.WITRANSFER

BPC.HTM.TASK.WITRANSFER inherits the XML elements from "BPC.HTM.TASK.BASE" on page 705.

Table 135. XML elements for BPC.HTM.TASK.WITRANSFER

XML element	Description
current	The name of the current user. This is the user whose work item was transferred to someone else.
target	The name of the user of the work item receiver.

Table 135. XML elements for BPC.HTM.TASK.WITRANSFER (continued)

XML element	Description
reason	The reason that the work item was transferred. This integer value indicates one of the following meanings: REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

Related reference

"Human task events"
Human task events are sent if monitoring is requested for the elements of the task in WebSphere Integration Developer. Use the information provided here for a detailed description of all of the events, that is, task events and escalation events, that can be emitted by human tasks.

Human task events

Human task events are sent if monitoring is requested for the elements of the task in WebSphere Integration Developer. Use the information provided here for a detailed description of all of the events, that is, task events and escalation events, that can be emitted by human tasks.

An event is emitted when the state of a task changes. The following types of events can be caused by human tasks:

Note: For tasks that were created at runtime, events are only emitted if the business relevance flag is set to true in the task model.

Inline tasks can emit both human task events and activity events. For a list of the activity events, see "Common Base Events for activities" on page 682.

All human task events can be emitted in both the CEI and the audit trail, with the exception of the task template events. The task template events TASK_TEMPLATE_INSTALLED and TASK_TEMPLATE_UNINSTALLED can only be emitted in the audit trail.

XML Schema Definition (XSD) files

The structure of the events that are sent to CEI is described in the following schema definition file *install_root*\ProcessChoreographer\client\HTMEvents.xsd

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.

Event name and extension name
This column contains two values. The name of the event and the value that is set in the *extensionName* attribute of the Common Base Event. The extension name identifies which event specific information is contained in the Common Base Event, and it is also the name of the XML element that provides additional data about the 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 715.

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	Event name and extension name	Situation	Event nature	Description
51001	TASK_CREATED BPC.HTM.TASK. INTERACT	Report	CREATED	Task created
51002	TASK_DELETED BPC.HTM.TASK.STATUS	Destroy	DELETED	Task deleted
51003	TASK_STARTED BPC.HTM.TASK.STATUS	Start	ENTRY	Task started
51004	TASK_COMPLETED BPC.HTM.TASK.STATUS	Stop	EXIT	Task completed
51005	TASK_CLAIM_ CANCELLED BPC.HTM.TASK.STATUS	Report	DEASSIGNED	Claim canceled

Code	Event name and extension name	Situation	Event nature	Description
51006	TASK_CLAIMED BPC.HTM.TASK. INTERACT	Report	ASSIGNED	Task claimed
51007	TASK_TERMINATED BPC.HTM.TASK.STATUS	Stop	TERMINATED	Task terminated
51008	TASK_FAILED BPC.HTM.TASK. FAILURE	Fail	FAILED	Task failed
51009	TASK_EXPIRED BPC.HTM.TASK.STATUS	Report	EXPIRED	Task expired
51010	TASK_WAITING_FOR_ SUBTASK BPC.HTM.TASK.STATUS	Report	WAITFORSUBTASK	Waiting for subtasks
51011	TASK_SUBTASKS_ COMPLETED BPC.HTM.TASK.STATUS	Stop	SUBTASKCOMPLETED	Subtasks completed
51012	TASK_RESTARTED BPC.HTM.TASK.STATUS	Report	RESTARTED	Task restarted
51013	TASK_SUSPENDED BPC.HTM.TASK.STATUS	Report	SUSPENDED	Task suspended
51014	TASK_RESUMED BPC.HTM.TASK.STATUS	Report	RESUMED	Task resumed
51015	TASK_COMPLETED_ WITH_FOLLOW_ON BPC.HTM.TASK. FOLLOW	Report	COMPLETEDFOLLOW	Task completed and follow-on task started
51101	TASK_UPDATED BPC.HTM.TASK.UPDATED	Report	UPDATED	Task updated
51102	TASK_INPUT_MESSAGE_ UPDATED BPC.HTM.TASK. MESSAGE	Report	INPUTSET	Input message updated. Business object payload is available.
51103	TASK_OUTPUT_ MESSAGE_UPDATED BPC.HTM.TASK. MESSAGE	Report	OUTPUTSET	Output message updated. Business object payload is available.

Code	Event name and extension name	Situation	Event nature	Description
51104	TASK_FAULT_ MESSAGE_UPDATED BPC.HTM.TASK. MESSAGE	Report	FAULTSET	Fault message updated. Business object payload is available.
51201	TASK_WORKITEM_ DELETED BPC.HTM.TASK. WISTATUS	Destroy	WI_DELETED	Work item deleted
51202	TASK_WORKITEM_ CREATED BPC.HTM.TASK. WISTATUS	Report	WI_CREATED	Work items created
51204	TASK_WORKITEM_ TRANSFERRED BPC.HTM.TASK. WITRANSFER	Report	WI_TRANSFERRED	Work item transferred
51205	TASK_WORKITEM_ REFRESHED BPC.HTM.TASK. WISTATUS	Report	WI_REFRESHED	Work items refreshed
51301	TASK_CUSTOMPROPERTY_ SET BPC.HTM.TASK. CUSTOMPROPERTYSET	Report	CP_SET	Custom property set. This event is generated when a custom property of a task instance is changed.
52001	TASK_TEMPLATE_ INSTALLED	Report	INSTALLED	These are task template events, which are only emitted in the audit trail. They are not emitted as common base events, and are included here for completeness.
52002	TASK_TEMPLATE_ UNINSTALLED	Report	UNINSTALLED	

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	Event name and extension name	Situation	Event nature	Description
53001	ESCALATION_UPDATED BPC.HTM.ESCALATION.UPDATED	Report	UPDATED	Escalation updated
53201	ESCALATION_WORKITEM_DELETED BPC.HTM.ESCALATION.WISTATUS	Destroy	WI_DELETED	Work item deleted
53202	ESCALATION_WORKITEM_CREATED BPC.HTM.ESCALATION.WISTATUS	Report	WI_CREATED	Work item created
53204	ESCALATION_WORKITEM_TRANSFERRED BPC.HTM.ESCALATION.WITRANSFER	Report	WI_TRANS-FERRED	Escalation transferred
53205	ESCALATION_WORKITEM_REFRESHED BPC.HTM.ESCALATION.WISTATUS	Report	WI_REFRESH-ED	Work item refreshed
51302	ESCALATION_CUSTOMPROPERTY_SET BPC.HTM.ESCALATION.CUSTOMPROPERTYSET	Report	CP_SET	Custom property set. This event is generated when a custom property of an escalation instance is changed.

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.

Related reference

"Event data specific to human tasks" on page 701
Events are created on behalf of tasks and escalations.

"Extension names for human task events" on page 701
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.

"Situations in human task events"
Human task events can be emitted in different situations. The data for these situations are described in situation elements.

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`

Related reference

"Human task events" on page 710
Human task events are sent if monitoring is requested for the elements of the task in WebSphere Integration Developer. Use the information provided here for a detailed description of all of the events, that is, task events and escalation events, that can be emitted by human tasks.

Part 6. Tuning

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% processor 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 usage.

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, choose to model microflows instead of long-running processes. 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. Because serializing and deserializing data is also expensive, consider using parallel execution paths in your processes instead of serializing activities, and minimizing the size and complexity of data that is part of your process. Also minimize the number of events that are emitted.

4. Tune the processes.

   The way in which you tune an individual process depends on whether it is a long-running process or a microflow.

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 Apache Derby, 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

Long-running business 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.

Related tasks

"Tuning microflows" on page 734
These processes tend to run for only a short time. They use the database only if audit logging or Common Event Infrastructure (CEI) are enabled for the microflow, and to retrieve the template information. The performance of microflows mainly depends on the services that they call.

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 systems to run the database and the application server. For high load or high availability configurations, consider using a WebSphere cluster on several systems for running the business processes, and a separate system for the database.

Procedure

1. On the database server, 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.
  - Fine-tune the heap size of the application server.
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

"Tuning the application server" on page 725
Use this task to tune the application server.

Specifying initial DB2 database settings for the Business Process Choreographer database

Use this task to specify initial DB2 database settings for the Business Process Choreographer database. Note that this information is provided only as an example.

About this task

Attention: The following information relates to the Business Process Choreographer database only.

To achieve good database operation, specify the initial database settings. In addition, use two separate logical disks with different stripe sizes; use a stripe size of 256 KB for table space containers, and a stripe size of 64 KB for database transaction logs. 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 server for the messaging database.

Note: This information does not apply to DB2 UDB for i5/OS 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.

   BPETEMP8K
   > Contains the temporary table space to support the 8 KB buffer pools that are needed when migrating the database schema.

   BPETS8K
   > Contains the table space to support the 8 KB buffer pools that are needed when migrating the database schema.

   INSTANCE
   > Holds process instance and task tables. It is always used intensively, regardless of the kind of long-running process that is run. Where possible, locate this table space on its own disk to separate the traffic from the rest of the process database.

   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 INSTANCE MANAGED BY SYSTEM USING( '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 725.

- 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 setting does not set a database-wide isolation level on the database. This setting is, however, used by the DB2 Configuration Advisor for choosing the best configuration.

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 table space 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 value of the OPEN_CURSORS parameter must be higher than the statement cache size of the Business Process Choreographer data source (BPEDB), which is set to 128 by default. 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 the WebSphere Application Server information center, to set up and tune the data stores for the messaging engines.

Results

Your messaging engines are operating optimally.

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.

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** → **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. Unless a high number of error situations, such as rollbacks occur, set x to 5.

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. Tune the data sources for the data stores that are used by the messaging engines for the Business Process Choreographer, SCA application, SCA system, and CEI buses.

   a. Set **Max Connections** to the value p. Ensure that if one database is used for all messaging engines, the messaging database supports $4 * m$ connections.

   b. Set the **SQL Statement cache** size to 50.

4. 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.

5. 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 720
You can improve the performance of long-running business processes by balancing the hardware resources.

"Specifying initial DB2 database settings for the Business Process Choreographer database" on page 721
Use this task to specify initial DB2 database settings for the Business Process Choreographer database. Note that this information is provided only as an example.

Fine-tuning the Business Process Choreographer database

Use this task to fine-tune the database.

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.

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. If you are using DB2 Version 8, 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. You can improve the performance of complex Business Process Choreographer API queries by turning on reoptimization for prepared statements. This requires creating a "NULLIDR1" package in the Business Process Choreographer database.

   a. 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
         tabname not in ('SAVED_ENGINE_MESSAGE_B_T') AND
         TBSPACEID IN (
             select TBSPACEID from sysibm.systablespaces
             where TBSPACE IN ('INSTANCE', 'WORKITEM', 'BPETS8K'
```

```
                              'STAFFQRY', 'AUDITLOG', 'SCHEDTS'))"
> runStatsScript.cmd

echo db2 connect reset >> runStatsScript.sql
```

Note:

- The select clause "IN ('INSTANCE', ..., 'SCHEDTS')" contains the names of the default table spaces that are created and used when creating the BPEDB database. If in your environment the tables are located in different table spaces, change the select clause accordingly.

- For larger databases with more than 500 000 process instances, you can accelerate the collection of statistics by replacing the `'with distribution and detailed indexes all'` statement with the `'with distribution and sampled detailed indexes all'` statement.

The resulting SQL 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 SQL 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.

b. Run the SQL script by entering the following command:

```
db2 -f runStatsScript.sql
```

c. Create the package "NULLIDR1" in the BPEDB database. Change to the `bnd` directory of your DB2 installation, and enter the following commands:

```
db2 connect to BPEDB
db2 bind db2clipk.bnd collection NULLIDR1
```

d. Customize the BPEDB datasource. Using the administrative console, navigate to the custom properties page for the BPEDB data source, and set the value of the property `currentPackageSet` to `NULLIDR1` .

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_TIMESTAMP) = ON

Unit of work (DFT_MON_UOW) = 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.

Procedure

If you use the service integration capabilities of WebSphere Application Server, refer to the information on tuning the JDBC data source for messaging engines in the WebSphere Application Server information center.

Results

The performance of your messaging provider is improved.

Improving the performance of business process navigation

You can tune the performance of long-running processes by enabling performance optimizations, and tuning various configuration parameters.

About this task

A long-running process spans multiple transactions. Before version 7.0, the default was for transactions to trigger by a Java Messaging Service (JMS) message. With version 7.0 the default is to use work-manager-based navigation, which provides better performance. However, if you migrated from an earlier version, and was using JMS-based navigation, you can improve the performance of process navigation, by configuring Business Flow Manager to use the work-manager-based implementation for triggering transactions. Regardless of whether you use JMS or work-manager navigation mode, you can tune the size of the intertransaction cache.

The following summarizes the characteristics of the two process navigation modes:

JMS message-based navigation
> Process navigation handled by JMS messages that are controlled by the process navigation message-driven bean (MDB).
>
> - If the topology is set up so that the messaging engines are local to the application, a process navigates with server affinity unless it is triggered by external events, such as asynchronous messages or human tasks.
> - If the topology is set up so that multiple servers in an application cluster use a single remote messaging engine, then navigation within a process is distributed over the servers in the cluster.

Work-manager-based navigation
> Process navigation is handled by a thread pool that is controlled by the work manager. Normal navigation of a process instance is done completely with server affinity.

To ensure transactional integrity, the messages that trigger navigation steps are stored in the Business Process Choreographer database. A background recovery thread periodically scans these messages, and if messages exist that are older than a specified maximum age, it sends them to the JMS queue to be picked up by the process navigation MDB. Business Process Choreographer guarantees that each message is executed exactly once.

If an error occurs that causes the navigation step to roll back, the navigation of the process reverts to JMS-controlled navigation.

Server affinity means that navigation within a process instance happens on one WebSphere Application Server, unless an asynchronous service is invoked, a wait or timeout condition is encountered, a receive or pick activity is activated, or a human task is executed. These events can cause navigation within a process to continue on another WebSphere Application Server.

Procedure

1. Configure Business Flow Manager to use work-manager-based process navigation.

   In the administrative console, perform the following steps:

   a. Click either **Servers** › **Clusters** › **WebSphere application server clusters** › *cluster_name* or **Servers** › **Server Types** › **WebSphere application servers** › *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Flow Manager**.

   b. On the **Configuration** tab, select the **Enable advanced performance operations** option. Now you can change the values of the following configuration parameters:

   - Message pool size
   - Maximum age for stalled messages
   - Recovery interval for stalled messages
   - Maximum process time on thread
   - Intertransaction cache size

2. Optional: Increase the maximum number of threads available to the workflow manager.

   Business Flow Manager requires two threads for internal processing. The remaining threads are available for process navigation. Start with one additional thread for each processor. If you increase the thread pool size, you also need to increase the connection pool size for the Business Process Choreographer database (BPEDB) and for the connection factory (BPECFC).

   To change the maximum number of threads, perform the following using the administrative console.

   a. Click **Resources** › **Asynchronous beans** › **Work managers** › **BPENavigationWorkManager**.

   b. Under **Thread pool properties**, change the value of **Maximum number of threads**.

   c. Set the value of **Work request queue size** to be the same as the value of **Maximum number of threads**.

3. Save your changes.
4. Restart the server to activate your changes.

Results

The work manager now controls your process navigation.

Tuning microflows

These processes tend to run for only a short time. They use the database only if audit logging or Common Event Infrastructure (CEI) are enabled for the microflow, and to retrieve the template information. The performance of microflows mainly depends on the services that they call.

About this task

Microflows run in memory, without any user-interaction or persistent messaging support. The processing of a microflow occurs in a single thread, and normally, in a single transaction. 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, refer to the related information on tuning the application server.

2. Tune the JVM garbage collection. The generational garbage collector policy achieves the best throughput. This policy is activated as a generic JVM argument in the JVM settings. Set the initial value of the collection to half of the total heap size. For example, `-Xgcpolicy:gencon -Xmn512M` activates the policy for a heap size of 1024 MB.

   Note: This information does not apply to DB2 UDB for i5/OS.

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, navigate to **Servers** › **Application Servers** › *server_name* › **Add properties** › **Thread pools** › **Default**.

Results

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.

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:

Procedure

- 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

Tuning Business Process Choreographer Explorer

The following suggestions provide various ways to improve the performance of the Business Process Choreographer Explorer.

Procedure

1. Use query tables for customized views.

   Business Process Choreographer Explorer provides default query table definitions for each of the predefined views. However, to exploit the full potential of query tables, you should also use query tables as the basis for your customized views. If you have customized views that do not use query tables, consider creating query tables that contain exactly those properties and filters that you need for your business scenario, and redefining these views based on the new query tables.

   Avoid using customized views that are not based on query tables. Restrict the use of searches that are not based on query tables to one-off searches where you need the flexibility to define specific filter criteria.

2. 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.

3. 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. To change these settings navigate to the following area of the administrative console: **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then in the **Additional Properties** section, click **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. Set this timeout to a low value, for example, 1, so that many users can work concurrently without having to wait for a free thread in the thread pool. A value of 0 indicates no wait time.

4. Decrease the search limit 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. To change this setting, click either **Servers** ▸ **Clusters** ▸ **WebSphere application server clusters** ▸ *cluster_name* or **Servers** ▸ **Server Types** ▸ **WebSphere application servers** ▸ *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Business Process Choreographer Explorer**, then change the setting.

Tuning the Business Choreographer Explorer reporting function

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 the report generation.

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 WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.EVENT_PRC_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_ACT_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_PRC_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.OPEN_EVENTS_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.QUERY_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.SLICES_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
```

For large databases, for example, with more than 500 000 process instances, use the `WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL` statement when you run the RUNSTATS utility.

- 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 database for the Business Choreographer Explorer reporting function was created. For more information about updating the database statistics, refer to the documentation for your database.

Reduce the number of emitted events

In WebSphere Integration Developer, you can define the logging of activities or processes at a very detailed level. Activity audit events are relevant for reporting only 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.

To get an accurate picture of an activity or a process, you should either audit all or none of the event types.

Use the SQL user-defined functions implementation

To create reports, you must install some specific user-defined functions (UDFs) in the Business Process Choreographer Explorer reporting 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.

For more information about the advantages and disadvantages of the SQL and Java implementations, read about selecting a user-defined function.

Use a separate database

If the reporting data is stored in the Business Process Choreographer database (BPEDB), report generation will have a negative impact on runtime performance. You will get better performance if the reporting data is stored in a separate reporting database and you can apply different tuning parameters to each of the databases. Also consider hosting the reporting database on a separate database server.
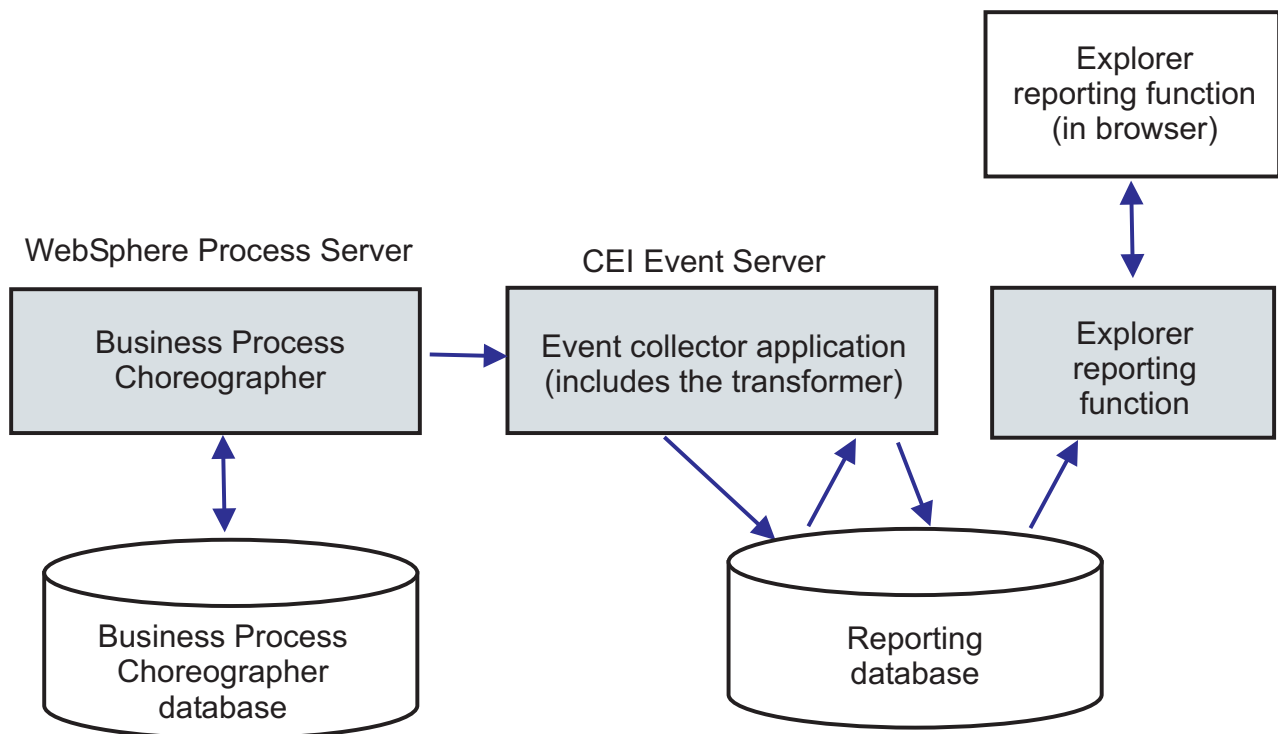
Figure 16. Business Process Choreographer Explorer reporting setup for production performance

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, click **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then in the **Container Settings** section, expand **Container Services** and click **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 by clicking **Resources** → **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 reporting database. Performance is reduced if large amounts of data are queried to produce the report. Report performance can improve if you reduce the number of process and activity instances that are in the reporting database. Regularly deleting unnecessary or old information can help to improve performance.

Part 7. Troubleshooting

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.

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 bpcaugment.wsadmin.log file in the same directory:

- **Windows** *install_root*/logs/manageprofiles/*profileName*/logs directory

- **Linux** **UNIX** *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

The administrative scripts in the admin subdirectory of the ProcessChoreographer directory do not write their own log files. All of the Business Process Choreographer scripts that are run using wsadmin are logged in the application server log files and 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 either use one of the -tracefile or -appendtrace options, or save the 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 setupEventCollector.log file.

- **Windows** setUpEventCollector.bat

- Linux UNIX setUpEventCollector.sh

  Also check the wsadmin.traceout file.

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.

Procedure

- 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 buffer pool. 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 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.

REST API: The URL is not configured correctly

The Representational State Transfer (REST) API must be configured correctly, otherwise you get an error when you try to use the process state view widget in the Business Process Choreographer Explorer or Business Space.

Reason

This can have the following causes:

- If you want to use the graphical process widget in a clustered environment, you must set the endpoints for the Business Flow Manager and Human Task Manager REST APIs manually.
- If you configured the Business Process Choreographer Explorer in a cluster, you must configure the correct host name and port for a Web server to achieve load balancing.
- If you change the context root or map Web modules to a Web server, you might need to change the URL for the REST API.

Resolution

To correct this problem:

- If you configured a Business Process Choreographer Explorer instances, check your log files for messages CWWBZ0052W or CWWBZ0053W, which contain information about the URL that the instance was configured to use.
- If you have multiple Business Process Choreographer configurations in a cell, and the REST API Web modules for the Business Flow Manager (BPEContainer application) and the Human Task Manager (TaskContainer application) are mapped to the same Web server, these Web modules must have unique context roots.

  1. To set the context roots for the Business Flow Manager, click **Applications** → **Application Types** → **WebSphere enterprise applications** then **BPEContainer_*suffix*** → **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules BFMRESTAPI and BFMJAXWSAPI are correct and unique.

  2. To set the context roots for the Human Task Manager, click **Applications** → **Application Types** → **WebSphere enterprise applications** then **TaskContainer_*suffix*** → **Context Root for Web Modules**, where *suffix* is either *node_name_server_name* or the *cluster_name* where Business Process Choreographer is configured. Then make sure that the context root for the Web modules HTMRESTAPI and HTMJAXWSAPI are correct and unique.

6.0.x Business Process Choreographer API client fails in a version 7.0 environment

You did not migrate your 6.0.x Business Process Choreographer API client when you upgraded to WebSphere Process Server Version version 7.0. When you try to run your client in the version 7.0 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 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 Java EE roles BPEAPIUser and TaskAPIUser are set to the value Everyone, which maintains compatibility with earlier versions by maintaining the 6.0.x behavior of not requiring a login when application security is enabled. For new installations these roles default to the value AllAuthenticated. The use of Everyone to map Java EE roles BPEAPIUser and TaskAPIUser is deprecated.

Resolution

Modify your API client to force the user to log on to the client before they 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 `AllAuthenticated` to `Everyone`, and click **OK**.

3. Repeat step 2 for the TaskContainer_suffix and the TaskAPIUser role.

4. After you have modified your client, you must change these roles back to `AllAuthenticated` 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`
  - `Linux` `UNIX` On Linux and UNIX systems, these files are located in the *profile_root*/logs/*server_name* directory.
  - `Windows` 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.

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

This topic describes the symptoms and solutions for problems that can happen when you install an application that contains business processes, human tasks, or both.

Symptom: Exceptions occur after the installation of business processes or human tasks

When you install an application containing business processes, human tasks, or both, you get exceptions similar to the following in the `SystemOut.log` file of the deployment manager or the stand-alone server:

- CWWBF0064E: server1 is not configured to run business process applications
- CWTCO0017E: server1 is not configured to run human task applications

Reason

Neither the business process container nor the human task container is configured on the deployment target.

Resolution

To use business process and human task functionality, you must configure both the business process container and the human task container. For more information about configuring the containers, see the *Related information* section.

Symptom: Application does not start after installation and a successful configuration repository update

An application that contains business processes, human tasks, or both does not start after it was installed successfully. This means that configuration changes were saved in the administrative console or saved using the wsadmin tool.

Reason

The installation of an application which contains business processes or human tasks is divided into two stages. The first stage is finished after the configuration change is saved to the configuration repository. Then the next stage starts. This second stage, called deployment, stores the business process and human task templates found within the application in the Business Process Choreographer

database. This step is started when either the configuration repository is synchronized within a network deployment environment, or when an attempt is made to start this application.

Depending on the number of templates within the application and the hardware you are using, the deployment stage can take some time, and therefore the application does not start.

There might also be an issue during the deployment to Business Process Choreographer database. In this case, examine the logs, traces, and FFDCs to get more information.

Resolution

Depending on the deployment environment, examine the `SystemOut.log` file, the `SystemErr.log` file, and FFDCs.

If the deployment environment is a stand-alone server, look for these logs and FFDCs on that server.

If the deployment environment is a network deployment one, look for these logs and FFDCs on all servers that are part of the deployment target and on all node agents which manage these servers.

If these logs and FFDCs do not indicate an issue, enable the following trace, and contact your support representative for help.

`*=info: com.ibm.bpe.*=all: com.ibm.task.*=all: com.ibm.ws.staffsupport.*=all`

If the deployment environment is a stand-alone server, enable tracing on this server.

If the deployment environment is a network deployment one, enable tracing on all servers that are part of the deployment target and on all node agents which manage these servers.

Symptom: Application does not deploy on a previous level WebSphere Process Server

An application created with a newer version WebSphere Integration Developer does not install on WebSphere Process Server.

Reason

The WebSphere Process Server runtime version must be the same or later than the .EAR file version that you are attempting to install.

Resolution

Use a WebSphere Process Server version which is the same or later than the version of the .EAR file generated by WebSphere Integration Developer. Alternatively, use an appropriate version of WebSphere Integration Developer.

Symptom: Application does not deploy on a mixed version cluster

In a cluster mixed version members, some of which were recently migrated, an application that contains business processes, human tasks, or both cannot be installed, updated, or uninstalled.

Reason

The installation, update, or uninstallation of applications that contain business processes or human tasks is **not** supported in mixed version environments, regardless of the version that you try to install.

Resolution

Finish the migration before you attempt to install, update, or uninstall these applications.

Symptom: Code generation does not work when using shared libraries

When shared libraries are accessed from an application that contains business processes, the application might not install, and it gives an error similar to the following:

```
com.ibm.bpe.plugins.DeploymentCodeGenerationCompileFailedException:
CWWBD0338E: Compiling java code for BPEL file com/ibm/test/bpel/DeployTestBpel.bpel' failed
```

Reason

There is a known limitation with application installation and shared libraries. See the following Technote for details: Technote 1268185.

Troubleshooting the uninstallation of business process and human task applications

This topic describes the symptoms and solutions to problems that can happen when you uninstall an application that contains business processes, human tasks, or both.

Symptom: Application uninstallation failed because instances exist

When you uninstall an application containing business processes, human tasks, or both, you get exceptions similar to the following in the SystemOut.log file of the deployment manager or the stand-alone server:

*   CWTCO0006E: Human Task *task_name* has got instances. Remove the instances before uninstalling the application.
*   CWWBF0025E: Process *process_name* still has instances. Terminate and delete all process instances before updating or uninstalling a process application.

Also, you get similar exceptions in the SystemErr.log file of the deployment manager or the stand-alone server.

Reason

The application you are trying to uninstall contains business processes, human tasks, or both. At least one template of such a business process or human task has instances associated with it. In order to uninstall an application containing business processes, human tasks, or both, associated instances must not exist.

The only exception to this rule is if you are working with a stand-alone server, and this server has the **Run in development mode** option enabled. In this case, you can uninstall applications, although they have existing instances. .

Resolution

Make sure that no instances of these business processes or human tasks exist that are part of your application. Use the Business Process Choreographer Explorer to browse for process instances and task instances and to delete these.

To uninstall your application, follow the instructions in Uninstalling business process and human task applications, using the administrative console or Uninstalling business process and human task applications, using an administrative command.

Symptom: Application uninstallation failed because instances exist although you cannot find them

When you uninstall an application containing business processes, human tasks, or both, the uninstallation failed because business process or human task instances exist that relate to this application although you cannot query for these instances. The application was not uninstalled.

Reason

This kind of issue can occur and it can be difficult to determine a common reason for this failure.

Resolution

Check with your Business Flow Manager system administrator and your Human Task Manager system administrator to make sure that all of the business process and human task instances that belong to your application are deleted. This is also the preferred method in a production environment. To delete completed process instances, use Business Process Choreographer Explorer or the script described in the topic: Deleting completed process instances.

To force the uninstallation of your application with the **-force** option, use the bpcTemplates.jacl script. **Caution: It is not recommended to use the -force option in a production environment.** To use the bpcTemplates.jacl script, follow the instructions in Uninstalling business process and human task applications, using an administrative command. This action deletes all of the existing process and task instances during the uninstallation of the application.

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

An activity is put in a stopped state, if all of the following happen:

- A fault is raised by either the activity's implementation or during the evaluation of a condition, timer, or counter value associated with the activity, for example, its join condition or any of the transition conditions of its outgoing links.
- The fault is not handled on the enclosing scope.
- For invoke activities, inline human tasks, and Java snippets, if either of the following happens:
  - The continueOnError attribute of the process is set to `no` and the continueOnError attribute of the activity is set to `inherit` or `no`.
  - The continueOnError attribute of the process is set to `yes` and the continueOnError attribute of the activity is set to `no`.
- For all other activities, the continueOnError attribute of the process 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.

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 triggered only 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 run only 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. Use the failed event manager console to display details about a failed message and to replay it.
2. Check if there are any failed message 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.

Hung threads when a long-running process is invoked synchronously (Message: WSVR0605W)

A long-running process invokes another long-running process synchronously. Under heavy workload conditions, the thread monitor reports hung threads in the SystemOut.log file (message WSVR0605W).

Reason

A long-running process that is called synchronously can often cause hung threads. A long-running process usually spans several transactions and needs a free thread to continue with its navigation. If all of the available threads are involved in the navigation step of the parent process that invokes the subprocess, the system becomes unresponsive. Because of the lack of free threads, the subprocess cannot complete.

Resolution

A long-running process should always invoke another long-running process asynchronously, even if the processes are separated by another component. For example, if a long-running process invokes a mediation and this mediation invokes another long-running process, then ensure that the preferred interaction style of the mediation is asynchronous.

Late binding calls the wrong version of a subprocess

A parent process invokes a subprocess using late binding. Both processes are in the same module. A new version of the subprocess is created by copying the module and changing the valid-from timestamp. After the module is deployed, the running instances of the parent process continue to invoke the old version of the subprocess instead of the new version.

Reason

In late binding, the process template name of the subprocess is specified as part of the reference partner properties of the invoke activity in the parent process. Business Process Choreographer determines the version of the process that is currently valid at runtime.

A common reason for late binding using the wrong version of a subprocess is that the module that contains the subprocess does not have a Service Component Architecture (SCA) export. Without an export, processes in other modules are not visible to the parent process and it always invokes the version of the subprocess that is in the same module.

Resolution

In the assembly editor in WebSphere Integration Developer, generate an SCA export with SCA native binding for the new version of the subprocess.

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 specifying an initial value when you define the variable, specifying 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.

A fault is not caught by the fault handler

A fault handler is attached to an invoke activity to catch specific faults that are thrown by the invoked service. However, even if the invoked service returns the expected fault, the fault handler is not run.

Reason

A common reason for this problem is that the fault handler does not have a fault variable to catch the data that is associated with the fault. If a fault has associated fault data, it is caught by a fault handler only when one of the following situations apply:

- The name of the fault handler matches the fault name and it has a fault variable with a data type that matches the type of the data associated with the fault
- The fault handler does not specify a fault name but it has a fault variable with a data type that matches the type of the data associated with the fault
- The catchAll fault handler is specified

Resolution

Add a fault variable to the fault handler. Ensure that the data type of the fault variable matches the type of the data that is associated with the fault.

Related concepts

"Retrieval of fault data for business processes" on page 36
Your process can handle runtime faults and BPEL standard faults. To handle these faults, you might need access to the information about the fault.

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, Oracle, or Informix as the database system, the process engine will serialize the execution of parallel paths. You cannot change this behavior. This is because the locks on database entities for these database systems are not as granular as, for example, those for DB2 databases. However, services that are triggered asynchronously by parallel branches still run in parallel; it is only the process navigation that is serialized for these database systems.

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.

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:

   - Windows *install_root*/bin/waslogbr.bat

   - Linux UNIX *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

      - Windows *profile_root*\profiles*profile_name*\logs\activity.log
      - Linux UNIX *profile_root*/profiles/*profile_name*/logs/ activity.log
      - *profile_root*/profiles/*profile_name*/logs/activity.log

   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.

Procedure

- 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, the 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.

Include a "while" loop for states other than the finished state. 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.

Procedure

- 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** then *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 either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.
  2. On 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 either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.
  2. 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 either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.
  2. On the **Configuration** tab, in the **Additional Properties**section, click **Custom Properties**.
  3. 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:
- User cannot administer process, scope, or activity instances
- 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
- Enabling additional messages about people assignment decisions
- Issues with group work items and the "Group" people assignment criteria
- Cleanup of stored people assignment results
- Adapted XSL transformation file has no effect

You can also search for additional information in the Technical support search page.

User cannot administer or monitor process, scope, or activity instances, and no administrative tasks are created

If process administration is restricted to system administrators, instance-based administration is disabled, and all administrative actions on processes, scopes, and activities are limited to users in the BPESystemAdministrator role. For more information about this administration mode, see "Alternate process administration authorization mode" on page 50.

If the Business Flow Manager has been switched to run in the alternate mode, you might need to perform one of the following actions:
- Make sure that all users and programs that perform administrative actions are using user IDs that are in the appropriate role. For example, BPESystemAdministrator or BPESystemmonitor.
- Restore instance-based administration, by turning the alternate process administration authorization mode off.

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 sample XSL transformation file used for the LDAP people directory provider is `LDAPTransformation.xsl`.

- **Linux** **UNIX** On Linux and UNIX platforms, this file is located in `install-root/ProcessChoreographer/Staff`.

- **Windows** On Windows platforms, this file is located in `install-root\ProcessChoreographer\Staff`.

The default Threshold value is `1000000`, therefore by default the threshold value is of no realistic importance. Do not lower this value without careful consideration.

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">1000000</xsl:variable>`

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 all or 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 136. 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	An exception occurs and the process is not started: EngineAdministratorCannotBe ResolvedException
Process reader	No reader	No reader

Table 137. 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 starter; applies to invocation tasks only	Everybody becomes potential starter	An exception occurs and the process is not started
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 138. 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	An exception is thrown and the task is not created
Task potential starter	Originator becomes potential starter	An exception is thrown and the task is not started

Table 138. Roles for stand-alone human tasks and their escalations (continued)

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 ...
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 administrative 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 administrative security is enabled. In the administrative console, go to **Security** ▸ **Global security** and make sure the **Enable administrative 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.

Enabling additional messages about people assignment decisions

You can set a custom property to log additional messages in the SystemOut.log. The messages record the following events:

- If people resolution did not find any users for a task role, and default users were selected.

- If you are using VMM, warnings when specified entities or specific attributes could not be found in the VMM people directory.

- If you are using substitution, logs decisions whether or not users have been substituted.

Because these messages can significantly increase the amount of data in SystemOut.log, only enable these additional messages for testing or debugging purposes.

To enable the staff diagnosis feature perform the following:

1. Using the administrative console, click either **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* or **Servers** → **Server Types** → **WebSphere application servers** → *server_name*, then on the **Configuration** tab, in the **Business Integration** section, expand **Business Process Choreographer**, and click **Human Task Manager**.

2. On the **Configuration** tab, set the value for the custom property `Staff.Diagnosis` to one of the following values:

   off   Never writes additional people assignment information.

   on   Always writes additional people assignment information.

   development_mode
   Only writes additional people assignment information when

the server is running in development mode. this is the default value. By default, the WebSphere Test Environment runs in development mode.

3. Restart the server.

The following messages are generated:

- `Core.StaffDiagMsgIsEnabled=CWTKE0057I: The output of people (staff) resolution diagnosis messages is enabled.` Indicates that the diagnosis feature is enabled. This message is generated when the Human Task Manager is started.

- `Core.EverybodyIsPotInstanceCreator=CWTKE0047I: Everybody is potential instance creator for task {0}.` Indicates that Everybody became the potential instance creator because no potential instance creator is defined.

- `Core.OriginatorBecomesPotStarter=CWTKE0046I: Originator becomes potential starter of task {0}.` For standalone tasks only: Indicates that the originator became the potential starter because no potential starter is defined.

- `Core.EverybodyIsPotentialStarter=CWTKE0045I: Everybody is potential starter of task {0}.` For inline tasks only: Indicates that Everybody became the potential starter because no potential starter is defined.

- `Core.OriginatorBecomesAdministrator=CWTKE0044I: Originator becomes administrator of task {0}.` Indicates that the originator became the administrator because no administrator is defined.

- `Core.EscalationReceiverDoesNotExist=CWTKE0043W: Administrator(s) will be the escalation receiver(s) of the escalation {0}.` Indicates that the administrators became the escalation receivers because staff resolution for the escalation receivers either failed or returned an empty list. If no escalation receiver is defined, the default is Everybody, and a trace message is written.

- `Core.EverybodyIsPotentialOwner=CWTKE0014I: Everybody is potential owner of task {0}.` Indicates that Everybody became the potential owner because no potential owner is defined.

- `Core.PotentialOwnerDoesNotExist=CWTKE0015W: Administrator(s) will be the potential owner(s) of the task {0}.` Indicates that the administrators became the potential owners because staff resolution for the potential owners either failed or returned an empty list. If no potential owner is defined, the default is Everybody, and a trace message is written.

- `StaffPlugin.VMMEntityNotFound=CWWBS0457W: The VMM entity could not be found, received VMM message is ''{0}''.` Indicates that a specified VMM entity (a group or person) was not found in the people directory and the reason. People or groups that cannot be found in the people directory are not included in the people resolution result.

- `StaffPlugin.VMMEntityAttributeNotFound=CWWBS0454W: VMM entity ''{0}'' has no attribute with name ''{1}'' of type ''{2}''.` Indicates that a specified attribute was not found when searching for a VMM entity (person) in the people directory. If no user e-mail address is found, the user cannot receive e-mail notifications for escalations. If no user `preferredLanguage` is found, the default language setting is used. If no substitution attributes (`isAbsent` or `substitutes`) are found when

reading, an attempt is made to initialize the attributes. If no substitution attributes are found when writing or updating, an exception is generated.

- `StaffPlugin.VMMResultIsEmpty=CWWBS0456W: The VMM invocation returned no requested result entities.` Indicates that a (get or search) invocation of VMM did not return any entities. No users are included in the people resolution result.

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.

Adapted XSL transformation file has no effect

When adapting an XSL transformation file, the server needs to be restarted before the changes take effect. In addition, the adapted XSL file is applied only to newly deployed processes and tasks. The changes have no effect on processes and tasks that have been deployed before the XSL file was changed.

Troubleshooting Business Process Choreographer Explorer

Use this information to solve problems relating to Business Process Choreographer Explorer.

About this task

Use the following information to solve problems relating to accessing or using Business Process Choreographer Explorer.

Errors while trying to access Business Process Choreographer Explorer from a browser

> 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 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.

Error message when using 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 CWWBc*nnnn*c, 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**.

Error message StandardFaultException with the standard fault missingReply (message CWWBE0071E)

> If you get a StandardFaultException error 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 761.

Some items not displayed when you log on to Business Process Choreographer Explorer

> If you can log on to 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. Depending on the authorization granted to your user ID, the administrative views and options are not visible, or they are not enabled.
> - Use WebSphere Integration Developer to check or modify the authorization settings defined in the business process.

If the Reports tab is not displayed, contact your system administrator and check that Business Process Choreographer Explorer is configured, including the reporting function.

Error message CWWBU0001E or a communication error with the HTMConnection function

If you get the error message CWWBU0001E: "A communication error occurred when the BFMConnection function was called" or "A communication error occurred when the HTMConnection function was called", use the following information to help resolve the problem.

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

If you get the error message WWBU0024E: "Could not establish a connection to local business process EJB" with a reason "Naming Exception", use the following information to help resolve the problem.

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.

Troubleshooting Business Process Choreographer Explorer reports

Refer to the information in this topic if you are experiencing difficulty with Business Process Choreographer Explorer reports.

Symptom: Setup of the reporting database using the create tables option fails with error message CWWBO4013E

In the System.out you see the following messages:
- CWWBO4015W: The Business Process Choreographer Explorer reporting database schema is incomplete. Use menu option 6 of $WAS_HOME/ ProcessChoreographer/config/setupEventCollector to install the JAR file.
- CWWBO4013E: The bpcodbutil.jar file could not be found on the Derby network server.

Reason

The setup of the reporting database uses the Derby working directory to install a UDF JAR file on the server. If the Derby network server has an incorrect working directory, the JAR file cannot be found.

Resolution

Start the Derby network server from the networkServer subdirectory as follows:
1. If the Derby network server is running, stop it.
2. On a command line, change to the directory $WAS_HOME/derby/bin/ networkServer.
3. Restart the Derby network server, for example, using startNetworkServer.bat.

4. Restart the Business Process Choreographer Explorer application, which triggers the tables creation again.

Symptom: No events displayed on the Reports tab of Business Process Choreographer Explorer

The reporting database of Business Process Choreographer Explorer does not contain any events, or the events are not transformed yet. Various reasons for this are provided in the following sections, including possible resolutions.

Reason

Events are transformed correctly, but do not show up in the reporting database of Business Process Choreographer Explorer.

Resolution

If the trace log contains trace entries for the message that events are received, and the startTransform message, but you do not see any events in Business Process Choreographer Explorer, check that the Business Process Choreographer Explorer and event collector are using the same data sources.

1. Using the administrative console, click **Applications** → **Application Types** → **WebSphere enterprise applications**, then select the BPCExplorer application, and click **Resource references**.
2. Note the value for the **Target Resource JNDI Name** of the modules. Typically, this has the value jdbc/BPEDB.
3. Repeat this and compare the value for the Event Collector application.
4. If they are not identical, then make it so.

Reason

No events are received because the CEI service is not enabled on the server.

Resolution

In the administrative console, click **Servers** → **Application servers** → *server_name*, then in the the **Business Integration** section, expand **Common Event Infrastructure**, and click **Common Event Infrastructure Destination**, then make sure that the check box **Enable service at server startup** is selected.

Reason

CEI logging is not enabled for the business process container.

Resolution

Make sure that CEI logging is enabled for the business process container. Refer to "Enabling logging for Business Process Choreographer" on page 317 to enable CEI logging.

Reason

The Common Event Infrastructure event server or the Business Process Choreographer event collector are not running.

Resolution

Use the administrative console to check that the Common Event Infrastructure event server and the Business Process Choreographer event collector are running.

Reason

Event monitoring for your business processes is disabled.

Resolution

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.

Reason

The event transformer is not triggered.

Resolution

Reduce the threshold setting for the event collector, as described in the documentation about changing configuration parameters for Business Process Choreographer Explorer reporting. Then create new events, which trigger the event collector.

Reason

Events are generated, and are visible in the CBE browser, but no events are shown in the reporting database of Business Process Choreographer Explorer because event distribution is disabled on the event server.

Resolution

In the administrative console, click **Service integration** ⟶ **Common Event Infrastructure** ⟶ **Event service** ⟶ **Event services** ⟶ **Default Common Event Infrastructure event server**, and make sure that **Enable event distribution** is selected.

Reason

Inappropriate configuration settings of Business Process Choreographer event collector prevent data from being visible in the reporting database of Business Process Choreographer Explorer.

Resolution

Call the setupEventCollector configuration script to change the Business Process Choreographer event collector configuration settings for BPCEventTransformerEventCount, BPCEventTransformerMaxWaitTime, and BPCEventTransformerToleranceTime. Refer to the documentation about changing configuration parameters for Business Process Choreographer Explorer reporting for more information about changing the Business Process Choreographer event collector configuration settings.

Reason

The BFMEvents event group must be defined.

Resolution

In the administrative console, click **Service integration** → **Common Event Infrastructure** → **Event service** → **Event services** → **Default Common Event Infrastructure event server** → **Event groups**, then check whether the group BFMEvents exists.

- If the group does not exist, install the event collector application again.
- If the event group exists, check the selector string. Typically it is set to the following string: `CommonBaseEvent[starts-with(@extensionName,'BPC.BFM.')]`

Symptom: The number of displayed events is less than the number expected

The reporting database of Business Process Choreographer Explorer does not contain any events, or the events are not transformed yet. Various reasons for this are provided in the following sections, including possible resolutions.

Reason

The emitted events are not supported. You can verify this using the trace facility. Enable the trace for `com.ibm.bpe.observer.*`. In the trace, look for messages similar to this: `Event Code` *eventCode* `is not relevant for Observer. Discarding event..` If you see such a message, the named event is ignored by the event collector.

Resolution

Ensure that events that will be emitted are supported, otherwise they will not be recognized.

Reason

Events are purged because they cannot be associated. The process started event must be emitted in every case, otherwise the events that are triggered by the activities are purged.

To verify if events are purged due to missing predecessor events, check for the message `CWWBO0014I: A Process Started event was not found for the process instance with the PIID '`*nnnnnn*`'. Discarding events.` *nnnnnn* is the identifier of the process instance.

If the problem continues

- Check the system log file SystemOut.log of the server for error messages.
- Check the deployment and configuration of the Business Process Choreographer event collector and of Business Process Choreographer Explorer. To check the configuration settings, use the administrative console or the clientconfig.jacl configuration script. For further information on how to change the Business Process Choreographer event collector configuration settings, refer to the documentation about changing configuration parameters for Business Process Choreographer Explorer reporting.

- Enable the tracing facility for reporting in the administrative console: Click **Troubleshooting** → **Logs and trace** → *server_name* → **Diagnostic Trace Service** → **Change Log Detail Levels**. Set detail level `all` for `com.ibm.bpe.observer.*`, and restart the BPCECollector and the BPCExplorer applications.

Part 8. Appendixes

Appendix. Database views for Business Process Choreographer

This reference information describes the columns in the predefined database views.

ACTIVITY view

Use this predefined Business Process Choreographer database view for queries on activities.

Table 139. 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.
SIID	ID	The scope instance ID.
STID	ID	The ID of the scope of the template.
EHIID	ID	The ID of the event handler instance if this activity is part of an event handler.
ENCLOSING_FEIID	ID	The ID of the enclosing forEach activity if this activity is nested in another forEach activity.
KIND	Integer	The kind of activity. Possible values are: KIND_INVOKE (21) KIND_RECEIVE (23) KIND_REPLY (24) KIND_THROW (25) KIND_RETHROW (46) KIND_TERMINATE (26) KIND_WAIT (27) KIND_COMPENSATE (29) KIND_SEQUENCE (30) KIND_EMPTY (3) KIND_SWITCH (32) KIND_WHILE (34) KIND_PICK (36) KIND_FLOW (38) KIND_SCOPE (40) KIND_SCRIPT (42) KIND_STAFF (43) KIND_ASSIGN (44) KIND_CUSTOM (45) KIND_FOR_EACH_PARALLEL (49) KIND_FOR_EACH_SERIAL (47) KIND_REPEAT_UNTIL (52)
COMPLETED	Timestamp	The time the activity is completed.
ACTIVATED	Timestamp	The time the activity is activated.

Table 139. Columns in the ACTIVITY view (continued)

Column name	Type	Comments
FIRST_ACTIVATED	Timestamp	The time at which the activity was activated for the first time.
STARTED	Timestamp	The time the activity is started.
PREVIOUS_EXPIRATION_ TIME	Timestamp	The previous expiration time for the activity.
STATE	Integer	The state of the activity. Possible values are: STATE_INACTIVE (1) STATE_READY (2) STATE_RUNNING (3) STATE_SKIPPED (4) STATE_FINISHED (5) STATE_FAILED (6) STATE_TERMINATED (7) STATE_CLAIMED (8) STATE_TERMINATING (9) STATE_FAILING (10) STATE_WAITING (11) STATE_EXPIRED (12) STATE_STOPPED (13) STATE_PROCESSING_UNDO (14)
SUBSTATE	Integer	The substate of the activity when the process instance was migrated. Possible values are: SUB_STATE_NONE (0) SUB_STATE_EXPIRING (1) SUB_STATE_SKIPPING (2) SUB_STATE_RESTARTING (3) SUB_STATE_FINISHING (4) SUB_STATE_FAILING (5)
STOP_REASON	Integer	The reason why the activity stopped. Possible values are: STOP_REASON_UNSPECIFIED (1) STOP_REASON_ACTIVATION _FAILED (2) STOP_REASON_IMPLEMENTATION _FAILED (3) STOP_REASON_FOLLOW_ON _NAVIGATION_FAILED (4) STOP_REASON_EXIT_CONDITION _FALSE (5)
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.

Table 139. Columns in the ACTIVITY view (continued)

Column name	Type	Comments
BUSINESS_RELEVANCE	Boolean	Specifies whether the activity is business relevant. Possible values are: **TRUE** The activity is business relevant. You can view the activity status in Business Process Choreographer Explorer. **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.
INVOKED_INST_ID	Integer	The instance ID of the invoked process or task. You can use the value of the INVOKED_INSTANCE_TYPE column to determine the instance type.
INVOKED_INST_TYPE	Integer	The type of the instance ID in the INVOKED_INST_ID column. Possible values are: INVOKED_INSTANCE_TYPE_ NOT_SET (0) INVOKED_INSTANCE_TYPE_ INLINE_TASK (1) INVOKED_INSTANCE_TYPE _CHILD_TASK (2) INVOKED_INSTANCE_TYPE _CHILD_PROCESS (3)
SKIP_REQUESTED	Boolean	Specifies whether the activity is marked for skipping.
CONTINUE_ON_ERROR	Boolean	Specifies what happens to a process if an unexpected fault is raised and a fault handler is not defined for that fault. This column is Initialized with the corresponding value from the activity template but can be overwritten by the forceComplete and forceRetry APIs. Possible values are: **True** Standard fault handling is applied. **False** Navigation of the process stops so that the process can be repaired.

ACTIVITY_ATTRIBUTE view

Use this predefined Business Process Choreographer database view for queries on custom properties for activities.

Table 140. 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.
DATA_TYPE	String	The class type for non-string custom properties.

ACTIVITY_SERVICE view

Use this predefined Business Process Choreographer database view for queries on activity services.

Table 141. 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.
NAME_SPACE_URI	String	The URI of the namespace.
OPERATION	String	The operation name of the service.

APPLICATION_COMP view

Use this predefined Business Process Choreographer database view to query the application component ID and default settings for tasks.

Table 142. Columns in the APPLICATION_COMP view

Column name	Type	Comments
ACOID	ID	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: **TRUE** The task is business relevant and it is audited. **FALSE** The task is not business relevant and it is not audited.

Table 142. Columns in the APPLICATION_COMP view (continued)

Column name	Type	Comments
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.

AUDIT_LOG_B view

Use this predefined Business Process Choreographer database view for queries on audit log information for business processes.

Inline tasks are logged in both the AUDIT_LOG_B view and in the TASK_AUDIT_LOG view. For example, claiming an inline participating task results in a TASK_CLAIMED event and an ACTIVITY_CLAIMED event. All other task types are only logged in the TASK_AUDIT_LOG view. Depending on the choices made in WebSphere Integration Developer, inline human tasks can also trigger Common Event Infrastructure (CEI) events to be emitted for Business Flow Manager staff activity events and Human Task Manager task events.

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).

The following table describes the AUDIT_LOG_B view. It lists the names of the columns, the event types, and gives a short description for the column.

Table 143. 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 143. 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: KIND_EMPTY 3 KIND_INVOKE 21 KIND_RECEIVE 23 KIND_REPLY 24 KIND_THROW 25 KIND_TERMINATE 26 KIND_WAIT 27 KIND_COMPENSATE 29 KIND_SEQUENCE 30 KIND_SWITCH 32 KIND_WHILE 34 KIND_PICK 36 KIND_FLOW 38 KIND_SCRIPT 42 KIND_STAFF 43 KIND_ASSIGN 44 KIND_CUSTOM 45 KIND_RETHROW 46 KIND_FOR_EACH_SERIAL 47 KIND_FOR_EACH_PARALLEL 49 KIND_REPEAT_UNTIL 52 These are the constants defined for ActivityInstanceData.KIND_*
ACTIVITY_STATE			x				State of the activity that is related to the event. Possible values are: STATE_INACTIVE 1 STATE_READY 2 STATE_RUNNING 3 STATE_SKIPPED 4 STATE_FINISHED 5 STATE_FAILED 6 STATE_TERMINATED 7 STATE_CLAIMED 8 STATE_TERMINATING 9 STATE_FAILING 10 STATE_WAITING 11 STATE_EXPIRED 12 STATE_STOPPED 13 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 143. Structure of the AUDIT_LOG_B audit trail view (continued)

Name	PTE	PIE	AIE	VAR	CLE	SIE	Description
EXCEPTION_TEXT		x	x			x	Exception message that caused an activity or process to fail. Applicable for: PROCESS_FAILED ACTIVITY_FAILED SCOPE_FAILED
DESCRIPTION		x	x	x	x	x	Description of activity or process, containing potentially resolved replacement variables.
CORR_SET_INFO		x					The string representation of the correlation set that was initialized at process start time. Provided with the processCorrelationSetInitialized event (42027).
USER_NAME		x	x				The name of the user whose work item has been changed. This is applicable for the following events: • Process instance work item deleted • Activity instance work item deleted • Process instance work item created • Activity instance work item created

Table 143. Structure of the AUDIT_LOG_B audit trail view (continued)

Name	PTE	PIE	AIE	VAR	CLE	SIE	Description
ADDITIONAL_ INFO		x	x			x	The contents of this field depends on the type of the event:
							ACTIVITY_WORKITEM_ TRANSFERRED, PROCESS_WORK_ITEM_ TRANSFERRED The name of the user that received the work item.
							ACTIVITY_WORKITEM_ CREATED, ACTIVITY_WORKITEM_ REFRESHED, ACTIVITY_ESCALATED 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).
							PROCESS_EVENT_RECEIVED, SCOPE_EVENT_RECEIVED 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.
							ACTIVITY_CHILD_ PROCESS_TERMINATING The substate of the activity when the process instance was migrated. Possible values are: SUB_STATE_NONE (0) SUB_STATE_EXPIRING (1) SUB_STATE_SKIPPING (2) SUB_STATE_RESTARTING (3) SUB_STATE_FINISHING (4) SUB_STATE_FAILING (5)

ESCALATION view

Use this predefined Business Process Choreographer database view to query data for escalations.

Table 144. Columns in the ESCALATION view

Column name	Type	Comments
ESIID	ID	The ID of the escalation instance.
ACTION	Integer	The action triggered by the escalation. Possible values are: **ACTION_CREATE_WORK_ITEM (1)** Creates a work item for each escalation receiver. **ACTION_SEND_EMAIL (2)** Sends an e-mail to each escalation receiver. **ACTION_CREATE_EVENT (3)** Creates and publishes an event.
ACTIVATION_STATE	Integer	An escalation instance is created if the corresponding task reaches one of the following states: **ACTIVATION_STATE_READY (2)** Specifies that the human or participating task is ready to be claimed. **ACTIVATION_STATE_RUNNING (3)** Specifies that the originating task is started and running. **ACTIVATION_STATE_CLAIMED (8)** Specifies that the task is claimed. **ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)** 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: **AT_LEAST_EXPECTED_STATE_CLAIMED (8)** Specifies that the task is claimed. **AT_LEAST_EXPECTED_STATE_ENDED (20)** Specifies that the task is in a final state (FINISHED, FAILED, TERMINATED or EXPIRED). **AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)** Specifies that all of the subtasks of the task are complete.
ESCALATION_TIME	Timestamp	The time when the escalation is to be raised.
ESTID	ID	The ID of the corresponding escalation template.
FIRST_ESIID	ID	The ID of the first escalation in the chain.

Table 144. Columns in the ESCALATION view (continued)

Column name	Type	Comments

INCREASE_PRIORITY	Integer	Indicates how the task priority will be increased. Possible values are:
		
		INCREASE_PRIORITY_NO (1)
		        The task priority is not increased.
		
		INCREASE_PRIORITY_ONCE (2)
		        The task priority is increased once by one.
		
		INCREASE_PRIORITY_REPEATED (3)
		        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: STATE_INACTIVE (1) STATE_WAITING (2) STATE_ESCALATED (3) STATE_SUPERFLUOUS (4)
TKIID	ID	The task instance ID to which the escalation belongs.

ESCALATION_CPROP view

Use this predefined Business Process Choreographer database view to query custom properties for escalations.

Table 145. Columns in the ESCALATION_CPROP view

Column name	Type	Comments
ESIID	ID	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 Business Process Choreographer database view to query multilingual descriptive data for escalations.

Table 146. Columns in the ESCALATION_DESC view

Column name	Type	Comments
ESIID	ID	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 147. Columns in the ESC_TEMPL view

Column name	Type	Comments
ESTID	ID	The ID of the escalation template.
ACTION	Integer	The action triggered by the escalation. Possible values are: **ACTION_CREATE_WORK_ITEM (1)** Creates a work item for each escalation receiver. **ACTION_SEND_EMAIL (2)** Sends an e-mail to each escalation receiver. **ACTION_CREATE_EVENT (3)** Creates and publishes an event.
ACTIVATION_STATE	Integer	An escalation instance is created if the corresponding task reaches one of the following states: **ACTIVATION_STATE_READY (2)** Specifies that the human or participating task is ready to be claimed. **ACTIVATION_STATE_RUNNING (3)** Specifies that the originating task is started and running. **ACTIVATION_STATE_CLAIMED (8)** Specifies that the task is claimed. **ACTIVATION_STATE_WAITING_FOR_SUBTASK (20)** 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: **AT_LEAST_EXPECTED_STATE_CLAIMED (8)** Specifies that the task is claimed. **AT_LEAST_EXPECTED_STATE_ENDED (20)** Specifies that the task is in a final state (FINISHED, FAILED, TERMINATED or EXPIRED). **AT_LEAST_EXPECTED_STATE_SUBTASKS_COMPLETED (21)** 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	ID	The ID of the first escalation template in a chain of escalation templates.

Table 147. 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:
		
		INCREASE_PRIORITY_NO (1)
		    The task priority is not increased.
		
		INCREASE_PRIORITY_ONCE (2)
		    The task priority is increased once by one.
		
		INCREASE_PRIORITY_REPEATED (3)
		    The task priority is increased by one each time the escalation repeats.

NAME	String	The name of the escalation template.
PREVIOUS_ESTID	ID	The ID of the previous escalation template in a chain of escalation templates.
TKTID	ID	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 148. Columns in the ESC_TEMPL_CPROP view

Column name	Type	Comments
ESTID	ID	The ID of the escalation template.
NAME	String	The name of the property.
TKTID	ID	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 149. Columns in the ESC_TEMPL_DESC view

Column name	Type	Comments
ESTID	ID	The ID of the escalation template.
LOCALE	String	The name of the locale associated with the description or display name.
TKTID	ID	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.

MIGRATION_FRONT view

Use this predefined Business Process Choreographer database view to query where the process instance was in its navigation when it was migrated to the new version of the process template. The migration front represents the activities that were either active or just completed when the migration occurred.

Table 150. Columns in the MIGRATION_FRONT view

Column name	Type	Comments
PIID	ID	The process instance ID.
SOURCE_PTID	ID	The ID of the process template that is associated with the process instance before migration.
TARGET_PTID	ID	The ID of the process template that is associated with the process instance after migration.
AIID	ID	The ID of an activity that was part of the process navigation front when the process was migrated. The migration front consists of the last activity reached by the process navigation on each parallel branch when the process was migrated.
SOURCE_ATID	ID	The ID of the activity template before migration.
TARGET_ATID	ID	The ID of the activity template after migration. If the activity was completed before the process instance was migrated, the activity template ID does not change.
MIGRATION_TIME	Timestamp	The time that the process instance was migrated.
STATE	Integer	The state of the activity when the process instance was migrated. Possible values are: STATE_READY (2) STATE_RUNNING (3) STATE_SKIPPED (4) STATE_FINISHED (5) STATE_FAILED (6) STATE_TERMINATED (7) STATE_CLAIMED (8) STATE_TERMINATING (9) STATE_FAILING (10) STATE_WAITING (11) STATE_EXPIRED (12) STATE_STOPPED (13)
SUBSTATE	Integer	The substate of the activity when the process instance was migrated. Possible values are: SUB_STATE_NONE (0) SUB_STATE_EXPIRING (1) SUB_STATE_SKIPPING (2) SUB_STATE_RESTARTING (3) SUB_STATE_FINISHING (4) SUB_STATE_FAILING (5)

Table 150. Columns in the MIGRATION_FRONT view (continued)

Column name	Type	Comments
STOP_REASON	Integer	The reason why the activity stopped. Possible values are: STOP_REASON_UNSPECIFIED (1) STOP_REASON_ACTIVATION _FAILED (2) STOP_REASON_IMPLEMENTATION _FAILED (3) STOP_REASON_FOLLOW_ON _NAVIGATION_FAILED (4) STOP_REASON_EXIT_CONDITION _FALSE (5)

PROCESS_ATTRIBUTE view

Use this predefined Business Process Choreographer database view for queries on custom properties for processes.

Table 151. 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.
DATA_TYPE	String	The class type for non-string custom properties.

PROCESS_INSTANCE view

Use this predefined Business Process Choreographer database view for queries on process instances.

Table 152. 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.

Table 152. Columns in the PROCESS_INSTANCE view (continued)

Column name	Type	Comments
STATE	Integer	The state of the process instance. Possible values are: STATE_READY (1) STATE_RUNNING (2) STATE_FINISHED (3) STATE_COMPENSATING (4) STATE_INDOUBT (10) STATE_FAILED (5) STATE_TERMINATED (6) STATE_COMPENSATED (7) STATE_COMPENSATION_FAILED (12) STATE_TERMINATING (8) STATE_FAILING (9) 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.
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.
CONTINUE_ON_ ERROR	Boolean	Specifies what happens to a process if an unexpected fault is raised and a fault handler is not defined for that fault. Possible values are: **True** Standard fault handling is applied. **False** Navigation of the process stops so that the process can be repaired.
IS_MIGRATED	Boolean	Specifies whether the process instance has been migrated from an older version of the process. If this attribute is set to TRUE, the process instance has been migrated.

PROCESS_TEMPLATE view

Use this predefined Business Process Choreographer database view for queries on process templates.

Table 153. 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: STATE_STARTED (1) STATE_STOPPED (2)
EXECUTION_MODE	Integer	Specifies how process instances that are derived from this process template can be run. Possible values are: EXECUTION_MODE_MICROFLOW (1) 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. Possible values are: COMP_SPHERE_REQUIRED (2) COMP_SPHERE_SUPPORTS (4)
DISPLAY_NAME	String	The descriptive name of the process.
CAN_RUN_SYNC	Boolean	Specifies whether a process can be invoked by the call method.
CAN_RUN_INTERRUP	Boolean	Specifies whether a process can be invoked by the initiate or sendMessage methods.
CONTINUE_ON_ ERROR	Boolean	Specifies what happens to a process if an unexpected fault is raised and a fault handler is not defined for that fault. Possible values are: **True** Standard fault handling is applied. **False** Navigation of the process stops so that the process can be repaired.

PROCESS_TEMPL_ATTR view

Use this predefined Business Process Choreographer database view for queries on custom properties for process templates.

Table 154. Columns in the PROCESS_TEMPL_ATTR view

Column name	Type	Comments
PTID	ID	The ID of the process template that has a custom property.
NAME	String	The name of the custom property.
VALUE	String	The value of the custom property.

QUERY_PROPERTY view

Use this predefined Business Process Choreographer database view for queries on process-level variables.

Table 155. 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. If the property type is Boolean, the value is mapped to 0 (for false), or 1 (for true).
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 date, time, or timestamp, the value is mapped to a timestamp type, and this column contains the value of the timestamp.

TASK_AUDIT_LOG view

Use this predefined Business Process Choreographer database view for queries on audit log information for human tasks.

Inline tasks are logged in the AUDIT_LOG_B view. All other task types are logged in the TASK_AUDIT_LOG view. Depending on the choices made in WebSphere Integration Developer, inline human tasks can also trigger Common Event Infrastructure (CEI) events to be emitted for Business Flow Manager staff activity events and Human Task Manager task events.

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 TASK_AUDIT_LOG view. It lists the names of the columns, the event types, and gives a short description for the column.

Table 156. Structure of the TASK_AUDIT_LOG 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.
CONTAINMENT_ CTX_ID	x	x		The identifier of the containing context, for example, ACOID, PTID, or PIID.
DESCRIPTION	x		x	Resolved description string, where placeholders in the description are replaced by their current values. All affected languages are logged together in this column, formatted as an XML document. Only languages with descriptions containing placeholders for create-like events, or that have been explicitly updated for update-like events, are logged.
ESIID			x	The identifier of the escalation instance that is related to the current event.
ESTID			x	The identifier of the escalation template that is related to the current event.
EVENT_TIME	x	x	x	The time when the event occurred in Coordinated Universal Time (UTC) format.
FAULT_NAME	x			The name of the fault message. This attribute is applicable to the following events: TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FAULT_NAME_SPACE	x			The namespace of the fault message type. This attribute is applicable to the following events: TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FAULT_TYPE_NAME	x			The local name of the fault message type. This attribute is applicable to the following events: TASK_FAILED TASK_FAULT_MESSAGE_UPDATED

Table 156. Structure of the TASK_AUDIT_LOG view (continued)

Name	TIE	TTE	EIE	Description
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: KIND_HUMAN 101 KIND_ORIGINATING 103 KIND_PARTICIPATING 105 KIND_ADMINISTRATIVE 106

Table 156. Structure of the TASK_AUDIT_LOG view (continued)

Name	TIE	TTE	EIE	Description
TASK_STATE	x			The state of the task or task template. Possible values for task templates are: STATE_STARTED 1 STATE_STOPPED 2 Possible values for task instances are: STATE_INACTIVE 1 STATE_READY 2 STATE_RUNNING 3 STATE_FINISHED 5 STATE_FAILED 6 STATE_TERMINATED 7 STATE_CLAIMED 8 STATE_EXPIRED 12 FORWARDED 101
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: POTENTIAL_OWNER 1 EDITOR 2 READER 3 OWNER 4 POTENTIAL_STARTER 5 STARTER 6 ADMINISTRATOR 7 POTENTIAL_SENDER 8 ORIGINATOR 9 ESCALATION_RECEIVER 10 POTENTIAL_INSTANCE_CREATOR 11 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.

TASK view

Use this predefined Business Process Choreographer database view for queries on task objects.

Table 157. 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.
ASSIGNMENT_TYPE	Integer	Specifies how the work for the task is assigned. Possible values are: **ASSIGNMENT_TYPE_SINGLE** The task is assigned to one person only. **ASSIGNMENT_TYPE_PARALLEL** The task is assigned to several people who work on it simultaneously.
BUSINESS_RELEVANCE	Boolean	Specifies whether the task 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.
COMPLETED	Timestamp	The time when the task completed.
CONTAINMENT_CTX_ID	ID	The containment context for this task. This attribute determines the life cycle of the task. When the containment context of a task is deleted, the task is also deleted.
CTX_AUTHORIZATION	Integer	Allows the task owner to access the task context. Possible values are: **AUTH_NONE** No authorization rights for the associated context object. **AUTH_READER** Operations on the associated context object require reader authority, for example, reading the properties of a process instance.
DUE	Timestamp	The time when the task is due.
EXPIRES	Timestamp	The date when the task expires.
FIRST_ACTIVATED	Timestamp	The time when the task was activated for the first time.
FOLLOW_ON_TKIID	ID	The ID of the instance of the follow-on task.

Table 157. Columns in the TASK view (continued)

Column name	Type	Comments
HIERARCHY_ POSITION	Integer	Possible values are: **HIERARCHY_POSITION_TOP_TASK (0)** The top-level task in the task hierarchy. **HIERARCHY_POSITION_SUB_TASK (1)** The task is a subtask in the task hierarchy. **HIERARCHY_POSITION_FOLLOW_ON_TASK (2)** The task is a follow-on task in the task hierarchy.
INHERITED_AUTH	Integer	The kind of authorization that is inherited by a subtask from the parent task. Possible values are: **INHERITED_AUTH_NONE (0)** The authorization roles for the parent tasks are not inherited by the subtask. **INHERITED_AUTH_ADMINISTRATOR (1)** The subtask inherits the administrators of the parent tasks. **INHERITED_AUTH_ALL (3)** The subtask inherits all of the authorization roles that are defined for the parent tasks.
IS_AD_HOC	Boolean	Indicates whether this task was created dynamically at runtime or from a task template.
IS_CHILD	Boolean	Indicates whether this task is a child of a business process.
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_READ	Boolean	Indicates that someone has read the task.
IS_TRANSFERRED_ TO_WORK_BASKET	Boolean	Indicates that the task was transferred to the named work basket given in WORK_BASKET.
INVOKED_ INSTANCE_ID	ID	The instance ID of the invoked service; usually a task or process or activity.

Table 157. Columns in the TASK view (continued)

Column name	Type	Comments
INVOKED_ INSTANCE_TYPE	Integer	Possible values are: **INVOKED_INSTANCE_TYPE_NOT_SET (0)** States that the task did not invoke a service. **INVOKED_INSTANCE_TYPE_PROCESS (1)** States that the task invoked a business process. **INVOKED_INSTANCE_TYPE_ACTIVITY (2)** States that the task invoked an activity in a business process. **INVOKED_INSTANCE_TYPE_TASK (3)** States that task invoked a stand-alone task. **INVOKED_INSTANCE_TYPE_EVENT (4)** States that the task invoked an activity in an event handler or an activity. The activity is not yet ready to receive the event.
IS_WAIT_FOR_ SUB_TK	Boolean	Indicates whether the parent task is waiting for a subtask to reach an end state.
KIND	Integer	The kind of task. Possible values are: **KIND_HUMAN (101)** States that the task is a *collaboration task* that is created and processed by a human. **KIND_ORIGINATING (103)** States that the task is an *invocation task* that supports person-to-computer interactions, which enables people to create, initiate, and start services. **KIND_PARTICIPATING (105)** States that the task is a *to-do task* that supports computer-to-person interactions, which enable a person to implement a service. **KIND_ADMINISTRATIVE (106)** 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.

Table 157. Columns in the TASK view (continued)

Column name	Type	Comments
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.
STATE	Integer	The state of the task. Possible values are: **STATE_READY (2)** States that the task is ready to be claimed. **STATE_RUNNING (3)** States that the task is started and running. **STATE_FINISHED (5)** States that the task finished successfully. **STATE_FAILED (6)** States that the task did not finish successfully. **STATE_TERMINATED (7)** States that the task has been terminated because of an external or internal request. **STATE_CLAIMED (8)** States that the task is claimed. **STATE_EXPIRED (12)** States that the task ended because it exceeded its specified duration. **STATE_FORWARDED (101)** 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.
WORK_BASKET	String	The name of the work basket that this task belongs to.

TASK_CPROP view

Use this predefined Business Process Choreographer database view to query custom properties for task objects.

Table 158. Columns in the TASK_CPROP view

Column name	Type	Comments
TKIID	ID	The task instance 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_DESC view

Use this predefined Business Process Choreographer database view to query multilingual descriptive data for task objects.

Table 159. Column in the TASK_DESC view

Column name	Type	Comments
TKIID	ID	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_HISTORY view

Use this predefined Business Process Choreographer database view for queries on the event log for a task.

Table 160. Columns in the TASK_HISTORY view

Column name	Type	Comments
EVENT	Integer	The event type.
TKIID	ID	The ID of the task instance.
EVENT_TIME	Timestamp	The time when the logged event occurred.
PRINCIPAL	String	The name of principal who triggered the event.
FROM_ID	String	The name of user whose work item was transferred to the TO_ID, or whose work item was deleted. Not all events have a FROM_ID value.
TO_ID	String	The name of user for whom a work item was created, or transferred to. Not all events have a TO_ID value.
WORK_ITEM_KIND	Integer	The authorization type. Possible values are: WORK_ITEM_KIND_EVERYBODY (1) WORK_ITEM_KIND_USER (2) WORK_ITEM_KIND_GROUP (3)

Table 160. Columns in the TASK_HISTORY view (continued)

Column name	Type	Comments
REASON	Integer	The reason for the assignment of the work item. Not all events have a REASON value. Possible values are: REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)
PARENT_TKIID	ID	The ID of the related task instance. If the task instance is a subtask or a follow-on task of a parent task, this column contains the ID of the parent task. Otherwise, this column is null.
ESIID	ID	If the event is associated with an escalation instance, this column contains the escalation instance ID. Otherwise, this column is null.

NEXT_TIME	Timestamp	The time when the event is due to occur. The time depends on the event type.
		
		TASK_STARTED
		     The expiration time of the task instance.
		
		TASK_SUSPENDED
		     The resumption time of the task instance.
		
		TASK_COMPLETED, TASK_TERMINATED, TASK_EXIRED, and TASK_FAILED
		     The time when the task instance is deleted automatically.
		
		ESCALATION_STARTED
		     The time when the escalation fires.
		
		ESCALATION_FIRED
		     The time when the escalation fires again.

TASK_TEMPL view

This predefined Business Process Choreographer database view holds data that you can use to instantiate tasks.

Table 161. Columns in the TASK_TEMPL view

Column name	Type	Comments
TKTID	ID	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.

Table 161. Columns in the TASK_TEMPL view (continued)

Column name	Type	Comments
APPLIC_NAME	String	The name of the enterprise application to which the task template belongs.

ASSIGNMENT_TYPE	Integer	Specifies how the work for the task is assigned. Possible values are:
		
		ASSIGNMENT_TYPE_SINGLE
		    The task is assigned to one person only.
		
		ASSIGNMENT_TYPE_PARALLEL
		    The task is assigned to several people who work on it simultaneously.

AUTONOMY	Integer	Specifies the relationship of a task instance to the parent process. Possible values are:
		
		AUTONOMY_PEER (1)
		    The task instance runs independently of its parent process.
		
		AUTONOMY_CHILD (2)
		    The execution of the task instance depends on the parent process.
		
		AUTONOMY_NOT_APPLICABLE (3)
		    The task instance is an inline task and therefore the autonomy attribute is not applicable.

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.

CTX_ AUTHORIZATION	Integer	Allows the task owner to access the task context. Possible values are:
		
		AUTH_NONE
		    No authorization rights for the associated context object.
		
		AUTH_READER
		    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.

Table 161. Columns in the TASK_TEMPL view (continued)

Column name	Type	Comments

INHERITED_AUTH	Integer	The kind of authorization that is inherited by a subtask from the parent task. Possible values are:
		
		INHERITED_AUTH_NONE (0)
		    The authorization roles for the parent tasks are not inherited by the subtask.
		
		INHERITED_AUTH_ADMINISTRATOR (1)
		    The subtask inherits the administrators of the parent tasks.
		
		INHERITED_AUTH_ALL (3)
		    The subtask inherits all of the authorization roles that are defined for the parent tasks.

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:
		
		KIND_HUMAN (101)
		    States that the task is a *collaboration task* that is created and processed by a human.
		
		KIND_ORIGINATING (103)
		    States that the task is an *invocation task* that supports person-to-computer interactions, which enables people to create, initiate, and start services.
		
		KIND_PARTICIPATING (105)
		    States that the task is a *to-do task* that supports computer-to-person interactions, which enable a person to implement a service.
		
		KIND_ADMINISTRATIVE (106)
		    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:
		
		STATE_STARTED (1)
		    Specifies that the task template is available for creating task instances.
		
		STATE_STOPPED (2)
		    Specifies that the task template is stopped. Task instances cannot be created from the task template in this state.

Table 161. 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.
WORK_BASKET	String	The name of the work basket that will contain instances of this template.

TASK_TEMPL_CPROP view

Use this predefined Business Process Choreographer database view to query custom properties for task templates.

Table 162. Columns in the TASK_TEMPL_CPROP view

Column name	Type	Comments
TKTID	ID	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 Business Process Choreographer database view to query multilingual descriptive data for task template objects.

Table 163. Columns in the TASK_TEMPL_DESC view

Column name	Type	Comments
TKTID	ID	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 Business Process Choreographer database view for queries on work items and authorization data for process, tasks, and escalations.

Table 164. Columns in the WORK_ITEM view

Column name	Type	Comments
WIID	ID	The work item ID.
OWNER_ID	String	The principal ID of the owner.
GROUP_NAME	String	The name of the associated group worklist.
EVERYBODY	Boolean	Specifies whether everybody owns this work item.

OBJECT_TYPE	Integer	The type of the associated object. Possible values are:
		
		OBJECT_TYPE_ACTIVITY (1)
		    Specifies that the work item was created for an activity.
		
		OBJECT_TYPE_PROCESS_TEMPLATE (2)
		    Specifies that the work item was created for a process template.
		
		OBJECT_TYPE_PROCESS_INSTANCE (3)
		    Specifies that the work item was created for a process instance.
		
		OBJECT_TYPE_TASK_INSTANCE (5)
		    Specifies that the work item was created for a task.
		
		OBJECT_TYPE_TASK_TEMPLATE (6)
		    Specifies that the work item was created for a task template.
		
		OBJECT_TYPE_ESCALATION_ INSTANCE (7)
		    Specifies that the work item was created for an escalation instance.
		
		OBJECT_TYPE_ESCALATION_ TEMPLATE (8)
		    Specifies that the work item was created for an escalation template.
		
		OBJECT_TYPE_APPLICATION_ COMPONENT (9)
		    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.

Table 164. Columns in the WORK_ITEM view (continued)

Column name	Type	Comments
ASSOC_OID	ID	The ID of the object associated object with the work item. For example, the process instance ID (PIID) of the process instance containing the activity instance for which this work item was created.
REASON	Integer	The reason for the assignment of the work item. This integer value indicates one of the following meanings: REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)
CREATION_TIME	Timestamp	The date and time when the work item was created.

IBM ®

Printed in USA