IBM WebSphere Application Server Enterprise,
Version 5

Process Choreographer

Contents

Using Process Choreographer

Process Choreographer provides the IBM® WebSphere® Application Server the ability to choreograph all kinds of business processes. These processes typically require both human and IT resources. The types of processes can vary greatly, ranging from Web services or Web page navigation to business transaction support. Processes can be automatic recoverable processes or processes that require human interaction.

With Process Choreographer you can combine business-process technology with the other services that are available in WebSphere, that is, services offered by the J2EE architecture. Process Choreographer can be used to script EJBs, allowing the manipulation of processes as EJBs. It allows you to create Web services consisting of processes of other Web services.

The Process Choreographer framework provides a meta-model based on the IBM Flow Composition Model (FCM), which is part of Studio Application Developer. For a detailed description of the Process Choreographer architecture, refer to the *Process Choreographer Concepts and Architecture* white paper in the WebSphere Application Server Library at http://www.ibm.com/software/webservers/appserv/library.

The Process Choreographer Web client provides a Web browser interface for work items involving people. For a high-level view of how Process Choreographer can be used to integrate your business processes, see the Process Choreographer overview.

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

How to use, manage, and develop business-process applications is described in the following topics:

- "Planning to use Process Choreographer" on page 8
- "Configuring the business process container" on page 9
- "Deleting the business process container" on page 29
- "Managing processes" on page 29
- "Using the Process Choreographer Web client" on page 32
- "Developing applications using the Process Choreographer API" on page 51
- "Configuring the staff service for Process Choreographer" on page 76
- "Using compensation in service choreography" on page 83
- "Troubleshooting the business process container" on page 28

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

Process Choreographer overview

Process Choreographer is a powerful tool for executing complex business processes. In the paragraphs that follow, certain terms specific to Process Choreographer appear in **bold type** and are explained in context.

Each **process** is designed as a series of **activities**, and these activities are assigned to various people in your organization. Activities can be almost any business task: filling out a form, approving a document or drawing, writing a letter, and so on. The process is the design for how a series of tasks is to be done. When a process is started (by someone in your organization or even anonymously by someone filling out a form at a Web site), certain information is recorded and passed on to the first activity. The activity then shows up in the worklists of the potential **activity owners** - all the people who are authorized to work on it. In a typical implementation of Process Choreographer, everyone can look at the contents of the activity. They may also be able to see information about the flow, such as who started it, when it is due, and so on.

When you **claim** an activity, you become the **owner** of that activity. Only you and the **business process administrator** can work on that particular activity in that particular instance of the process. You can save intermediate stages of an activity if it is complex or elaborate. When you have finished the activity, you **complete** the activity. The resulting information in this activity is then routed to a new activity with a new set of potential activity owners. The process continues in this manner until all activities have been completed.

The activities and information pages in Process Choreographer appear as Web pages. If you know how to use a Web browser, you already know a lot about how Process Choreographer works.

Process Choreographer benefits

The benefits of Process Choreographer are many and diverse, depending on your organization’s goals and the kinds of things it does. The following are just some examples.

Faster work completion

If your organization’s processes are mostly paper-based, Process Choreographer can dramatically reduce the time it takes to accomplish tasks. Automating processes and moving documents electronically virtually eliminates the time required to move paper around your organization. In addition, the instructions and routing rules contained in the process definitions assure that problems are caught early and that repetitive tasks are automated.

Gains in productivity

Process Choreographer saves valuable time by helping you organize your work and set priorities. Work arrives with all the supporting documents. Your work may be listed by priority or due date, so you can work on the most important or most time-sensitive tasks first.

Improvements in process control

When procedures and rules live only inside the heads of workers, consistency and quality can suffer. Process Choreographer can be a powerful knowledge management tool for standardizing your organization's processes while providing great flexibility for modifying and improving them.

Improvements in customer service

When your organization's processes are standardized and work is completed faster, your customers - internal and external - benefit. Processes can even be designed that allow customers to initiate work and track its progress. Process Choreographer is a cost-effective way to increase customer satisfaction. Your organization benefits from repeat business and customer retention.

More effective collaboration

Over time, the process definitions created under Process Choreographer can become a storehouse of "best practices" for the organization, which can be shared across the enterprise.

Process Choreographer glossary

This glossary defines terms and abbreviations used by Process Choreographer.

Activity
An activity is one of the nodes that make up a process. An activity can be one of the following types:

- Process activity
- Elemental activity
- Empty activity
- Person activity
- Event activity

Activity instance
An activity instance is an instance of an activity template. It is created as part of a process instance during the execution of a process. At any given time, there can be more than one instance of an activity template.

Activity template
An activity template contains the specifications for an activity.

Activity types
An activity type describes alternative forms of implementations of an activity template. There are the following activity types:

- Process activity
- Elemental activity
- Empty activity
- Person activity
- Event activity

Adapter
A software system that invokes or is invoked by programming interfaces in other software systems and presents a standard interface, for example, one based on XML messages.

Audit trail
The audit trail is a log that contains an entry for predefined events that

occur during the execution of a process instance. Audit logs are written by plug-ins. Process Choreographer provides a default plug-in that writes the log entries into a relational database.

Business process container
The business process container is the part of the Web application server that is responsible for the execution of business processes. Business processes that are installed on a Web application server are executed in their corresponding business process containers.

Block A process that is defined within its parent business process. Also known as an *inline subprocess*.

Claim If a user takes over the responsibility for working on an activity, this is referred to as claiming. If an activity is claimed, it is reserved for exclusive use.

Complete
When an activity is completed, the results of the activity are returned to the business process container for further processing. You can only complete activities that you have claimed.

Control link
Defines the possible flow of control between two nodes in a process. The actual flow of control is determined at run time based on the value of the transition conditions associated with the control link.

Custom attribute
An attribute that can be added to a process or activity by process modelers according to their needs.

Dead-path elimination
To guarantee the completion of a process, the business process container removes run-time paths that can no longer be reached during the execution of a process instance.

Deployment
To make a business process executable, it must be deployed. During the deployment of a process, the run-time artifacts required to run the process are generated, for example, Java™ classes specified in the FDML. The process is translated to a deployed form, which can be stored in a database. The result of deploying a process is a deployed process module (FAR file) that can be installed in a business process container.

See also *6*.

EAR file
Enterprise application archive.

Early binding
The binding of activity implementations, such as services or processes, to elemental or process activities at process modeling or deployment time.

Editor (authorization level)
A list of users that provides edit access to the messages associated with an activity.

EJB Enterprise Java bean.

Elemental activity
An elemental activity is an activity in a process that is implemented by a service.

Empty activity
An empty activity is an activity without an implementation, for example:
- Explicit synchronization points for parallel branches
- Explicit split or join nodes
- Explicit branch nodes

Event An external, asynchronous notification that is sent to a process. For example, an e-mail from a customer, or an EDI message.

Event activity
An event activity is an activity in a process that waits for an event.

Facade EJB
An Enterprise Java bean that provides a synchronous interface to a specific process with direct and type-safe access.

Facade MDB
A message-driven bean that provides an asynchronous interface to a specific process using JMS messages.

FAR file
Process archive.

Fault (also known as exception)
Faults indicate exception conditions. If a fault occurs, a fault message is sent to the corresponding fault terminal.

Fault message
The message that is associated with a fault.

Fault terminal
Activities and processes can have fault terminals. Fault terminals are used to model the error case. A fault terminal becomes active if the activity or process detect the corresponding error.

FDML Flow Definition Markup Language.

Input message
The input message contains all IN-parameters of an activity or a process.

Interruptible process
A long-running process that includes several transactions for execution. Individual activities can be executed in parallel, possibly even on different servers. A running instance of an interruptible process is associated with a persistent state in the process database so that the process can be recovered at any time. An interruptible process can contain all kinds of activities, including asynchronous invocations and activities that require human interaction.

JCA J2EE Connector Architecture.

JMS Java Message Service.

Late binding
The binding of activity implementations, such as processes, to activities, such as process activities, at run time.

MDB Message-driven bean.

Message
A data instance that is manipulated by a process is called a message. This includes data, which is:
- Passed when the process is started

- Passed to the input of an activity
- Produced by an activity
- Stored in a variable
- The result of the process
- The result of exceptions

Non-interruptible process
A short-running, non-interruptible process that is run as part of a single call or invocation to or from the process engine. A running instance of a non-interruptible process is associated with a thread of the Web application server executing on behalf of the non-interruptible process for its entire duration. A non-interruptible process can contain only activities that are implemented by synchronous operations.

Output message
An output message contains all OUT-parameters of an activity or a process.

Owner of an activity
The person who has claimed an activity and is therefore responsible for its completion.

Owner of a work item
The person who has received a work item. Ownership of a work item implies the responsibility to complete the associated activity.

Parent process
Is a process that contains an activity that is implemented by another process.

Person activity
An activity in a process that represents a human interaction. A person activity has associated staff queries that specify the set of eligible users to perform it, and additional properties to specify its behavior, such as GUI parameters and required user inputs.

Potential owner
A person who is eligible to claim the activity.

Process
A process consists of activities, where each activity represents a task, such as the invocation of a service, another process, or a human interaction. The sequence of execution of the activities is defined by using control links. Processes allow you to compose a service out of other services.

Process activity
An activity that is implemented by a process. The activity becomes a subprocess of the current process.

Process administrator (authorization level)
A list of users that provides administrative access to a business process and its activities. The process administration access right includes all other authorizations.

Process instance
A process instance is an instance of a process template. Multiple process instances of the same process template can be executed in parallel.

Process model
Is a synonym for business process and contains all definitions of a process.

Process module
A process module is an archive file that contains one or more processes, that is, FDML files that have been generated by the WebSphere Studio Tools. Process modules are deployed and installed as part of EAR files. When a process module is deployed, it can contain additional run-time artifacts, such as generated Java classes that are needed for the execution of the contained processes. Process modules are manageable entities that appear in the WebSphere Application Server Administrative Console.

Also known as a *FAR file*.

Process navigation
The process of execution of processes by the business process container.

Process template
A process template is a deployed and installed process that is used as a template for process instances at run time. It contains all information of a process model. The process templates can be managed in the WebSphere Application Server Administrative Console.

Reader (authorization level)
A list of users that provides read-only access to an activity or a process.

Services
Operations that are made available by providing a service description (WSDL). Services are used to implement elemental activities of a process. There are Web services and local services. An example for a Web service is a SOAP service. Examples for local services are:

- Java methods
- EJB methods
- CICS® transactions
- SAP remote function calls

Subprocess
A subprocess is a process that implements a process activity in a parent process.

Top-level process
A parent process at the highest level.

Transition condition
Determines the truth value of the associated control connector at run time, determining which paths of the process are taken at run time, and which paths are eliminated. The start condition of an activity refers to the truth values of the control links.

Variable
The schema of a message is described as a variable. The schema is described in the form of a WSDL message.

WAR file
Web application archive.

Web service
A Web service is a software application identified by a URL, whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts. It supports direct interactions with other software applications using XML-based messages via Internet-based protocols.

Work item
Representation of work to be done in the context of an activity in a process instance.

Worklist
A worklist contains a set of work items obtained by a query using certain filter and sort criteria.

WSDL
Web Services Description language.

WSIF Web Services Invocation Framework.

Planning to use Process Choreographer

For each application server where you want to use Process Choreographer, you must configure the business process container before installing any enterprise applications that contain processes. To prepare for configuring the business process container, plan the following:

Steps for this task

1. To gain more understanding about the Process Choreographer architecture, refer to the white paper in the WebSphere Application Server Library at http://www.ibm.com/software/webservers/appserv/library.
2. If you intend to use Process Choreographer in a network deployment (ND) environment, refer to the white paper in the WebSphere Application Server Library at http://www.ibm.com/software/webservers/appserv/library.
3. Decide whether you want to use the JMS messaging service provided by:
   - The messaging service that is embedded in WebSphere.
   - A separate MQ (MQSeries® or WebSphere MQ) installation.

   Note: If you intend to use embedded messaging, you must have selected this option when you installed WebSphere. If you want to use MQ messaging, it must be installed before you start configuring the business process container.
4. Which database system will you use?

   One of:
   - DB2® UDB
   - Cloudscape
   - Oracle - using an OCI driver and the Oracle 32 bit library.
   - Sybase ASE - since Process Choreographer uses distributed transactions, you will need to purchase and install the DTM feature for Sybase ASE.

   Check the required software levels at http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html.
5. Which machine will host the database?

   If the database machine is remote, you need a suitable database client.
6. Default or production database configuration?

   For non-production environments, there are quick database creation scripts for DB2 and Cloudscape which create a default database suitable for testing, development, and demos.
7. Name of the database.

   The business process container will use this as the data source. Clusters running the same process must use the same database. The quick database creation scripts for DB2 and Cloudscape create a database named `bpedb`.

8. User ID of the database administrator.

   This user ID will be used to create the database. The same user ID must also be used to access the database at runtime. Depending on your security requirements, you can create a user ID just for creating and accessing the database. This user ID will own the schema.
9. Plan the business process container settings.

Results

You are ready to configure the business process container.

Configuring the business process container

Before you begin

You must have completed planning to use Process Choreographer.

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

Steps for this task

1. Make sure that your IBM WebSphere Application Server is running.
2. Decide whether you want to configure the business process container using the Install Wizard or manually.

   To use the Install Wizard, continue with step 3. To configure the business process manually, continue with step 4.
3. To configure the business process container **using the Install Wizard**, perform the following:
   a. In the Administrative console, click **Servers > Application Servers > *YourServer***.
   b. In the **Additional Properties** section, click **Business process container**.
   c. Ignore the Business process container settings, near the bottom of the page, click on the link for the **Install Wizard**, and follow the step-by-step instructions provided.
   d. Create the database and tables using the scripts provided in the directory `%WAS_HOME%\ProcessChoreographer`.
   e. If you are using WebSphere MQ (as opposed to embedded messaging), create the queue manager and the queues using the scripts provided in the directory `%WAS_HOME%\ProcessChoreographer`.
4. To configure the business process container **manually**, perform the following:
   a. Creating and configuring the database for the business process container.
   b. Creating and configuring the queues for the business process container.
   c. Creating and configuring the scheduler.
   d. Installing the business process container.
5. Activating the business process container.
6. Verifying that the business process container works.

   In case of problems, see Troubleshooting the business process container.

Results

The business process container is configured and working.

What to do next

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

Creating and configuring the database for the business process container

Before you begin

Your database system must be installed and available.

The business process container requires a database, JDBC provider, and data source.

Steps for this task

1. If your database server is not on the same machine as your Enterprise Application Server: Copy the ddl scripts for your database system to your database server machine.

   On Windows®, copy the ddl files from `%WAS_HOME%\ProcessChoreographer`. On UNIX®, copy them from `$WAS_HOME/ProcessChoreographer`.
2. On the machine hosting the database server, create the database according to the description for your database system.
   - Creating a DB2 database for the business process container.
   - Creating an Oracle database for the business process container.
   - Creating a Cloudscape database for the business process container.
   - Creating a Sybase database for the business process container.
3. If the database is not on the same machine as your Enterprise Application Server, refer to the documentation provided with your database system to:
   a. Install a suitable database client on the Enterprise Application Server machine.
   b. Make the new database known to the database client.
   c. Verify that you can access the remote database using the client.
4. On your Enterprise Application Server machine, configure your JDBC provider and data source according to the description for your database system:
   - Configuring a DB2 JDBC provider and data source for the business process container.
   - Configuring an Oracle JDBC provider and data source for the business process container.
   - Configuring a Cloudscape JDBC provider and data source for the business process container.
   - Configuring a Sybase JDBC provider and data source for the business process container.

Results

The Process Choreographer database exists and the JDBC provider and data source have been defined.

What to do next

Create the queues, and configure the JMS resources and listeners.

Creating a DB2 database for Process Choreographer

Steps for this task

1. Change to the directory where the configuration scripts for Process Choreographer are located:
   - If your database server is on the **same** machine as your Enterprise Application Server: On Windows, enter: `cd %WAS_HOME%\ProcessChoreographer`. On UNIX, enter: `cd $WAS_HOME/ProcessChoreographer`.
   - If your database server is on a **different** machine than your Enterprise Application Server, change to the directory where you copied the ddl scripts.
2. If you want to use an existing database, skip to step 4.

   Note: Make sure that the database supports Unicode (UTF-8) . Without Unicode support, it cannot store all characters that can be handled in Java, and you may run into code page conversion problems when a client uses an incompatible code page.

   To avoid deadlocks, be sure that the DB2 isolation level is set to 'read stability'. If necessary, enter the command:

   ```
   db2set DB2_RR_TO_RS=YES
   ```

   then restart the DB2 instance to activate the change.
3. If you want to create a new database named `BPEDB`:
   a. Make sure that you are using a user ID that has administrator rights for the database system.

   b. In the DB2 command line processor, enter the command to run the quick database creation script:

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

   c. Make sure that the script's output contains no errors.

   In some cases, the CLI packages are not bound to the new database. To be sure that this happens, for a database named `BPEDB`:

   On Windows, enter:

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

   On UNIX, enter:

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

   A DB2 database named `BPEDB` has been created.
4. To create the tablespace and schema:
   a. **(Optional)** Analyze the results of your experiences during development and system testing.

   The size of your database depends on many factors. Non-interruptible processes require very little space. Each process template may require tens or hundreds of kilobytes. If possible, distribute tablespace containers across different logical disks, and implement an appropriate security policy. Consider the performance implications of your choices for buffer pools and log file settings.

   b. Edit the tablespace creation script `createTablespaceDb2.ddl` according to the instruction at the top of the file.

   c. Make sure that you have administrator rights for the database system.

The user ID you use to create the schema must be the one that you specify for WebSphere to use to access the database.

d. Make sure that you are attached to the correct instance.

   Check the environment variable `DB2INSTANCE`.

e. To connect to a database named `BPEDB`, in the DB2 command line processor, enter the command:

   ```
   db2 connect to BPEDB
   ```

f. To create the tablespace, enter the command:

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

   Make sure that the script's output contains no errors. If there were any errors, you can drop the tablespace using the script `dropTablespaceDb2.ddl`.

g. To create the schema (tables and views), in the DB2 command line processor, enter the command:

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

   Make sure that the script's output contains no errors. If there were any errors, you can use `clearSchemaDb2.ddl` to clear the schema, and `dropSchemaDb2.ddl` to drop the schema.

Results

The DB2 database for Process Choreographer exists.

Creating an Oracle database for Process Choreographer

There is no script to quickly create a default Oracle database for Process Choreographer.

Steps for this task

1. Be sure that you are using OCI drivers and the 32 bit Oracle library `lib32`.
2. On UNIX, create soft links to the following three Oracle libraries in the `/usr/lib` directory.
   - For Oracle 8i: Link to: `libwtc8.so`, `libclntsh.so.8.0`, and `libocijdbc8.so`.
   - For Oracle 9i: Link to: `libwtc9.so`, `libclntsh.so.9.0`, and `libocijdbc9.so`.
3. Create an Oracle database using the Database Configuration Assistant.

   Make sure that you select the JServer option for the database. It is recommended that you use a Unicode codepage when creating the database. The text data you pass to the APIs must be compatible with the selected codepage.
4. Change to the directory where the configuration scripts for Process Choreographer are located:
   - If your database server is on the **same** machine as your Enterprise Application Server: On Windows, enter: `cd %WAS_HOME%\ProcessChoreographer`. On UNIX, enter: `cd $WAS_HOME/ProcessChoreographer`.
   - If your database server is on a **different** machine than your Enterprise Application Server, change to the directory where you copied the ddl scripts.
5. Edit the tablespace creation script according to the instructions at the top of the file:
   - For Oracle 8i: Edit `createTablespaceOracle8.ddl`
   - For Oracle 9i: Edit `createTablespaceOracle9.ddl`

6. Make sure that you are using the user ID that has administrator rights for the database system.
7. If you do not want the schema to be created in the default instance, set the environment variable `ORACLE_SID`.
8. To create the tablespace, run the script `createTablespaceOracleX.ddl`, where 'X' is your Oracle version digit (8 or 9).

   For test purposes you can use the same location for all tablespaces and pass the path as command line argument to the script, for example, on Windows, using Oracle 8i, user ID `bpeuser`, password `bpepwd`, database name `BPEDB`, and tablespace path `d:\mydb\ts`, enter:

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

   If you get any errors creating the tablespace, you can use `dropTablespaceOracleX.ddl` to drop the tablespace, where 'X' is your Oracle version digit (8 or 9).
9. To create the schema, run the script `createSchemaOracleX.ddl`, where 'X' is your Oracle version digit (8 or 9).

   For example, on Windows using Oracle 9i, enter:

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

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

Results

The Oracle database for Process Choreographer exists.

Creating a Cloudscape database for Process Choreographer

Cloudscape is a database system implemented in Java. It comes with the WebSphere Application Server as three JAR files (db2j.jar, db2jtools.jar, and db2jcview.jar). The Cloudscape license that comes with WebSphere is only for development and test, not for production purposes.

To create a Cloudscape database named `BPEDB`:

Steps for this task

1. Make sure that you have administrator rights.
2. Either change to the directory where you want the new database to be created, or edit the script `createDatabaseCloudscape.ddl` located in the `ProcessChoreographer` subdirectory of the IBM WebSphere Application Server installation directory, and add the fully qualified path to the database name, for example, on Windows change the `connect` command line to:

   ```
   connect "d:\db\BPEDB;create=true" as BPEDB
   ```

   Note: Cloudscape only allows one local connection. If WebSphere Application Server is running and accessing a Cloudscape database, attempting to open a second connection to the database from the command line will be rejected.
3. At the command prompt, enter the command to run the database creation script using the Cloudscape command line processor:

   On Windows, enter:

   ```
   java -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
        %WAS_HOME%\ProcessChoreographer\createDatabaseCloudscape.ddl
   ```

On UNIX, enter:

```
java -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
     $WAS_HOME/ProcessChoreographer/createDatabaseCloudscape.ddl
```

Note: If the db2j commands cannot be found, add the Cloudscape JAR files `db2j.jar` and `db2jtools.jar` to the `classpath`, and try again. On Windows, these files are located in `%WAS_HOME%\AppServer\lib`. On UNIX they are located in `$WAS_HOME/AppServer/lib`.

Results

The Cloudscape database for Process Choreographer exists.

Creating a Sybase database for the Process Choreographer

Steps for this task

1. Make sure that you have administrator rights.
2. Make sure that you have the DTM option for Sybase ASE installed and configured:
   a. Set `enable DTM` to `1` in the Sybase server configuration.
   b. Set `enable xact coordination` to `1` in the Sybase server configuration.
   c. Add the role `dtm_tm_role` to the Sybase administration user ID, for example, user ID `sa`.
   d. Restart the Sybase server.
3. Change to the directory where the configuration scripts for Process Choreographer are located:
   - If your database server is on the **same** machine as your Enterprise Application Server: On Windows, enter: `cd %WAS_HOME%\ProcessChoreographer`. On UNIX, enter: `cd $WAS_HOME/ProcessChoreographer`.
   - If your database server is on a **different** machine than your Enterprise Application Server, change to the directory where you copied the ddl scripts.
4. If you want to create a non-production database for stand-alone development, evaluation, or demo purposes, you only need to enter the command:

   ```
   isql -S <serverName> -U <userid> -P <password> -i createDatabaseSybase120.ddl
   ```

   A Sybase database named `BPEDB` has been created.
5. If you prefer to create your database manually:
   a. Inspect the options used in the quick database creation script `createDatabaseSybase120.ddl`, and make sure that you include these in the database you create.
   b. Create the database.
   c. Run the script to create the schema, by entering the command:

   ```
   isql -S <serverName> -U <userid> -P <password> -D <databaseName> -i createSchemaSybase120.ddl
   ```

   Where

   serverName
   is the name of the Sybase server as defined in the `dsedit` tool.

   userid is the user ID to be used.

   password
   is the password for *userid*.

databaseName
is the name of the database.

Results

The Sybase database for Process Choreographer exists.

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

You can either provide the parameters to the script file on the command line, or you can do everything using the administrative console. Perform only one of the following steps:

Steps for this task

1. Enter all the customization parameters on the command line.

   On Windows, enter:

   ```
   cd %WAS_HOME%
   bin\wsadmin -f ProcessChoreographer\create_ds_Db2.jacl <server> <DBPath>
    <database> <userid> <password>
   ```

   Note: The second line of the previous command extended beyond the width of the page. Type the second and third lines as one continuous line.

   On UNIX, enter:

   ```
   cd $WAS_HOME
   bin/wsadmin.sh -f ProcessChoreographer/create_ds_Db2.jacl <server> <DBPath>
    <database> <userid> <password>
   ```

   Note: The second line of the previous command extended beyond the width of the page. Type the second and third lines as one continuous line.

   Where:

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

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

   server is the application server name.

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

   database
   is the database name.

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

password
is the password for *userid*.

2. Configure the JDBC provider and data source for the business process container using the administrative console:
   a. In the Administrative Console, click **Resources > JDBC Providers**.
   b. Click **New**.
   c. Enter your values for the following fields:

Field	Example value
Name	BPEJdbcDriverDB2
Description	JDBC Provider for Process Choreographer
Classpath	e:/sqllib/java12/db2java.zip
Implementation Class Name	COM.ibm.db2.jdbc.DB2XADataSource

   d. Click **Apply** and **Save**.
      The provider `BPEJdbcDriverDB2` is now listed in the JDBC Provider Panel.
   e. Select your new provider, `BPEJdbcDriverDB2`.
   f. In **Additional Properties**, select **Data Sources**.
   g. Click **New**.
   h. Enter your values for the following fields:

Field	Example value
Name	BPEDataSourceDb2
JNDI Name	jdbc/BPEDB
Description	DataSource for Process Choreographer
Category	Process Choreographer
Statement Cache Size	0
Datasource Helper Class Name	com.ibm.websphere. rsadapter.DB2DataStoreHelper

   i. Click **Apply** and **Save**.

   The data source `BPEDataSourceDb2` has been created and is now listed in the Data Source panel.

Results

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

What to do next

Create the queues, and configure the JMS resources and listeners.

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

Before you begin

To configure an Oracle JDBC provider and data source, you must be using an OCI driver.

Steps for this task

1. Enter all the configuration parameters on the command line.

   For example, on Windows, enter:

```
cd %WAS_HOME%
bin\wsadmin -f ProcessChoreographer\create_ds_Oracle.jacl
       <server> <DBPath> <tnsname> <userid> <password>
```

   On UNIX, enter:

```
cd cd $WAS_HOME
bin/wsadmin.sh -f ProcessChoreographer/create_ds_Oracle.jacl
          <server> <DBPath> <tnsname> <userid> <password>
```

   Where:

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

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

   server is the application server name.

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

   tnsname
   is the TNS name. It should match the name used for the Process Choreographer database in the file `tnsnames.ora`.

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

   password
   is the password for *userid*.

   For example, on Windows:

```
bin\wsadmin -f ProcessChoreographer\create_ds_Oracle.jacl server1
 "c:/oracle/ora90" BPEDB frank pwd123
```

   Note: The previous command extended beyond the width of the page. Type the command as one continuous line.

Results

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

What to do next

Create the queues, and configure the JMS resources and listeners.

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

Steps for this task

1. Enter all the configuration parameters on the command line.

   For example, on Windows, enter:

   ```
   cd %WAS_HOME%
   bin\wsadmin -f ProcessChoreographer\create_ds_Cloudscape.jacl <server> <database>
   ```

   On UNIX, enter:

   ```
   cd $WAS_HOME
   bin/wsadmin.sh -f ProcessChoreographer/create_ds_Cloudscape.jacl <server> <database>
   ```

   Where:

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

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

   server the application server name.

   database
   the path to the Process Choreographer database directory. Note, on Windows use forward slashes ('/') instead of back slashes ('\'). For example, "c:/mydbs/BPEDB".

   For example, on Windows:

   ```
   bin\wsadmin -f ProcessChoreographer\create_ds_Cloudscape.jacl server1 "c:/mydbs/BPEDB"
   ```

Results

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

What to do next

Create the queues, and configure the JMS resources and listeners.

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

Steps for this task

1. Enter all the configuration parameters on the command line.

   For example, on Windows, enter:

   ```
   cd %WAS_HOME%
   bin\wsadmin -f ProcessChoreographer\create_ds_Sybase.jacl
      <server> <DBPath> <dbServerName> <port> <database> <userid> <password>
   ```

   On UNIX enter:

   ```
   cd $WAS_HOME
   bin/wsadmin.sh -f ProcessChoreographer/create_ds_Sybase.jacl
      <server> <DBPath> <dbServerName> <port> <database> <userid> <password>
   ```

Where:

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

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

server is the application server name.

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

dbServerName
: is the name of the database server machine.

port is the port used on the database server.

database
: is the name of the Process Choreographer database.

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

password
: is the password for *userid*.

For example, on Windows:

```
%WAS_HOME%\bin\wsadmin -f %WAS_HOME%\ProcessChoreographer\create_ds_Sybase.jacl
         server1 "c:/sybase" mymachine 4100 BPED frank pwd123
```

Results

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

What to do next

Create the queues, and configure the JMS resources and listeners.

Creating and configuring the queues for the business process container

The business process container uses a JMS provider for sending and receiving messages. If you have installed the IBM WebSphere Application Server "embedded messaging" option, it is recommended to use it. Otherwise, you must have MQ installed.

Steps for this task

1. Create and configure the queues, JMS provider, and message listener service, by performing one of the following:

- **WebSphere JMS provider:** Creating and configuring queues using the JMS provider embedded in WebSphere.
- **MQ JMS provider:** Creating and configuring queues using an external MQ JMS provider.

Results

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

What to do next

Create and configure the process scheduler

Creating and configuring queues for the business process container using the JMS provider embedded in WebSphere

Each business process container needs message queues, a JMS provider, and a message listener service. This topic explains how to create these resources using WebSphere's embedded messaging. The process for using the embedded messaging is easier than using a stand-alone MQ installation.

Steps for this task

1. Make sure that you have the embedded messaging option installed on your WebSphere Application Server.
2. Make sure that your user ID has the authority to create queues.
3. Change to the IBM WebSphere Application Server installation directory.

   On Windows, enter:
   ```
   cd %WAS_HOME%
   ```
   On UNIX, enter:
   ```
   cd $WAS_HOME
   ```
4. To create and configure the queues and queue resources, run `config_queue_emb.jacl` using the `-f` option. This file is in the `ProcessChoreographer` subdirectory.

   On Windows, enter:
   ```
   bin\wsadmin -f ProcessChoreographer\config_queue_emb.jacl
     -userid <userid> -passwd <password> [-node <nodeName>] -server <serverName>
     [-internalQueue <internalQueueName>] [-apiQueue <apiQueueName>] [-holdQueue <holdQueueName>]
     [-retentionQueue <retentionQueueName>]
   ```
   On UNIX, enter:
   ```
   bin/wsadmin.sh -f ProcessChoreographer/config_queue_emb.jacl
     -userid <userid> -passwd <password> [-node <nodeName>] -server <serverName>
     [-internalQueue <internalQueueName>] [-apiQueue <apiQueueName>] [-holdQueue <holdQueueName>]
     [-retentionQueue <retentionQueueName>]
   ```
   The default values for the optional parameters are:

node	<local node>
internalQueueName	BPEIntQueue
apiQueueName	BPEApiQueue

holdQueueName
BPEHldQueue

retentionQueueName
BPERetQueue

5. **(Optional)** You can check or change the configuration settings using the Administrative Console.

Results

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

What to do next

Create and configure the scheduler service.

Creating and configuring queues for the business process container using an external MQ JMS provider

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

Steps for this task

1. Make sure that your user ID has the authority to create MQ queues.
2. Change to the IBM WebSphere Application Server installation directory.

   On Windows, enter:
   ```
   cd %WAS_HOME%
   ```
   On UNIX, enter:
   ```
   cd $WAS_HOME
   ```
3. To create the queues:

   On Windows, enter:
   ```
   ProcessChoreographer\CreateQueues.bat <QueueManager>
   ```
   On UNIX, enter:
   ```
   ProcessChoreographer/CreateQueues.sh <QueueManager>
   ```
   If the named queue manager does not exist, it will be created and started.
4. To configure the queues and queue resources, run `config_queue_mq.jacl` using the `-f` option. This file is in the `ProcessChoreographer` subdirectory.

   On Windows, enter:
   ```
   bin\wsadmin -f ProcessChoreographer\config_queue_mq.jacl
     -userid <userid> -passwd <password> [-node <nodeName>] -server <serverName>
     -mqPath <MQPath> [-qmName <queue manager name>] [-qmNamePut <cluster queue manager name>
     [-internalQueue <internalQueueName>] [-apiQueue <apiQueueName>] [-holdQueue <holdQueueName>]
     [-retentionQueue <retentionQueueName>]
   ```
   On UNIX, enter:
   ```
   bin/wsadmin.sh -f ProcessChoreographer/config_queue_mq.jacl
     -userid <userid> -passwd <password> [-node <nodeName>] -server <serverName>
     -mqPath <MQPath> [-qmName <queue manager name>] [-qmNamePut <cluster queue manager name>]
     [-internalQueue <internalQueueName>] [-apiQueue <apiQueueName>] [-holdQueue <holdQueueName>]
     [-retentionQueue <retentionQueueName>]
   ```

The default values for the optional parameters are:

node	<local node>
qmName	WAS_<nodeName>_<serverName>
qmNamePut	WAS_<nodeName>_<serverName>
internalQueueName	BPEIntQueue
apiQueueName	BPEApiQueue
holdQueueName	BPEHldQueue
retentionQueueName	BPERetQueue

5. If you have problems using the script `config_queue_mq.jacl`, you can also configure the resources manually using the Administrative Console.
6. **(Optional)** You can check or change the configuration settings using the Administrative Console.

Results

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

What to do next

Create and configure the scheduler service.

Configuring MQ resources for the business process container using the Administrative Console

Before you begin

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

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

Steps for this task

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

Field	Example value
Name	BPECF
JNDI Name	jms/BPECF
User ID	*userID*
Password	*password*
Queue Manager	WAS_QueueManager

Leave the other fields blank or accept the default values.

5. Click **Apply**, and click **Save**.

   The MQ Queue Connection factory `BPECF` has been created and is listed on the MQ queue connection factory panel.
6. Select **Resources > JMS > WebSphere MQ JMS Providers**.
7. To define the MQ Queue Destination for the external message queue `BPEApiQueue`:
   a. Click **WebSphere MQ Queue Destination**.
   b. Click **New**.
   c. Enter the following values:

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

   Leave the other fields blank or accept the default values.

   d. Click **Apply**, and click **Save**.
8. Repeat step 7 for each of the following queues, substituting their names where appropriate:
   a. The hold queue: `BPEHoldQueue`.
   b. The internal messages queue: `BPEInternalQueue`.
   c. The retention queue: `BPERetentionQueue`.

   The four queue destinations have been defined and are listed on the queue panel.
9. To create the listener port for external messages, `BPEApiListenerPort`:
   a. Click **Servers > Applications >** ***YourServer*** **> Message Listener Service > Listener Ports**.
   b. Click **New**.
   c. Enter the following values:

Field	Example value
Name	BPEApiListenerPort
Description	API Listener Port for Process Choreographer
Connection Factory JNDI Name	jms/BPECF
Destination JNDI Name	jms/BPEApiQueue
Max Sessions	5
Max Retries	10
Max Messages	1

   d. Click **Apply**, and click **Save**.
10. Repeat step 9 for each of the following listener ports, substituting their names where appropriate:
    a. The listener port for the hold queue: `BPEHoldListenerPort`.
    b. The listener port for internal messages: `BPEInternalListenerPort`.

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

Results

The JMS queue connection factory, queue destinations, and listener ports are configured.

What to do next

Create and configure the scheduler service.

Creating and configuring the scheduler service for Process Choreographer

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

Steps for this task

1. Change to the IBM WebSphere Application Server installation directory:

   On Windows, enter:
   ```
   cd %WAS_HOME%
   ```
   On UNIX, enter:
   ```
   cd $WAS_HOME
   ```
2. Run the createScheduler.jacl script using the -f option:

   On Windows, enter:
   ```
   bin\wsadmin -f ProcessChoreographer\createScheduler.jacl -server
   <Server> -node <Node>
   ```
   Note: The previous line is one contiuous line, but does not fit within the width of the page.

   On UNIX, enter:
   ```
   bin/wsadmin.sh -f ProcessChoreographer/createScheduler.jacl -server
   <Server> -node <Node>
   ```
   Note: The previous line is one contiuous line, but does not fit within the width of the page.

   Where:

   Server is the name of the application server.

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

   If you have problems using the createScheduler.jacl, you can use the Administrative console to configure the resources manually.

   For more information about the scheduler service, see ″Using the scheduler service″, which is located in the InfoCenter.
3. **(Optional)** You can check or change the configuration settings using the Administrative Console.

Results

The scheduler has been created and configured for Process Choreographer.

What to do next

Enter the business process container settings.

Installing the business process container

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

Steps for this task

1. Click **Servers > Application Servers >** *YourServer*.
2. In the **Additional Properties** section, click **Business process container**.
3. On the configuration page, if necessary replace the default values with your values.
4. Click **Install** (even if you didn't change any of the default values).

   This causes the business process container's EAR file to be installed and configured. It is OK if you get an error message telling you that the container is already configured. However, if you want to change any of the container's configuration settings, you must uninstall the `ProcessContainer.ear` application and then repeat this task from step 1.
5. Click **Save**.

Results

The business process container has been installed and configured.

What to do next

You must activate the business process container.

Process Container settings

Use this page to manage process containers.

A Process Choreographer process container provides services to execute business processes within an application server.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Process Container** .

Note: These fields cannot be changed after they have been applied. If you want to change any of these values after the container has been configured, you must uninstall and reinstall the enterprise application file `BPEContainer.ear`. This can result in the loss of data such as pending messages in queues, process templates, and process instances in the Process Choreographer database.

Datasource

The JNDI name of the data source, which contains process data.

Data type
String

Default
jdbc/BPEDB

Security Role For Process Administrator

The security role that the process container uses to run a process.

Data type
String

Default
ProcessAdministrator

Listener Port For Internal Messages

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

Data type
String

Default
BPEInternalListenerPort

Listener Port For External Requests

This listener port provides the settings for the message-driven bean that handles requests from Process Choreographer API clients.

Data type
String

Default
BPEApiListenerPort

Listener Port For Unprocessed Messages

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

Data type
String

Default
BPEHoldListenerPort

Retry Limit

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

Data type
Integer

Default
5

Range 2 to 10 (recommended)

Retention Queue

Queue that contains those messages that temporarily could not be processed.

Data type
String

Default
jms/BPERetentionQueue

Retention Queue Factory

Factory for the retention queue.

Data type
String

Default
jms/BPECF

Retention Queue Message Limit

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

Data type
Integer

Default
20

JMS API User ID

The user ID that the Process Choreographer MDB will use when processing asynchronous API calls. This is only required when WebSphere security is enabled.

Data type
String

JMS API Password

The password for the JMS API User ID.

Data type
String

Scheduler Calendar

The JNDI name of the scheduler UserCalendar to be used by the process container.

Data type
String

Activating the business process container

Before you begin

You must have already entered the business process container settings, and created and configured the database and queue resources.

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

Steps for this task

1. Stopping the server.
2. Starting the server.

Results

The process container is ready to run flows.

What to do next

Verify that the process container works.

Verifying that the business process container works

Before you begin

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

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

Steps for this task

1. If you want to run the **Process Choreographer** sample, open the Samples Gallery.
2. **(Optional)** If you want to use your own enterprise application that contains processes:

   Install your application using the administrative console. If it contains processes, the process templates will be written into the Process Choreographer database. The process templates are automatically enabled, and process instances will be created as soon as requests arrive from a client.
3. In case of problems, see: Troubleshooting the business process container.

Results

The business process container is working.

Troubleshooting the business process container

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

Steps for this task

1. If tables or views cannot be found in the database.

   When configuring the data source, you must specify the same user ID that was used to create the database (or to run the scripts to create it).
2. I get a database error when I install an enterprise application that contains a process.

   When an enterprise application is installed, any process templates are written into the Process Choreographer database. Make sure that the database system used by the business process container is running and accessible.
3. Can't invoke Cloudscape tools.

   Make sure that you have set up the Java environment, and have included the necessary JAR files in the `classpath` environment variable.
4. Can't configure a JDBC provider or data source for Oracle

   The jacl scripts provided with Process Choreographer assume that you are using an OCI driver. For performance reasons, a thin driver should not be used.
5. Check your process properties settings.

   Process choreographer uses some settings stored in the file `bpe.properties`, in the `lib` subdirectory of the WebSphere installation directory.
6. An error occurs when process-related Java classes are generated or compiled.

The business process container property `DirectoryForGeneratedClasses` must point to an existing directory. Check the entry in the properties file named `bpe.properties` in the WebSphere `lib` directory. Be sure that the directory specified exists and that the path name is specified using forward slashes or double backslashes.

7. Problems using national characters.

   Make sure that your database was created with support for Unicode character sets.
8. Problem creating the queues using MQSeries: MQSeries dll not found.

   Add the MQSeries Java lib directory to your `path` environment variable. For example, on Windows, enter the command:

   ```
   set path=%MQ_HOME%\java\lib;%path%
   ```
9. Problem on AIX® connecting to the queue manager.

   Edit the file `/var/mqm/mqs.ini`, and add the following property to the definition for your queue manager:

   ```
   IPCCBaseAddress=12
   ```
10. Preventing deadlocks in DB2.

    To avoid deadlocks, be sure that the DB2 isolation level is set to 'read stability'. If necessary, enter the command `db2set DB2_RR_TO_RS=YES` and restart the DB2 instance to activate the change.
11. If you get an 'Error accessing Process MBeans' error message about having 'insufficient or empty credentials' when staring or stopping process templates.

    You must start or stop process templates using a Console User ID that has either an 'operator' or 'administrator' role.

Deleting the business process container

Before you begin

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

To delete the business process container, in the Administrative Console:

Steps for this task

1. Select **Servers > Local Server > Business process container**.
2. Click **Delete**.

Results

The business process container has been deleted.

Managing processes

Before you begin

For each application server where you want to use Process Choreographer, the business process container must be installed and configured.

Managing processes involves the following:

Steps for this task

1. Installing process applications.
2. Uninstalling process applications.
3. Stopping and starting process templates.

Installing process applications

To install an enterprise application that contains process modules:

Steps for this task

1. Make sure that your application server's business process container is configured.
2. Install your application on your application server.

Results

All process modules in the enterprise application (all FAR files contained in the EAR file) are started automatically. This means that instances of the process templates will be created as soon as any requests arrive.

Uninstalling process applications

To uninstall an enterprise application that contains process modules:

Steps for this task

1. Stop all process templates in the process application by "Stopping and starting process templates" in the application. This prevents new process instances from being created.
2. Make sure that no process instances are running. If necessary, a process administrator can use the "Using the Process Choreographer Web client" on page 32 to stop running process instances.
3. To stop the application and uninstall it:
   a. Select **Applications > Enterprise Applications** from the navigation pane on the left.
   b. Select the application you want to stop.
   c. Click **Stop**.
   d. Click **Uninstall**.

Stopping and starting process templates

When an enterprise application that contains process modules is installed and deployed, the process templates that are contained in the process modules are started automatically. You can stop these process templates if required, and restart them again. To do this:

Steps for this task

1. Be sure that you are using a Console User ID that has either the role `operator` or `administrator`.
2. In the navigation pane of the administrative console, select **Applications > Enterprise Applications**.
3. Click the application you want to manage.
4. In the Related Items section of the configuration properties page, click **Business Process Modules**.

5. Select the process module you want to manage.
6. In the Additional Properties section, click **Templates**.

   The Process Templates page is displayed. It lists the process templates that are contained in the selected process module.

   You can stop a running process template, and start it again at a later point in time, depending on the Valid from Time that is specified. The Valid from Time is specified in Coordinated Universal Time (UTC). It specifies when a process template is started. A process template will not be started until the Valid From Time is reached.
7. To stop a process template, check the checkbox next to the process template and click **Stop**.
8. To start a process template, check the checkbox next to the process template and click **Start**.

Process modules collection

Use this page to view a list of the process modules defined for this application.

A process module contains process templates. When a process module is started, all its templates are enabled, and instances of the process templates can be created.

To view this administrative console page, click **Servers > Applications >** *application_name* **> Process Modules** .

Process module settings

Use this page to manage process modules.

A business process module contains one or more business process templates.

To view this administrative console page, click **Applications > Applications >** *application_name* **> Process Modules >** *processmodule_name*.

Process templates collection

Use this page to manage business process templates associated with a process module.

To view this administrative console page, click **Applications > Applications >** *application_name* **> Process Modules >** *processmodule_name* **> Templates**.

Configuration tab

Name Name of the process template.

Data type
String

Valid From
Specifies the point in time from which the process can be started.

You can stop a running process template, and start it again at a later point in time, depending on the Valid From time that is specified.

Data type
String

Units Coordinated Universal Time (UTC)

Current State
Specifies the current state of the process template.

Range

Started
The selected process template is started, which means that new instances of it will be created when request messages arrive.

Stopped
The selected process template is finished and then stopped. No new instances will be created.

Process template settings

Use this page to modify business process template settings.

A process template is the definition of one particular business process.

To view this administrative console page, click **Applications > Applications >** *application_name* **> Process Modules >** *processmodule_name* **>** *processtemplate_name*.

Configuration tab

Name Name of the process template.

Data type
String

Valid From
Specifies the point in time from which the process can be started.

You can stop a running process template, and start it again at a later point in time, depending on the Valid From time that is specified.

Data type
String

Units Coordinated Universal Time (UTC)

Current State
Specifies the current state of the process template.

Range

Started
The selected process template is started, which means that new instances of it will be created when request messages arrive.

Stopped
The selected process template is finished and then stopped. No new instances will be created.

Using the Process Choreographer Web client

This set of topics describes how you can use the Process Choreographer Web client to work with business processes (and their activities) that you have deployed in WebSphere Application Server Enterprise.

Before you begin

Before you can use the Process Choreographer Web client, you must have completed the following prerequisite steps:

1. Configured the Business Process Container. This step enables the Process Choreographer service and the Process Choreographer Web client.

2. Configured the WebSphere Application Server security environment for secured applications, including assigning users and groups to roles defined in business process applications and configuring authentication mechanisms. For more information about configuring the WebSphere Application Server security environment for secured applications, see Managing secure applications.
3. Deployed the applications that use the business processes.

You can use the Process Choreographer Web client to display information about process templates, processes, and activities. You can also act on processes and activities; for example, to start new processes or to claim and complete an activity in your To Do list.

The process templates, processes, and activities that you can see depend on the authority that has been assigned to your user ID or the role that you are working as.

The following tasks describe how to use the Process Choreographer Web client to work with business processes that you have deployed in WebSphere Application Server Enterprise.

- Starting the Process Choreographer Web client (See the on-line documentation for more information)
- Working with business process activities (See the on-line documentation for more information)
  - Displaying information about an activity (See the on-line documentation for more information)
  - Claiming an activity (See the on-line documentation for more information)
  - Completing an activity (See the on-line documentation for more information)
- Working with business processes (See the on-line documentation for more information)
  - Working with business processes you administer (See the on-line documentation for more information)
  - Working with business processes you have started (See the on-line documentation for more information)
  - Displaying information about a business process template (See the on-line documentation for more information)
  - Starting a new business process (See the on-line documentation for more information)
  - Displaying information about a business process (See the on-line documentation for more information)

To customize the Process Choreographer Web client, refer to the white paper in the WebSphere Application Server Library at http://www.ibm.com/software/webservers/appserv/library.

Starting the Process Choreographer Web client

This topic describes how to start the Process Choreographer Web client, which you can use to work with business processes (and their activities) that you have deployed in WebSphere Application Server Enterprise.

Before you begin

Before you can start the Process Choreographer Web client, you must have completed the following prerequisite steps:

1. Configured the Business Process Container. This step enables the Process Choreographer service and the Process Choeographer Web client.
2. Configured the WebSphere Application Server security environment for secured applications, including assigning users and groups to roles defined in business process applications and configuring authentication mechanisms. For more information about configuring the WebSphere Application Server security environment for secured applications, see Managing secure applications.
3. Deployed the applications that use the business processes.

To start the Process Choreographer Web client, complete the following steps.

Note: If you want to work with business process applications on several application servers at the same time, you need to start a Process Choreographer Web client for each application server.

Steps for this task

1. Use a Web browser to open the following URL:

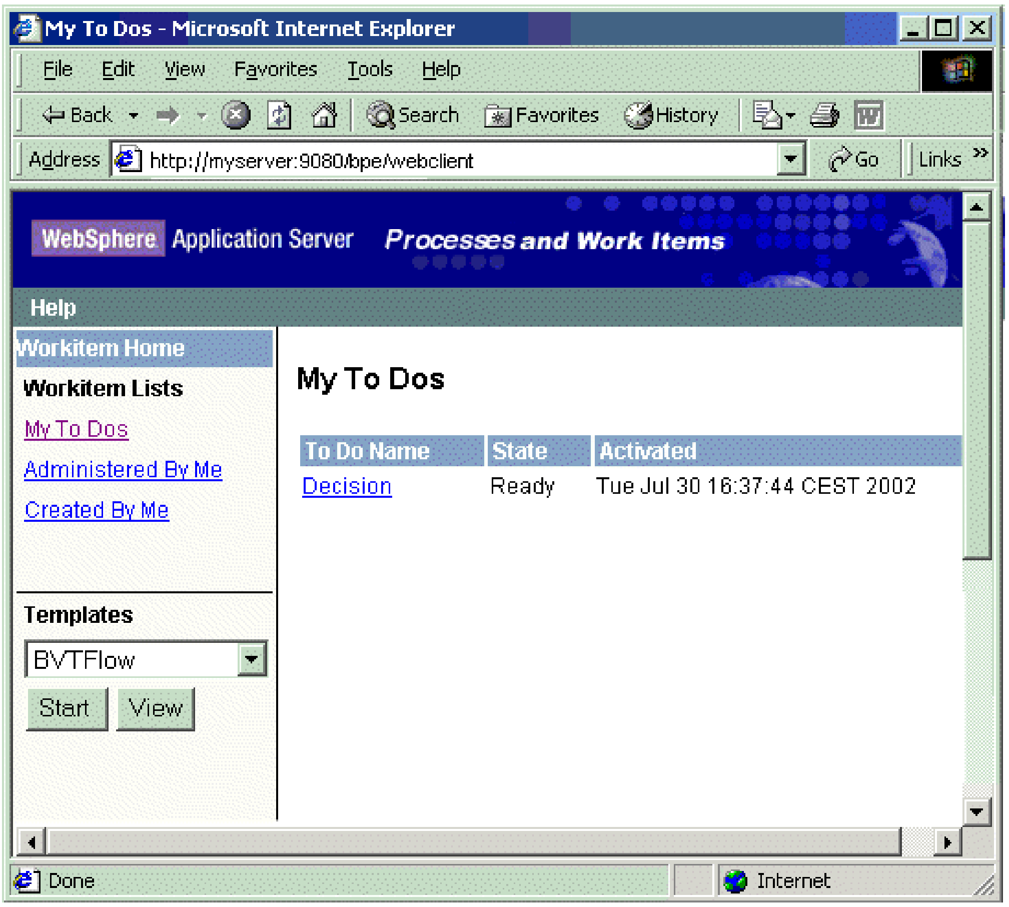
   ```
   http://app_server_host:9080/bpe/webclient
   ```

   Where *app_server_host* is the network name for the host for the application server that provides the business process application that you want to work with.

   If security is enabled, the Web client displays an authentication window for you to provide a user ID and password.
2. If the authentication window is displayed, type a user ID and password then click **OK**.

Results

This task displays the initial page of the Process Choreography Web client, which shows the work item lists and the activities in your To Do list; for example, as shown in the following figure:


My To Dos - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://myserver:9080/bpe/webclient
Go Links
WebSphere Application Server Processes and Work Items
Help
Workitem Home
Workitem Lists
My To Dos
Administered By Me
Created By Me
Templates
BVTFlow
Start View
Done
Internet

My To Dos

To Do Name	State	Activated
Decision	Ready	Tue Jul 30 16:37:44 CEST 2002

Figure 1. The Process Choreographer Web client

The Process Choreographer Web client. The initial page of the Process Choreographer Web client, showing 1) the navigation pane that contains features for working with work item lists and process templates, and 2) the content pane that shows the activities in the To Do list.

Working with business process activities

This set of topics describes how you can use the Process Choreographer Web client to work with activities of interruptible business processes.

For a non-interruptible process there is no human interaction, and the process output message is displayed immediately after the process finishes.

For an interruptible process, you can use the Process Web client to work with activities that require human interaction. If you are authorized as a potential owner, editor, or reader of a new activity, it is added automatically to your To Do list which is displayed in the My To Dos page of the Web client.

To work with an activity, you must claim the activity then perform the actions needed to complete it.

When you complete an activity, it passes its output message to the next activity in the process. The new activity is added to the To Do list of the potential owners, editors, or readers of that activity.

You can display information about an activity that is in your To Do list, and can display information about the process that the activity is part of. You may also have been authorized to view information about other activities that you cannot otherwise work with.

Note: The Process Choreographer Web client does not refresh the My To Dos page automatically. If you change an activity, or want to check the latest contents of your To Do list, you need to refresh the My To Dos page manually.

You can use the following tasks to work with activities:
- Displaying activities in your To Do list (See the on-line documentation for more information)
- Displaying information about an activity (See the on-line documentation for more information)
- Claiming an activity (See the on-line documentation for more information)
- Completing an activity (See the on-line documentation for more information)

Displaying activities in your To Do list
This set of topics describes how you can use the Process Choreographer Web client to display the list of business process activities in your To Do list.

If you are authorized to work with a new activity, it is added automatically to your To Do list. You can view your To Do list by displaying the My To Dos page of the Web client. The My To Dos page is displayed when you first start the Web client, and you can return to that page at any time from other Web client pages.

From the My To Dos page, you can select an activity to work with; for example, to claim an activity that you want to complete.

To display the My To Dos page, click **My To Dos** under Workitem Lists in the navigation pane of the Web client.

Results

This task displays the My To Dos page of the Process Choreographer Web client.

Displaying information about an activity
This topic describes how you can use the Process Choreographer Web client to display information about a business process activity.

You can use the Web client in the following ways to view information about any activity it displays:
- On the My To Dos page, click the name of the activity.

  This displays the Activity page, which provides the set of information and actions needed to work with the activity in its current state.
- On the Process page, click the name of the activity listed under Activities.

  This displays the Activity page, which provides the set of information and actions needed to work with the activity in its current state.
- On the Activity page, click **View more details about this activity**

  This displays the Activity Information page, which provides a more detailed set of information about the activity. The Activity Information page also provides information about the process that the activity is part of.

Claiming an activity

This topic describes how you can use the Process Choreographer Web client to claim a business process activity that you want to work on.

To work with an activity, you must claim the activity then perform the actions needed to complete it. You can claim an activity that is in the ready state if you are a potential owner of that activity. If you claim an activity, you become the owner of that activity and are responsible for completing it.

You can claim a ready activity from its Activity page; for example, by completing the following steps:

Steps for this task

1. Click **My To Dos** under Workitem Lists in the navigation pane of the Web client.

   This action displays the My To Dos page, which lists the states of activities that you can work with.
2. Click the activity name.

   This action displays the Activity page, with **Claim Activity** displayed as an available action. This page also displays the required user input and the current output message for the activity. You cannot change the fields displayed until you have claimed the activity.
3. **(Optional)** For more detailed information about the activity, you can click **View more details about this activity**, but should return to the Activity page to claim the activity.
4. **(Optional)** Click **Claim Activity**.

Results

When you have claimed an activity, the Activity page is displayed with the required user input fields and output message fields enabled for you to work with.

What to do next

You should use the Activity page to complete the activity, as described in “Completing an activity”.

Completing an activity

This topic describes how you can use the Process Choreographer Web client to complete a business process activity that you have claimed.

Before you begin

Before you can complete an activity, you must have claimed the activity, as described in “Claiming an activity”.

This topic describes the task of working with an activity up until you click the action that finally completes that activity.

When you have claimed an activity, you can perform the actions needed to complete it.

You do not have to complete an activity in one step. While working with an activity, you can save changes you make to the activity and leave the activity to do other things. For example, you may want to save your changes if your work is interrupted or if you need to get more information before completing the activity.

To complete an activity that you have claimed, use the following steps:

Steps for this task

1. Display the Activity page; for example, by clicking the name of the activity on your My To Dos page.
2. Complete the required user inputs shown.

   The **Required User Inputs** section displays fields for input that you must specify before you can complete the activity. The content of this section, including the number and types of input fields, depends on the design of the activity within the process template.
3. **(Optional)** If appropriate, change the output message shown.

   The **Activity Output Message** section displays fields of the output message to be passed to the next activity when you complete this activity. The content of this section, including the number and types of input fields, depends on the design of the activity within the process template.
4. **(Optional)** If you want to save your changes, click **Save**.

   For example, you may want to save your changes if your work is interrupted or if you need to get more information before completing the activity.
5. If you have finished all the required user inputs, and want to complete the activity, click **Complete Activity**.

   Note: If you click **Complete Activity**, you will not be able to make any more changes to the activity,

   If you have specified appropriate required user input, the activity is completed, and the Web client window refreshed to display your My To Dos page.

   If you have not specified required user input, the Activity page displays messages that indicate which of the user inputs you should act on.

Results

When you complete an activity, it passes its output message to the next activity in the process. The new activity is added to the To Do list of the potential owners of that activity. If you complete the last activity in a process, the output message is passed to the person who started the process.

Working with business processes

This set of topics describes how you can use the Process Choreographer Web client to work with business processes.

You can use the Web client to display information about business processes or act on business processes. The Web client displays only those processes that you are authorized to work with, and displays only those actions that you are authorized to use on the processes.

Also, if you are authorized to start new processes, you can display information about their process templates.

You can use the following tasks to work with activities:

- Working with business processes you administer (See the on-line documentation for more information)

- Working with business processes you have started (See the on-line documentation for more information)
- Displaying information about a business process template (See the on-line documentation for more information)
- Starting a new business process (See the on-line documentation for more information)
- Displaying information about a business process (See the on-line documentation for more information)

Working with business processes you administer

This topic describes how you can use the Process Choreographer Web client to work with a business process that you are authorized to administer.

Before you begin

To administer processes, you need to have administer authority for the business process.

The Administered By Me page of the Web client displays a list of only those processes that you can administer.

To work with a process on the Administered By Me page, complete the following steps:

Steps for this task

1. To display the Administered By Me page, click **Administered By Me** under Workitem Lists in the navigation pane of the Web client.
2. To work with a process, click the name of the process.

   If the process is running, the Web client displays a Process page with information and available actions for the process. The information displayed includes a list of the current activities in the process. Optionally, you can click an activity name to work with that activity.

   If the process has ended, the Web client displays the Output Message page for the process.
3. **(Optional)** To act on the process, click an action displayed under Available Actions.

   Any actions displayed depend on the current state of the process. For example, if a process is in the running state, **Terminate Process** is displayed. If you needed to stop the process without its activities completing normally, you could click **Terminate Process**.

Working with business processes you have started

This topic describes how you can use the Process Choreographer Web client to work with a business process that you have started.

The Created By Me page of the Web client displays a list of the processes that you have started.

To work with a process on the Started By Me page, complete the following steps:

Steps for this task

1. To display the Created By Me page, click **Created By Me** under Workitem Lists in the navigation pane of the Web client.
2. To work with a process, click the name of the process.

   If the process is running, the Web client displays a Process page with information and available actions for the process. The information displayed includes a list of the current activities in the process. Optionally, you can click an activity name to work with that activity.

   If the process has ended, the Web client displays the Output Message page for the process.
3. **(Optional)** To act on the process, click an action displayed under Available Actions.

   Any actions displayed depend on your authorization and the current state of the process. For example, if the process is in the running state, **Terminate Process** is displayed. If you needed to stop the process without its activities completing normally, you could click **Terminate Process**.

Displaying information about a business process template

This topic describes how you can use the Process Choreographer Web client to view information about a process template from which business processes can be started.

To display information about a process template, complete the following steps:

Steps for this task

1. Select a process template from the Templates drop-down list in the navigation pane.
2. Click **View**.

Results

This task displays a Process Template page with information about the process template. If you are authorized to start a process from the template, a **Start Process** action is also displayed.

Starting a new business process

This topic describes how you can use the Process Choreographer Web client to start a new business process.

You can start a new business process from any of the process templates that you are authorized to use.

To start a new business process, complete the following steps:

Steps for this task

1. Select a process template from the Templates drop-down list in the navigation pane.
2. Click **Start**.

   This displays the Process Input Message page in the content pane, which provides a description of the process template, fields for you to change the input message properties, and a button for you to complete starting the process.

3. **(Optional)** If appropriate, change the values of the process input message properties.
4. To complete starting the process, click the available action **Start Process**.

Results

If the business process contains an activity that needs user input, the activity is added to the To Do list of potential owners. If you are one of those potential owners, you can display your My To Dos page to work with the activity.

When the business process ends, it displays the output message for the process.

Displaying information about a business process

This topic describes how you can use the Process Choreographer Web client to display information about a business process.

You can use the Web client in the following ways to view information about any business process it displays:

- To display information about a process that you started:
  1. Click **Created By Me** under Workitem Lists in the navigation pane of the Web client.
  2. In the content pane, click the name of the process.
- To display information about a process that you administer:
  1. Click **Administered By Me** under Workitem Lists in the navigation pane of the Web client.
  2. In the content pane, click the name of the process.
- On an Activity page, click **View more details about this process**

Results

If the process is running, the Web client displays a Process page with information and available actions for the process. The information displayed includes a list of the current activities in the process. Optionally, you can click an activity name to display information about that activity.

If the process has ended, the Web client displays the Output Message page for the process.

Process Choreographer Web client

This topic provides links to information about the pages of the Process Choreographer Web client and describes the features that are common to all those pages.

The Process Choreographer Web client comprises the following pages that you can use to work with business processes and activities:

- Process Template page, which displays details about a process template. If appropriate, you can use this page to start new processes.
- Process Input Message page, which displays properties of the process template and the input message used to start a business process. You can use this page to change input message properties and complete starting a process.
- Process Output Message page, which displays the results of a business process that you started.

- Administered By Me page, which displays a list of the business processes that you can administer. You can use this page to select processes to administer.
- Created By Me page, which displays a list of the business processes that you have started. You can use this page to select processes to display more information or work with a process.
- Process page, which displays information about a business process. If appropriate, you can use this page to terminate the process.
- My To Dos page, which displays a list of the activities in your To Do list. You can use this page to work with the activities; for example, to claim an activity.
- Activity page, which displays information about an activity. If appropriate, you can use this page to work with the activity.
- Activity Information page, which displays detailed information about an activity.

The Web client pages display the following common features, as shown in the figure The Process Choreographer Web client.

- Navigation pane.

  This displays the following sections that you can use to select activities, processes, or process templates:

  – Workitem Lists

    You can use this section to work with activities in you To Do list and to work with processes that you can administer or have created.

  – Templates

    You can use this section to display details about a process template or to start a process from a template.
- Content

  You can use this pane to work with processes and activities. The content of this pane depends on the actions that you have taken with the Process Choreographer Web client. When you first start the Web client, this pane displays the activities in your To Do list.

The Process Choreographer Web client. This figure show the initial page of the Process Choreographer Web client, which contains the following features: A navigation pane containing Workitem Lists and a Templates drop-down list; and a content pane. The use of the Process Choreographer Web client through these features is described in the information that accompanies this figure.

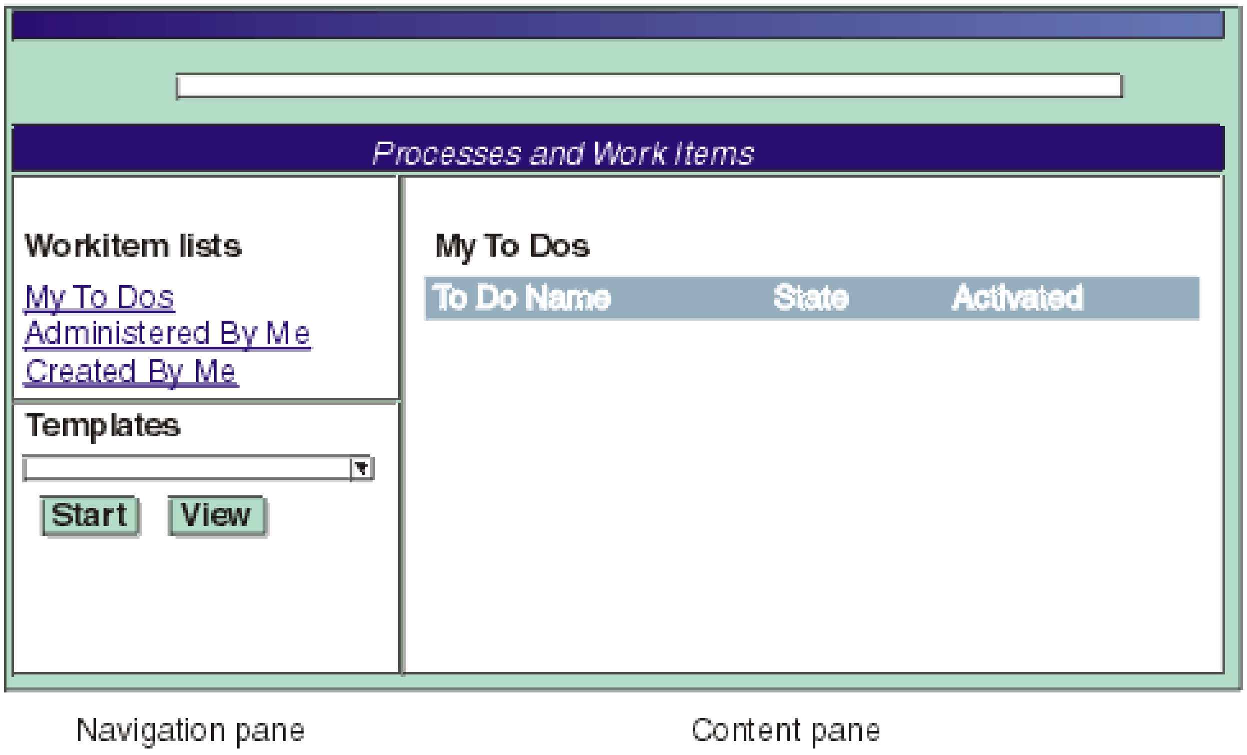


Figure 2. The Process Choreographer Web client

Workitem Lists

Use this section to select a type of work item that you can work with.

This section displays a list of the following work item types. Each entry in the list is a link that you can click to work with the type of work item.

My To Dos
: Click this link to work with activities in your To Do list.

Administered By Me
: Click this link to work with processes that you are an administrator for.

Created By Me
: Click this link to work with processes that you started.

Templates

Use this section to start a process from a process template or to display details about a process template.

This section displays a drop-down list of process templates, from which you can select a template, and the following buttons for you to work with a selected template.

Start Click this button to start a process from a template.

View Click this button to display details about a template.

Process Template page

The Process Template page of the Process Choreographer Web client displays details about a process template, from which you can start process instances.

Use this page to view information about a process template, which provides the design for business processes. If you are authorized to do so, you can start processes based on the template from this page.

Template Name
The name of the process template.

Version
The version of the template.

Created
The date and time when the template was created.

Valid From
The date and time when the template was officially put into use.

Some of the fields displayed are usually of interest only to business process administrators or designers.

If you are authorized to start a process from this template, you can see a **Start** button. To start a process, click **Start**.

Process Input Message page

The Process Input Message page of the Process Choreographer Web client displays properties of the process template and the input message used to start a business process.

This page is displayed when first starting a business process that requires an input message. You can use this page to change the properties of the input message before you complete the actions to start the business process.

This page displays the following main sections:

- Process Template Description

  This section provides information about the properties of the process template from which the process is being started.
- Process Input Message

  This section provides entry fields for the properties of the input message. The number and types of properties depends on the process template.

To complete the actions to start the process, take one of the following actions:

- If you want to change the properties of the input message, use the fields provided, then click **Start**.
- If you want to start the process with the default input messages properties provided, then click **Start**.

Process Output Message page

The Process Output Message page of the Process Choreographer Web client displays the results of a business process that you started.

This page is displayed when a business process ends, if you started that business process and it is created to generate an output message. You can use this page to view information that was generated by a business process.

The information displayed by the output message depends on the process template that defines the model for the process.

When you have finished with this page, you can use the navigation pane to work with other activities or processes, or can stop the Process Choreographer Web client.

Administered By Me page

The Administered By Me page of the Process Choreographer Web client displays a list of the business processes that you can administer, with a summary of information about those processes.

This page is displayed if you click the Administered By Me link in the Workitem Lists section of the navigation pane. You can use this page to display the following information about each process that you can administer and, optionally, to select a process to administer.

Owner
The owner of the process.

Process
The business process identifier for the process. This is a link that you can use to administer the process.

State The current state of the process.

Started
The date and time that the process was started.

Template Name
The name of the process template for the process.

To administer a process, click the process identifier in the list. This displays the Process page, which provides details about the process and, if appropriate, provides buttons for you to act on the process.

Created By Me page

The Created By Me page of the Process Choreographer Web client displays a list of the business processes that you have started, with a summary of information about those processes.

This page is displayed if you click the Created By Me link in the Workitem Lists section of the navigation pane. You can use this page to display the following information about each process that you have startedand, optionally, to select a process to work with.

Owner
The owner of the process.

Process
The business process identifier for the process. This is a link that you can use to administer the process.

State The current state of the process.

Started
The date and time that the process was started.

Template Name
The name of the process template for the process.

To work with a process, click the process identifier in the list. This displays the Process page, which provides details about the process and, if appropriate, provides buttons for you to act on the process.

Process page

The Process page of the Process Choreographer Web client displays information about a business process, including its current and completed activities.

This page is displayed if you click a business process link on the Administered By Me page or Created By Me page or another link for more information about a process. You can use this page to display information about the process and, optionally, to act on the process. You can also view the activities for the process and, optionally, select an activity to work with.

This page displays the following sections:

Available Actions
Buttons that you can use to work with the process. This section is displayed only if you have authority to act on the process in its current state.

For example, if you are an administrator for the business process, and the process is running, a **Terminate Process** button is displayed. If you want to terminate the process, click **Terminate Process**.

Terminate
Terminates the process.

Process Description
A table of information about the process, including:

Process Name
The identifier for the process.

Process Description
A brief description of the process.

Template Name
The name of the business process template from which the process was started.

Starter
The user ID that started the process.

State The current state of the process.

Started
The date and time when the process was started.

Activities
Information about the activities for the process:

Name The name of the activity. Each name is a link that you can use to select an activity to work with.

State The current state of the activity.

Activated
The date and time when the activity was started.

Completed
The date and time when the activity was completed.

My To Dos page

The My To Dos page of the Process Choreographer client displays the list of activities in your To Do list.

This page is displayed if you click the My To Dos link in the Workitem Lists section of the navigation pane. You can use this page to display the following information about each activity that you can work with and, optionally, to select an activity to work with.

To Do Name
The name of the activity. This is a link that you can click to work with the activity.

State
The current state of the activity

Activities in Ready state appear on the My To Dos page of all potential activity owners. If you claim such an activity, you become the activity owner (its state is changed to Claimed) and are responsible for its completion. Activities displayed in Claimed state have been claimed by the user listed in the Owner column.

Started
The date and time that the activity was started.

Owner
The owner of the activity.

Reason
The reason for starting the activity.

To work with an activity, click the activity name under **To Do Name** in the list. This displays the Activity page, which provides details about the activity and, if appropriate, provides buttons for you to act on the activity.

Activity page

The Activity page of the Process Choreographer Web client displays information about an activity and, if appropriate, buttons for you to work with the activity.

This page is displayed if you click **Claim Activity** on the Activity Information page or the Activity page displayed if you click a link for an activity in the Ready state. You can use this page to work with the activity and, optionally, to display more information about the activity or its business process.

The Activity page displays the following sections:

Available Actions
Buttons that you can use to work with the activity. This section is displayed only if you have authority to act on the activity in its current state; for example:

Claim Activity
Displayed if the activity is in a Ready state and you are a potential

owner of the activity. If you click **Claim Activity**, you become the activity owner (its state is changed to Claimed) and are responsible for its completion.

Save Changes
Displayed if the activity is in a Claimed state and you are the owner of the activity. If you change the user input properties or the output message properties, you can click **Save Changes**, to save the changes without completing the activity. For example, you might save changes to an activity if your work is interrupted or if you need to obtain further information.

If you click **Save Changes**, the Activity page is replaced by the My To Dos page. You can display the Activity page later to complete the activity.

Complete Activity
Displayed if the activity is in a claimed state and you are the owner of the activity. If you have specified appropriate required user input, and want to use the activity output message shown, you can click **Complete Activity** to complete the activity.

Process Context
This section lists the name and description of the activity. It also provides the following links for more information about the activity and its business process:

- View more details about this activity
- View more details about this process

Required User Input
This section displays fields for input that you must specify before you can complete the activity. The content of this section, including the number and types of input fields, depends on the design of the activity within the process template.

If you have not claimed the activity, you cannot change the fields in this section.

Activity Output Message
This section displays fields of the output message to be passed to the next activity when you click **Complete Activity** to complete this activity. The content of this section, including the number and types of input fields, depends on the design of the activity within the process template.

If you have not claimed the activity, you cannot change the fields in this section.

Activity Information page

The Activity Information page of the Process Choreography Web client displays detailed descriptions of the activity and the business process that it is part of.

This page is displayed if you click a link for more information about an activity; for example, on the Activity page. You can use this page to view details about the activity and its process.

The Activity Information page displays the following sections:

Activity Description
This section lists the details about the properties of the activity. The content

of this section depends on the design of the activity within the process template, the state of the activity, and your authority for the activity. For example, this section can display the following details:

Activity Name
The name of the activity.

Description
A short description of the activity

Potential Owners
A list of the user IDs for potential owners of this unclaimed activity.

It can also provide links, for example, for you to Work with this activity.

Process Description
This section lists the details about the properties of the business process that the activity is part of. The content of this section depends on the design of the activity within the process template. For example, this section can display the following details:

Process Name
The identifier for the process.

Process Description
A brief description of the process.

Template Name
The name of the business process template from which the process was started.

Starter
The user ID that started the process.

State The current state of the process.

Started
The date and time when the process was started.

It can also provide links, for example, for you to View more details about this process.

The Process Choreographer Web client

You can use the Process Choreographer Web client to work with business processes (and their activities) that you have deployed as applications in WebSphere Application Server Enterprise.

The Web client runs as a Web application on the host where your business process application is deployed:

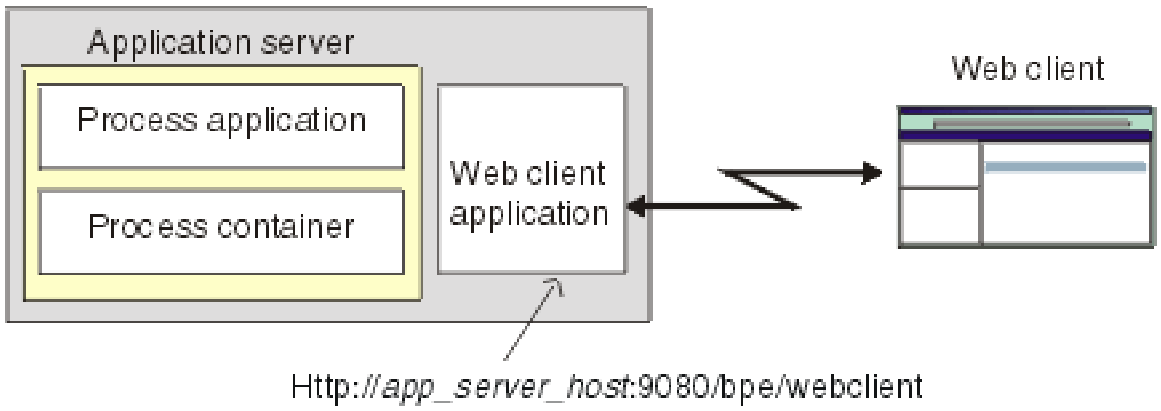


Figure 3. The Process Choreographer Web Client relative to the business process application

Note: If you want to work with business process applications on several application servers at the same time, you need to start a Process Choreographer Web client for each application server.

You can use the Web client to display information about process templates, processes, and activities. You can also act on processes and activities; for example, to start new processes or to claim and complete an activity in your To Do list.

The main concepts for using the Web client are as follows:

My To Do list
: This lists the activities that you own or have been assigned to you as a potential owner. This list is displayed in the My To Dos page of the Web client.

  When you claim an activity, you become the owner of that activity.

Process template
: A process template defines the activities that form a business process, and is used to start a new business process. You can use the Web client to display the process templates that you can work with.

Processes you have started
: If you start a new process, you are registered as the creator of that process. You can use Created By Me page of the Web client to display the processes that you have started.

Processes you administer
: You can be authorized as the administrator for a process template. You can use Administered By Me page of the Web client to display the processes that you can administer.

  As an administrator, you can use the Web client to manage processes and, if needed, to terminate processes that have problems.

For more information about the pages of the Web client, see "Process Choreographer Web client" on page 41.

The actions you can take through the Web client depends on the role that you have been assigned, as follows:

Actions for process roles

Action	Administrator	Starter	Reader
Start process	Yes	Yes	–
Terminate process	Yes	–	–
Delete process	Yes	–	–
Display process	Yes	Yes	Yes
Force restart activity	Yes	–	–
Claim activity	Yes	–	–
Save activity	Yes	–	–
Display activity	Yes	–	–
Complete activity	Yes	–	–
Force complete stopped activity	Yes	–	–

Actions for activity roles

Action	Potential owner	Owner	Reader	Editor
Claim activity	Yes	See note	–	–
Save activity	Yes	Yes	–	Yes
Display activity	Yes	Yes	Yes	Yes
Complete activity	Yes	Yes	–	–

Note: If a potential owner claims an activity, the potential owner becomes the owner of the activity.

Actions for system administrator role

Action	System administrator
Start process	Yes
Terminate process	Yes
Delete process	Yes
Display process	Yes
Force restart activity	Yes
Claim activity	Yes
Save activity	Yes
Display activity	Yes
Complete activity	Yes
Force complete stopped activity	Yes

Developing applications using the Process Choreographer API

The Process Choreographer API provides the following renderings for developing applications:

- An EJB rendering that allows the API to be called remotely using IIOP. A stateless session bean, BusinessProcess, exposes the functions that can be called by an application program.

- A JMS rendering that allows a subset of the API functions to be called remotely using JMS.

You can use the API to develop the following types of applications:
- Business-process applications for non-interruptible processes
- Business-process applications for interruptible processes
- Administration applications for interruptible processes

For more information, see the Javadoc.

Developing applications for non-interruptible processes

You can develop the following applications for non-interruptible processes:
- Execute a non-interruptible process using the EJB interface
- Execute a non-interruptible process using the JMS interface

Also see the Javadoc.

Executing a non-interruptible process using the EJB interface

Steps for this task

1. **(Optional)** List the process templates to find the name of the non-interruptible process you want to execute.

   This step is optional if you already know the name of the process.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.CAN_RUN_SYNC=TRUE",
    "PROCESS_TEMPLATE_NAME",
    newInteger(50),
    null);
   ```

   The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as non-interruptible processes.
2. Start the process with an input message.

   In the following, Customer and OrderNo are message types known to the system.

   ```
   Customer input = new Customer("Smith");
   ...
   ClientObjectWrapper output = process.call("CustomerTemplate", new ClientObjectWrapper(input));
   OrderNo order = (OrderNo) output.getObject();
   ```

   This 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.

Queries on business-process objects: You can use the process query interface to retrieve business-process information that is stored persistently. You use SQL-like syntax to query the following objects:
- Process templates
- Process instances
- Activity instances
- Work items

The query function is provided by the BusinessProcess session bean's remote interface. For process templates, the query function has the following syntax:

```
queryProcessTemplates (java.lang.String whereClause,
                       java.lang.String orderByClause,
                       java.lang.Integer threshold,
                       java.util.Timezone timezone);
```

For the other business-process objects, the query function has the following syntax:

```
QueryResultSet query (java.lang.String selectClause,
                       java.lang.String whereClause,
                       java.lang.String orderByClause,
                       java.lang.Integer threshold,
                       java.util.Timezone timezone);
```

The query is made up of:

- "Select clause" on page 57
- "Where clause" on page 58
- "Order-by clause" on page 58
- "Threshold parameter" on page 59
- "Timezone parameter" on page 59

For example, a list of work items accessible to the caller of the function is retrieved by:

```
QueryResultSet result = process.query("WORK_ITEM.WIID",
                                      null, null,
                                      null, null);
```

The query function returns objects according to the caller's authorization. The query result set contains only those objects that the caller is authorized to see.

For more information, see the Javadoc.

Predefined views for queries on business process objects: Process Choreographer provides the following predefined views for queries on business process objects:

- "PROCESS_TEMPLATE view"
- "PROCESS_ATTRIBUTE view" on page 54
- "PROCESS_INSTANCE view" on page 54
- "ACTIVITY view" on page 55
- "ACTIVITY_ATTRIBUTE view" on page 56
- "EVENT view" on page 56
- "WORK_ITEM view" on page 56

Table 1. PROCESS_TEMPLATE view

Column name	Type	Comments
PTID	ID	Process template ID.
NAME	String	Name of the process template.
APPLICATION_NAME	String	Name of the enterprise application to which the process template belongs.
VALID_FROM	Timestamp	The time from when the process template can be instantiated.
VERSION	String	User-defined version.

Table 1. PROCESS_TEMPLATE view (continued)

Column name	Type	Comments
CREATED	Timestamp	The time the process template is created in the database.
STATE	Integer	Specifies whether the process template is available for process instances to be created. Possible values: `STATE_STARTED` `STATE_STOPPED`
DESCRIPTION	String	Description of the process template.
CATEGORY	String	The category to which the process template belongs.
CAN_RUN_SYNC	Boolean	Specifies if the process can run non-interrupted.
CAN_RUN_INTERRUP	Boolean	Specifies if the process can run interrupted.

Table 2. PROCESS_ATTRIBUTE view

Column name	Type	Comments
PIID	ID	The ID of the process instance that has a customer attribute.
NAME	String	Name of the customer attribute.
VALUE	String	Value of the customer attribute.

Table 3. PROCESS_INSTANCE view

Column name	Type	Comments
PTID	ID	Process template ID.
PIID	ID	Process instance ID.
NAME	String	Name of the process instance.
STATE	Integer	State of the process instance. Possible values: `STATE_READY` `STATE_RUNNING` `STATE_FINISHED` `STATE_COMPENSATING` `STATE_FAILED` `STATE_TERMINATED` `STATE_COMPENSATED` `STATE_TERMINATING` `STATE_FAILING`
CREATED	Timestamp	The time the process instance is created.
STARTED	Timestamp	The time the process instance is started.

Table 3. PROCESS_INSTANCE view (continued)

Column name	Type	Comments
COMPLETED	Timestamp	The time the process instance is completed.
PARENT_NAME	String	The name of the parent process instance.
TOP_LEVEL_NAME	String	The name of the top-level process instance. This is the current process instance if there is no top level.
STARTER	String	Principal ID of the starter.
TEMPLATE_NAME	String	The name of the associated process template.
TEMPLATE_DESCR	String	Description of the associated process template.
TEMPLATE_CATEGORY	String	The category to which the associated process template belongs.

Table 4. ACTIVITY view

Column name	Type	Comments
PIID	ID	Process instance ID.
AIID	ID	Activity instance ID.
PTID	ID	Process template ID.
ATID	ID	Activity template ID.
KIND	Integer	Kind of activity. Possible values: `KIND_PROCESS_SUBFLOW` `KIND_PROCESS_BLOCK` `KIND_EMPTY` `KIND_SINK` `KIND_SOURCE` `KIND_ELEMENTAL` `KIND_FAULT` `KIND_PERSON` `KIND_EVENT`
RUN_MODE	Integer	Possible values: `RUN_MODE_SYNCHRONOUS` `RUN_MODE_INTERRUPTIBLE` `RUN_MODE_ATOMIC_SPHERE` `RUN_MODE_CHAINED`
ACTIVATED	Timestamp	The time the activity is activated.
STARTED	Timestamp	The time the activity is started.
COMPLETED	Timestamp	The time the activity is completed.

Table 4. ACTIVITY view (continued)

Column name	Type	Comments
STATE	String	State of the activity. Possible values: `STATE_INACTIVE` `STATE_READY` `STATE_RUNNING` `STATE_SKIPPED` `STATE_FINISHED` `STATE_FAILED` `STATE_TERMINATED` `STATE_CLAIMED` `STATE_TERMINATING` `STATE_FAILING` `STATE_WAITING` `STATE_EXPIRED` `STATE_STOPPED`
OWNER	String	Principal ID of the owner.
TEMPLATE_NAME	String	Name of the associated activity template.
TEMPLATE_DESCR	String	Description of the associated activity template.

Table 5. ACTIVITY_ATTRIBUTE view

Column name	Type	Comments
AIID	ID	The ID of the activity instance that has a customer attribute.
NAME	String	Name of the customer attribute.
VALUE	String	Value of the customer attribute.

Table 6. EVENT view

Column name	Type	Comments
EIID	ID	The ID of the awaited event.
AIID	ID	The ID of activity waiting for the event.
PIID	ID	The ID of the process instance that contains the event.
NAME	String	Name of the event.

Table 7. WORK_ITEM view

Column Name	Type	Comment
WIID	ID	Work item ID.
OWNER_ID	String	Principal ID of the owner.
EVERYBODY	Boolean	Flag indicating whether everybody owns this work item.

Table 7. WORK_ITEM view (continued)

Column Name	Type	Comment
OBJECT_TYPE	Integer	Type of the associated object. Possible values: `OBJECT_TYPE_ACTIVITY` `OBJECT_TYPE_PROCESS_INSTANCE` `OBJECT_TYPE_EVENT`
OBJECT_ID	ID	ID of the associated object, for example, the associated process or activity.
ASSOC_OBJECT_TYPE	Integer	Type of the object associated with, or containing, the work item's associated object. Possible values: `OBJECT_TYPE_ACTIVITY` `OBJECT_TYPE_PROCESS_INSTANCE` `OBJECT_TYPE_EVENT`
ASSOC_OID	ID	ID of the object associated to or containing the work item's associated object. For example, the PIID of the process instance containing the activity instance for which this work item has been created.
REASON	Integer	The reason the work item was assigned. Possible values: `REASON_POTENTIAL_OWNER` `REASON_EDITOR` `REASON_READER` `REASON_OWNER` `REASON_POTENTIAL_STARTER` `REASON_STARTER` `REASON_ADMINISTRATOR`

Select clause: The select clause describes the query result. It specifies a list of names that identify the object properties (columns of the result) to be returned. Its syntax is the same as an SQL select clause; use commas to separate parts of the clause. Each part of the clause must specify a property from one of the predefined views. The columns returned in the QueryResultSet appear in the same order as the properties specified in the select clause.

Note: The select clause does not support SQL aggregation functions, such as AVG(), SUM(), MIN(), MAX(), or COUNT().

Example select clauses

- ″WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON″

  Gets the object types of the associated objects and the assignment reasons for the work items.
- ″DISTINCT WORK_ITEM.OBJECT_ID″

Gets all object IDs of objects for which the caller has a work item without duplicates.
- ″ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON″

  Gets the names of the activities the caller has work items for and their assignment reason.
- ″ACTIVITY.STATE, PROCESS_INSTANCE.STARTER″

  Gets the states of the activities and the starters of their associated process instances.

If an error occurs during the processing of the select clause, a QueryUnknownTable or QueryUnknownColumn exception is thrown with the name of the property that is not recognized as a table or column name.

Where clause: The where clause describes the filter criteria that are to be applied to the query domain. Its syntax is the same as an SQL where-clause. If you do not want to filter a query, you must specify **null** for the where clause.

The where-clause syntax supports:
- Keywords: AND, OR, NOT
- Comparison operators: =, <=, <, <>, >,>=, LIKE
- Set operation: IN

The LIKE operation supports the wildcard characters defined for the queried database.

The following special rules also apply:
- Specify object ID constants as ID(’string-rep-of-oid’).
- Specify timestamp constants as TS(’yyyy-mm-ddThh:mm:ss’).
- Specify binary constants as BIN(’UTF-8 string’)
- It is recommended that you use symbolic constants instead of integer enumerations. For example, instead of specifying an activity state expression ″ACTIVITY.STATE=2″, specify ″ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY″.

Examples of where clauses
- Comparing an object ID with an existing ID
  ```
  "WORK_ITEM.WIID = ID('_WI:800c00ed.df8d7e7c.feffff80.38')"
  ```
  This type of where clause is usually created dynamically with an existing object ID from a previous call. If this object ID is stored in a variable wiid1, the clause could be constructed as:
  ```
  "WORK_ITEM.WIID = ID('" + wiid1.toString() + "')"
  ```
- Using timestamps
  ```
  "ACTIVITY.STARTED >= TS('2002-06-1T16.00.00')"
  ```
- Using symbolic constants
  ```
  "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER"
  ```
- Using boolean values true and false
  ```
  "PROCESS_TEMPLATE.CAN_RUN_SYNC = TRUE"
  ```

Order-by clause: Use the order-by clause to specify the sort criteria for the query result set. The order-by clause syntax is the same as an SQL order-by clause; use commas to separate each part of the clause. Each part of the clause must specify a property from one of the predefined views.

If you identify more that one property, the query result set is ordered by the values of the first property, then by the values of the second property, and so on.

If you do not want to sort the query result set, you must specify **null** for the order-by clause.

Examples of order-by clauses

- ″PROCESS_TEMPLATE.NAME″

  Sorts the query result alphabetically by the process-template name.
- ″PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC″

  Sorts the query result by the creation date and, for a specific date, sorts the results alphabetically by the process-instance name in reverse order.
- ″ACTIVITY.OWNER, ACTIVITY_TEMPLATE.NAME, ACTIVITY.STATE″

  Sorts the query result by the activity owner, then the activity-template name, and then the state of the activity.

Threshold parameter: The threshold parameter in the query function restricts the number of objects returned in the query result set. This can be useful, for example, in a GUI where only a small number of items should be displayed. If you set the threshold parameter accordingly, it improves the performance; the database query is faster and less data needs to be transferred from the server to the client.

If the parameter is set to **null**, a threshold is not applied and all the qualifying objects are returned.

Example of a threshold parameter

- new Integer(50)

  Specifies that only 50 qualifying objects are to be returned.

Timezone parameter: Timezones can differ between the client that starts the query and the process engine that processes the query. Use the timezone parameter to specify the timezone of the timestamp constants used in the where clause, for example, to specify local times. The dates returned in the query result set have the same timezone as that specified in the query.

If the parameter is set to **null**, the timestamp constants are assumed to be UTC times.

Examples of timezone parameters

-
  ```
  process.query("ACTIVITY.AIID",
            "ACTIVITY.STARTED > TS ('2002-01-01T17:40')",
            null,
            null,
            java.util.Timezone.getDefault() );
  ```

  Specifies that object IDs are to be returned for activities that have been started later than 17:40 local time on 1 January 2002.

-
  ```
  process.query("ACTIVITY.AIID",
            "ACTIVITY.STARTED > TS ('2002-01-01T17:40')",
            null,
            null,
            null;
  ```

  Specifies that object IDs are to be returned for activities that have been started later than 17:40 UTC on 1 January 2002. This is, for example, 6 hours earlier in eastern standard time.

Query results: A query result set contains the results of a query. The elements of the set are objects that the caller is authorized to see. Elements can be read in a relative fashion using the **next()** method or in an absolute fashion using the **first()** and **last()** methods. Because the implicit cursor of a query result set is initially positioned before the first element, you must call either **first()** or **next()** 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.

Note: The numbering of the column indexes starts with 1.

Attribute type	**Method**
String	getString
ID	getOID
Timestamp	getTimestamp getString
Integer	getInteger getShort getLong getString getBoolean
Boolean	getBoolean getShort getInteger getLong getString

Example

The following query is run:

```
xQueryResultSet resultSet = process.query("ACTIVITY.STARTED, ACTIVITY.TEMPLATE_NAME AS NAME, WORK_ITEM.
WIID, WORK_ITEM.REASON",
                                  null,
                                  null,
                                  null,
                                  null);
```

Note: The first line of the previous example wrapped to a second line to fit the width of the page.

The returned query result set has four columns:
- Column 1 is a timestamp
- Column 2 is a string
- Column 3 is an object ID
- Column 4 is an integer

You can use the following methods to retrieve the attribute values:

```
while (resultSet.next())
{
 java.util.Calendar activityStarted = resultSet.getTimestamp(1);
 String templateName = resultSet.getString(2);
 WIID wiid = (WIID) resultSet.getOID(3);
 Integer reason = resultSet.getInteger(4);
}
```

You can use the display names of the result set, for example, as headings for a printed table. These are the column names of the view or the name defined by the **AS** clause. You can use the following methods to retrieve the display names in the example:

```
resultSet.getColumnDisplayName(1) returns "STARTED"
resultSet.getColumnDisplayName(2) returns "NAME"
resultSet.getColumnDisplayName(3) returns "WIID"
resultSet.getColumnDisplayName(4) returns "REASON"
```

Executing a non-interruptible process using JMS

Steps for this task

1. Create a message, for example, an ObjectMessage.

   ```
   ObjectMessage message = queueSession.createObjectMessage();
   ```

2. **(Optional)** Set the JMSReplyToQueue.

   If you do not want to receive a reply, this step is optional.

   ```
   //Specify the destination object replies are to be sent to
   message.SetJMSReplyTo(replyToQueue);
   ```

3. **(Optional)** Specify the JMS properties.

   If you do not specify any properties, the process template name **Dispatch** is assumed. If JMSReplyToQueue is set, **call** is issued. If JMSReplyToQueue is not set, **initiate** is issued.

   ```
   message.SetStringProperty("wf$verb", "call");
   message.SetStringProperty("wf$processTemplateName", "CustomerTemplate");
   ```

4. Start the process with an input message.

   Specify the input message as the body, the payload, of the message. In the example, Customer is a message type known to the system.

   ```
   //Create Customer input message
   Customer input = new Customer();
   input.setLastName("Smith");

   message.setObject(new ClientObjectWrapper(input));

   //Send message
   queueSender.send(message);
   ```

   This creates an instance of the process template, CustomerTemplate, and passes some customer data. The operation returns only when the flow is complete and when a JMSReplyToQueue is specified. The result of the process, OrderNo, is returned as the payload of the reply message. Because an ObjectMessage was passed, an object message is returned.

5. **(Optional)** Get the result of the process.

   You can get the results of the process only if you specified a queue in step 2. In the example, OrderNo is a message type known to the system.

```
Message m = queueReceiver.receive();
if (m instanceof ObjectMessage)
{
 ClientObjectWrapper wrapper = (ClientObjectWrapper)m.getObject();
 OrderNo output = (OrderNo)wrapper.getObject();
}
```

Characteristics of non-interruptible business processes

A non-interruptible business process has the following characteristics:

- Runs as one transaction.
- Consists of only synchronous services, EJBs, Java snippets, empty activities, and non-interruptible subprocesses.
- Usually started using the **call** method so that an output message is returned when the process is complete.
- Normally short running.
- Is not visible during execution because run-time values are not stored in the database.

Developing applications for interruptible processes

You can use the EJB interface to develop the following services for interruptible processes:

- Start an interruptible process
- Process a person activity
- Send an event to a process instance
- Analyze the result of a process
- Use work lists to query information

You can use the JMS interface to develop the following services for interruptible processes:

- Start an interruptible process
- Send an event to a process instance
- Analyze the result of a process

For more information, see the Javadoc.

Starting an interruptible process using the EJB interface

Steps for this task

1. **(Optional)** List the process templates to find the name of the interruptible process you want to start.

   This step is optional if you already know the name of the process.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.CAN_RUN_INTERRUP=TRUE"
    "PROCESS_TEMPLATE.NAME",
    newInteger(50),
    null);
   ```

   The results are sorted by name. The query returns an array containing the first 50 sorted templates that can be started as interruptible processes.
2. Start the process with an input message of the appropriate type.

Note: If you specify a process-instance name, it must not start with an underscore. If a process-instance name is not specified, the PIID in String format is used as the name.

```
Customer input = new Customer("Smith");
PIID piid = process.initiate("CustomerTemplate", "CustomerOrder", new ClientObjectWrapper(input));
```

This creates an instance,CustomerOrder, of the process template, CustomerTemplate, and passes some customer data. When the process starts, the operation returns the object ID of the new process instance to the caller.

The starter of the process instance is set to the caller of the request and receives a work item for the process instance. The process administrators, readers, and editors of the process instance are determined and receive work items for the process instance. The starting activity instances are determined and either started automatically or, if they are person activities or receive events, work items are created for the potential owners.

Processing person activities using the EJB interface

Steps for this task

1. List the activities belonging to a logged-on person that are ready to be worked on:

   ```
   QueryResultSet result = process.query("ACTIVITY.AIID",
                           "ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY
                           AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                           null,
                           null,
                           null);
   ```

   This returns a query result set that contains the activities that can be started by the logged-on person.
2. Claim the activity to be worked on:

   ```
   if (result.size() > 0)
   {
    result.first();
    AIID aiid = (AIID) result.getOID(1);
    {
     ClientObjectWrapper input = process.claim(aiid);
     Order activityInput = (Order) input.getObject();
    }
   }
   ```

   When the activity is claimed, the input message of the activity is returned.
3. When work on the activity is complete, complete the activity.

   ```
   OrderNo output = new OrderNo(4711);
   process.complete (aiid, new ClientObjectWrapper(output));
   ```

Person activities

When a person activity is activated, the process engine creates work items and distributes them to the potential owners of the activity. One of these people claims the associated activity, works with the associated data and components, and completes the activity. The completion of the activity triggers the next transaction and continues the process. You can only use the EJB interface to process person activities.

Sending an event to a process instance using the EJB interface

Steps for this task

1. **(Optional)** List the processes that are waiting for a specific event from the logged-on user.

```
QueryResultSet result = query("DISTINCT EVENT_PIID",
                        "EVENT.NAME = 'OrderEvent'
                         AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                         null,
                         null,
                         null);
```

2. Send an event.

   The caller must be a potential owner of the awaited OrderEvent or an administrator of the process instance, CustomerOrder.

```
if (result.size() > 0)
{
 result.first();
 Order input = new Order("Chocolate");
 process.sendEvent ((PIID)result.getOID(1), "OrderEvent", new ClientObjectWrapper(input));
}
```

   Sends the specified OrderEvent to the waiting process instance and passes some order data.

Event activities

An event is an asynchronous notification that can be sent to a process instance. It is used to synchronize the execution of a process instance with the systems external to it. An event activity waits for the occurrence of an external event, several event activities might be waiting for the same event. All these activities receive the external event when it is sent. The event is consumed; subsequent event activities waiting for the same external event require a new event with the same name to be sent.

Analyzing results of a process using the EJB interface

An interruptible process runs asynchronously. Its output message is not automatically returned when the process completes. The message must be retrieved explicitly.

Note: The results of the process are stored in the database only if the process template from which the process instance was derived does not specify automatic deletion of the output message.

Steps for this task

1. Analyze the results of the process:

```
QueryResultSet result = process.query("PROCESS_INSTANCE.PIID",
                        "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                        PROCESS_INSTANCE.STATE = PROCESS_INSTANCE.STATE.STATE_FINISHED",
                        null,
                        null,
                        null);
if (result.size() > 0)
{
 result.first();
 PIID piid = (PIID) result.getOID(1);
 ClientObjectWrapper output = process.getOutputMessage(piid);
 OrderNo order = (OrderNo) output.getObject();
}
```

Using worklists to query information

A worklist is a query that is persistently stored in the database. It represents a set of items which have the same characteristics. Although worklist definitions are stored persistently, items contained in the worklist are assembled dynamically when they are queried. All worklists are publicly accessible.

Steps for this task

1. **(Optional)** List the available worklists:

   ```
   String[] worklists = getWorklistNames();
   ```

2. **(Optional)** Check the query defined by a specific worklist:

   ```
   WorkListData worklist = process.getWorklist("CustomerOrdersStartingWithA");
   String selectClause = worklist.getSelectClause();
   String whereClause = worklist.getWhereClause();
   String orderByClause = worklist.getOrderByClause();
   Integer threshold = worklist.getThreshold();
   ```

3. Run the query defined by the worklist:

   QueryResultSet result =
   process.executeWorklist(″CustomerOrdersStartingWithA″);

Starting an interruptible process using the JMS interface

Steps for this task

1. Create a message, for example, an ObjectMessage.

   ```
   ObjectMessage message= queueSession.createObjectMessage();
   ```

2. **(Optional)** Set the JMSReplyToQueue.

   If you do not want to receive a reply, this step is optional.

   ```
   // Specify the destination object replies are to be sent to
   message.SetJMSReplyTo(replyToQueue);
   ```

3. Set the JMS properties.

   If you do not specify any properties, the process-template name **Dispatch** is assumed. If JMSReplyToQueue is set, **call** is issued. If JMSReplyToQueue is not set, **initiate** is issued. If the process-instance name is set, it must not start with an underscore. If the process-instance name is not set, the PIID in String format is used as the name.

   ```
   message.SetStringProperty("wf$verb", "initiate");
   message.SetStringProperty("wf$processTemplateName",  "CustomerTemplate");
   message.SetStringProperty("wf$processInstanceName", "CustomerOrder");
   ```

4. Start the process with an input message.

   Specify the input message as the body, the payload, of the message. In the following, Customer is a message type known to the system.

   ```
   // Create Customer input message
   Customer input= new Customer();
   input.setLastName("Smith");

   message.setObject(new ClientObjectWrapper(input));

   // Send message
   queueSender.send(message);
   ```

   This creates an instance, CustomerOrder, of the process template, CustomerTemplate, and passes some customer data. The operation returns the object ID of the newly created instance as the value of the JMS property wf$piid if a JMSReplyToQueue is specified. The reply message does not contain a payload.

5. Get the result of the process initiation.

   ```
   Message m = queueReceiver.receive();
   String fiid = m.getStringProperty("wf$piid");
   ```

Sending an event to a process instance using the JMS interface

Steps for this task

1. Create a message, for example, an ObjectMessage.
   ```
   ObjectMessage message= queueSession.createObjectMessage();
   ```
2. **(Optional)** Set the JMSReplyToQueue.

   If you do not want to receive a reply, this step is optional.
   ```
   // Specify the destination object replies are to be sent to
   message.SetJMSReplyTo(replyToQueue);
   ```
3. Set the JMS properties.

   You can specify the PIID of the process instance instead of the process-instance name. If you specify both properties, the processInstanceName is used.
   ```
   message.SetStringProperty("wf$verb", "sendEvent");
   message.SetStringProperty("wf$processInstanceName",  "CustomerOrder");
   message.SetStringProperty("wf$event",  "OrderEvent");
   ```
4. Send the specified OrderEvent to the process instance, CustomerOrder.
   ```
   // Create event input message
   Order input = new Order("Chocolate");

   message.setObject(new ClientObjectWrapper(input));

   // Send message
   queueSender.send(message);
   ```

   If JMSReplyToQueue is set, this returns an empty reply message if the event was sent successfully. The JMSCorrelationID is set to the JMSMessageID of the sendEvent request. Neither properties nor payload are set on the reply message.

Analyzing results of a process using the JMS interface

An interruptible process runs asynchronously. Its output message is not automatically returned when the process completes. The message must be retrieved explicitly. You can get the results of the process only if you specified a JMSReplyToQueue and used the **call** verb to instantiate the process.

Steps for this task

1. Analyze the results of the process:

   In the example, OrderNo is a message type known to the system. When an ObjectMessage is passed in the request, an object message is returned.
   ```
   Message m = queueReceiver.receive();
   if (m instanceof ObjectMessage)
   {
    ClientObjectWrapper wrapper = (ClientObjectWrapper)m.getObject();
    OrderNo output = (OrderNo)wrapper.getObject();
   }
   ```

Characteristics of interruptible processes

An interruptible process has the following characteristics:

- Runs as several transactions.
- Can consist of services, EJBs, Java snippets, event, person, empty, and process activities.
- Usually started using the **initiate** method because the output message cannot be retrieved synchronously.

- Normally long running.
- Is visible during execution because run-time values are stored persistently.

Authorization for EJB renderings

Security must be enabled in WebSphere. When an instance of the BusinessProcess session bean is created, WebSphere associates a session context with the instance. The session context contains the caller's principal. This is used by both the container and the process engine to check the caller's authorization for each call.

The following work-item assignment reasons are used:
- For processes: reader, starter, administrator
- For activities: reader, editor, potential owner, owner

These assignment reasons are mapped to authorization authorities:
- Activity reader authority: allowed to see properties of the associated activity instance, and its input and output messages
- Activity editor authority: allowed everything the activity reader allows, and write access to messages and other data associated with the activity
- Potential activity owner authority: allowed everything the activity editor allows, and the right to claim the activity
- Activity owner authority: allowed everything the potential activity owner allows, and the right to complete the activity
- Process starter authority: allowed to see properties of the associated process instance, and its input and output messages, and write other data associated with the process
- Process reader authority: allowed to see properties of the associated process instance, and its input and output messages, and everything the activity reader allows for all contained activities, including those in blocks but not those of the independent subprocesses
- Process administrator authority: allowed everything the process reader and starter allow, and the right to influence the execution of a process, for example, terminating it

Required authorizations for process requests

Access to the remote BusinessProcess interface does not guarantee that all functions can be executed; the caller must also be authorized to perform the request. The following minimum authorization authorities are needed for process requests:

Request	Required authorization
getProcessInstance	reader
getInputMessage	reader
getOutputMessage	reader
getFaultMessage	reader
getVariableMessage	reader
getCustomAttribute	reader
getUISetting	reader
setCustomAttribute	starter
delete	process administrator
forceTerminate	process administrator

Required authorizations for activity requests

Access to the remote BusinessProcess interface does not guarantee that all functions can be executed; the caller must also be authorized to perform the request. The following minimum authorization authorities are needed for activity requests:

Request	Required authorization
getActivityInstance	activity reader or process reader
getCustomAttribute	activity reader or process reader
getInputMessage	activity reader or process reader
getOutputMessage	activity reader or process reader
getFaultMessage	activity reader or process reader
getUserInput	activity reader or process reader
getUISetting	activity reader or process reader
setCustomAttribute	activity editor or process administrator
setOutputMessage	activity editor or process administrator
setFaultMessage	activity editor or process administrator
setUserInput	activity editor or process administrator
claim	potential activity owner or process administrator
sendEvent	potential activity owner or process administrator
cancelClaim	activity owner or process administrator
complete	activity owner or process administrator
forceRetry	process administrator
forceComplete	process administrator

Developing administration applications for interruptible processes

You can develop the following administration applications for interruptible processes. You can use the EJB interface to render all of these applications. The JMS interface can be used only to terminate a process instance.

- Cancel a claimed activity
- Force the completion of an activity
- Retry the execution of a stopped activity
- Delete a process instance
- Terminate a process instance using the EJB interface
- Terminate a process instance using the JMS interface
- Manage work lists

For more information, see the Javadoc.

Canceling a claimed activity

Sometimes it is necessary for someone with process administrator rights to cancel an activity that has been claimed by someone else. This might happen, for example, when an activity must be completed but the owner of the activity is absent.

Steps for this task

1. List the claimed activities owned by a specific person to find the ID of the activity in question.

```
QueryResultSet result = process.query("DISTINCT ACTIVITY.AIID",
                    "ACTIVITY.STATE = ACTIVITY.STATE.STATE_CLAIMED AND ACTIVITY.OWNER = 'Smith'
                    AND ACTIVITY.TEMPLATE_NAME = 'CustomerTemplate'",
                    null,
                    null,
                    null);
```

This returns a query result set that lists the activities claimed by the specified person, Smith.

2. Cancel the claimed activity.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 process.cancelClaim(aiid);
}
```

This returns the activity to the ready state so that it can be claimed by one of the other potential owners.

Forcing the completion of an activity

If an activity in an interruptible process encounters a system exception or an unconnected fault terminal and the associated activity template specifies that the activity should stop when an error occurs, the activity is put into the stopped state so that it can be repaired. You can force completion of the activity. You can also pass parameters in the force-complete call, such as the message that should have been computed or the fault that should have been raised.

Steps for this task

1. List the stopped activities.

```
QueryResultSet result = process.query("DISTINCT ACTIVITY.AIID",
           "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND PROCESS_INSTANCE.NAME='CustomerOrder'",
           null,
           null,
           null);
```

This returns the stopped activities for the process instance, CustomerOrder.

2. Complete the activity.

In this example an output message is passed.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 OrderNo output = new OrderNo(4711);
 process.forceComplete(aiid, new ClientObjectWrapper(output), true);
}
```

For more information, see the Javadoc.

This completes the activity. If a system error occurs, the **continueOnError** parameter determines whether the activity stays in the stopped state. In the example, **continueOnError=true**. This means that if an error occurs during processing of the forceComplete request, the activity is put into the failed execution state. Navigation continues and the process is put into the failing state.

For more information, see the Javadoc

Retrying the execution of a stopped activity

If an activity in an interruptible process encounters a system exception or an unconnected fault terminal and the associated activity template specifies that the activity should stop when an error occurs, the activity is put into the stopped state so that it can be repaired. You can retry the execution of the activity by passing a new input message.

Steps for this task

1. List the stopped activities.

   ```
   QueryResultSet result = process.query("DISTINCT ACTIVITY.AIID",
                                  "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
   PROCESS_INSTANCE.NAME='CustomerOrder'",
                                   null,
                                   null,
                                   null);
   ```

   Note: The second line of the previous example wrapped to a second line to fit the width of the page.

   This returns the stopped activities for the process instance CustomerOrder.
2. Retry the execution of the activity.

   ```
   if (result.size() > 0)
   {
    result.first();
    AIID aiid = (AIID) result.getOID(1);
    Order input = new Order("Chocolate");
    process.forceRetry(aiid, new ClientObjectWrapper(input), true);
   }
   ```

   For more information, see the Javadoc.

   This retries the activity. If a system error occurs, the **continueOnError** parameter determines whether the activity stays in the stopped state. In the example, **continueOnError=true**. If an error occurs during processing of the forceRetry request, the activity is put into the failed execution state. Navigation continues and the process is put into the failing state.

Deleting a process instance

Processes instances are only automatically deleted when they complete if this is specified in the process template from which the process instances are derived. To delete all finished process instances:

Steps for this task

1. List the process instances that are finished.

   ```
   QueryResultSet result = process.query("DISTINCT PROCESS_INSTANCE.PIID",
                        "PROCESS_INSTANCE.STATE = PROCESS_INSTANCE.STATE.STATE_FINISHED",
                        null,
                        null,
                        null);
   ```

   This returns a query result set that lists finished process instances.
2. Delete the finished process instances.

   ```
   PIID piid;
   while (result.next() )
   {
    piid = (PIID) result.getOID(1);
    process.delete(piid);
   }
   ```

Terminating a process instance using the EJB interface

Sometimes it is necessary for someone with process administrator rights to terminate a process instance that is known to be in an unrecoverable state.

Steps for this task

1. Retrieve the process instance to be terminated.

   ProcessInstanceData processInstance = process.getProcessInstance("CustomerOrder");
2. Terminate the process instance.

   ```
   PIID piid = processInstance.getID();
   process.forceTerminate(piid);
   ```

Terminating a process instance using the JMS interface

Steps for this task

1. Create a message, for example, an ObjectMessage.

   ```
   ObjectMessage message= queueSession.createObjectMessage();
   ```
2. **(Optional)** Set the JMSReplyToQueue.

   If you do not want to receive a reply, this step is optional.

   ```
   // Specify the destination object replies are to be sent to
   message.SetJMSReplyTo(replyToQueue);
   ```
3. Set the JMS properties.

   ```
   message.SetStringProperty("wf$verb", "forceTerminate");
   message.SetStringProperty("wf$processInstanceName",  "CustomerOrder");
   ```
4. Terminate the process instance, CustomerOrder.

   ```
   // Send message
   queueSender.send(message);
   ```

   If JMSReplyToQueue is set, this returns an empty reply message if the process instance was terminated successfully. The JMSCorrelationID is set to the JMSMessageID of the forceTerminate request. Neither properties nor payload are set on the reply message.

Managing worklists

A worklist is a query that is stored persistently in the database. It represents a set of items which have the same characteristics. Although worklist definitions are stored persistently, items contained in the worklist are assembled dynamically when they are queried. All worklists are publicly accessible.

Steps for this task

1. Create a worklist

   To create a worklist, save a query with a specific name:

   ```
   process.newWorklist("CustomerOrdersStartingWithA",
               "PROCESS_INSTANCE.NAME, DISTINCT PROCESS_INSTANCE.PIID",
               "PROCESS_INSTANCE.NAME LIKE 'A%'",
               "PROCESS_INSTANCE.NAME",
               null,
               null);
   ```

   This query returns a sorted list of all the process-instance names that begin with the letter A with their associated PIIDs.
2. Delete a worklist.

```
process.deleteWorklist("CustomerOrdersStartingWithA");
```

Accessing the Process Choreographer EJB interface

An application accesses the BusinessProcess session bean through the bean's home and remote interfaces.

Steps for this task

1. Add a reference to the BusinessProcess session bean to your application deployment descriptor.

   Add the reference to:

   - application-client.xml, for a J2EE client application
   - web.xml, for a Web application
   - ejb-jar.xml, for an EJB application

   Add the reference as follows:

   ```
   <ejb-ref>
    <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.bpe.api.BusinessProcess</home>
    <remote>com.ibm.bpe.api.BusinessProcess</remote>
   </ejb-ref>
   ```

2. Make the BusinessProcess session bean's home interface available to the application.

   You can do this using JNDI-lookup mechanisms:

   ```
   // Obtain the default initial JNDI context
   Context initialContext = new InitialContext();

     // Lookup the home interface of the BusinessProcess bean
     Object result = initialContext.lookup("java:comp/env/ejb/BusinessProcessHome");

   // Convert the lookup result to the proper type
      BusinessProcessHome processHome = (BusinessProcessHome)javax.rmi.PortableRemoteObject.
   narrow(result,BusinessProcessHome.class);
   ```

   Note: The eigth line of the previous example, beginning with ″BusinessProcessHome″ wrapped to a second line to fit the width of the page.

   The BusinessProcess session bean's home interface contains a create method for EJB objects. The method returns the session bean's remote interface.

3. Access the BusinessProcess session bean's remote interface:

   ```
   Process process = processHome.create();
   ```

4. Call the business functions exposed by the BusinessProcessService interface, for example:

   ```
   process.initiate("MyProcessModel",input);
   ```

   Calls from applications are executed as transactions. A transaction is established and ended in one of the following ways:

   - Automatically by WebSphere (the deployment descriptor specifies TX_REQUIRED).
   - Explicitly by the application. You can bundle application calls into one transaction:

     ```
     // Obtain user transaction interface
        UserTransaction transaction= (UserTransaction)initialContext.lookup("jta/usertransaction");

        // Begin a transaction
     ```

```
transaction.begin();

  // Applications calls ...

// On successful return, commit the transaction
transaction.commit();
```

Accessing the Process Choreographer JMS interface

Process Choreographer accepts JMS messages that follow the point-to-point paradigm. An application that sends or receives JMS messages must:

Steps for this task

1. Create a connection to Process Choreographer.

   Use JNDI lookup to retrieve the connection factory. The JNDI-lookup name must be the same as the name specified when the Process Choreographer external request queue is configured.

   ```
   //Obtain the default initial JNDI context
   Context initialContext = new InitialContext();

   // Look up the connection factory
   QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
   initialContext.lookup("jms/BPECF");

   // Create connection
   QueueConnection queueConnection = queueConnectionFactory.createQueueConnection();
   ```

   Note: The fifth line of the previous example wrapped to a second line to fit the width of the page.
2. Create a session so that message producers and consumers can be created.

   ```
   //Create a nontransacted session using autoacknowledgement
   QueueSession queueSession = queueConnection.createQueueSession( false, Session.AUTO_ACKNOWLEDGE );
   ```
3. Create a message producer to send messages.

   The JNDI-lookup name must be the same as the name specified when the Process Choreographer external request queue is configured.

   ```
   // Look up the destination of the Process Choreographer queue to send messages to
   Queue bpeQueue = (Queue) initialContext.lookup("jms/BPEApiQueue");
   // Create message producer
   QueueSender queueSender = queueSession.createSender(bpeQueue);
   ```
4. Create a message consumer to receive replies.

   The JNDI-lookup name must specify a user-defined destination to send replies to.

   ```
   // Look up the destination of the reply to queue
   Queue replyToQueue = (Queue) initialContext.lookup("MyReplyToQueue");
   // Create message consumer
   QueueReceiver queueReceiver = queueSession.createReceiver(replyToQueue);
   ```
5. Send requests and receive replies.

   ```
   // Start sending and receiving messages
   queueConnection.start();

   // Create message - see the task descriptions for examples - and send
   ObjectMessage message = queueSession.createObjectMessage();
   // message.SetStringProperty(..);
   // message.setObject(...);

   queueSender.send(message);

   // Receive message and analyze reply - see the detailed task descriptions for examples
   Message reply = queueReceiver.receive();
   ```
6. Close the connection to the free resources.

   ```
   // Final housekeeping: free resources
   queueConnection.close();
   ```

Structure of a Process Choreographer JMS message

A JMS message consists of:

- A message header for message identification and routing information.
- Properties for JMS-specific data, application-specific data, and provider-specific data (optional).
- The body (payload) of the message that holds the content (optional).

Process Choreographer supports text, object, and bytes-message formats.

Message header

The following fields can be set by a Process Choreographer client application:

- JMSReplyTo

  The destination where a reply to the request should be sent. If this is not specified, a reply is not returned.
- JMSMessageID

  Uniquely identifies a message. This is set by Process Choreographer when the message sent returns. This is used as the JMSCorrelationID in the reply message.
- JMSCorrelationID

  Links messages. Do not set this field. A Process Choreographer reply message contains the JMSMessageID of the request message.

Data properties

This data is passed as name-value pairs. Process Choreographer adds the following properties:

- wf$verb

  The function to be called. Possible values are: initiate, call, forceTerminate, sendEvent.

  If wf$verb is not set and JMSReplyTo is set, then **call** is issued. If neither wf$verb nor JMSReplyTo are set, **initiate** is issued.
- wf$processTemplateName

  The name of the process template to be instantiated. This must be set for **call** and **initiate**. If the template name is not set, **Dispatch** is assumed.
- wf$piid

  The object ID of the process instance. It can be used with **forceTerminate** and **sendEvent** to identify the process instance. It is set as the result of **initiate** if JMSReplyTo is set. If both wf$piid and wf$processInstanceName are set, the value for wf$processInstanceName is used.
- wf$processInstanceName

  The name of the process instance. It can be used with **forceTerminate** and **sendEvent** to identify the process instance to be processed. It can be used with **call** and **initiate** to specify the process instance name of the process instance to be created.
- wf$eventName

  The name of the event to be sent. This must be set for **sendEvent**.
- wf$processState

  The state of the process instance. This is set in reply messages.
- wf$exceptionText

Specifies the message text of the exception that ended the process.

Message payload

Used to specify input, output, and fault messages.

- Request messages
  - TextMessage must contain an object of the type String. This object contains the input message to be passed. It requires that the defined input message is also of type String.
  - ObjectMessage must contain an object of the type ClientObjectWrapper. This object contains the input message to be passed.
  - BytesMessage must contain a byte array that represents the streamed version of the input message to be passed.
- Reply messages
  - If possible, the reply message is the same format as the request message. That is, if the request message is a TextMessage, then the reply message is also a TextMessage. If a TextMessage cannot be used, for example, because the type of the process instance output message is not a String, then an ObjectMessage is returned.
  - If an ObjectMessage is returned, it contains an object of the type ClientObjectWrapper. This object contains the output message or the message of the fault terminal.
  - If a BytesMessage is returned, it contains the streamed output message or the streamed message of the fault terminal.
  - If a process can be navigated until its output or fault terminal is reached, the properties wf$processInstanceName and wf$processState are set.
  - If an exception occurs during the processing of a request or if an exception occurs during the execution of a process instance and the fault is not connected to a fault terminal of the process, the reply message does not contain a payload. The properties wf$processInstanceName, wf$processState, and wf$exceptionText are set. wf$exceptionText contains the message text of the exception that ended the process. Nested exceptions follow the message text. These are separated by new-line characters.

Authorization for JMS renderings

Security must be enabled in WebSphere. When an instance of the BusinessProcess MDB is deployed, the role ’MDBUser’ must be mapped to a specific user ID. This user ID is used by both the business process container and the process engine to check the caller’s authorization for each request.

The following authorization authorities are needed:

Request	Required authorization
forceTerminate	process administrator
sendEvent	potential activity owner or process administrator

Configuring the staff service for Process Choreographer

Process Choreographer uses staff plug-ins to determine who can start a process or claim an activity. Your business processes can also use the staff plug-in services to resolve staff queries. Each type of directory service requires a different staff plug-in. You can register multiple staff plug-ins. The User Registry and System plugins are already installed and can be used without any configuration. To configure a staff plug-in provider:

Steps for this task

1. In the Administrative Console, click **Resources > Staff Plugin Provider**.

   The System plug-in and the User Registry plug-in require no customization and are ready to use. The preconfigured LDAP staff plugin configuration assumes that the LDAP server is on the same host as the Enterprise Application Server.
2. To create a new LDAP configuration:
   a. Click on the name of the LDAP staff plug-in provider.
   b. Select **Staff Plugin Configuration**.
   c. Click **New**.
   d. Click **Browse**, and select the XSL transformation file to be used.

      The standard XSL transformation for LDAP is located on UNIX in `$WAS_HOME/ProcessChoreographer/Staff/LDAPTransformation.xsl` and on Windows, in `%WAS_HOME%\ProcessChoreographer\Staff\LDAPTransformation.xsl`.

      Note: Do not modify this transformation file. If you need to customize the transformations to match your organization's XML schema, modify a copy that has a different file name.
   e. Click **Next**.
   f. Enter an administrative name for the staff plug-in provider.
   g. Enter a description.
   h. Enter the JNDI name that will be used by business processes to reference this plug-in, for example, `bpe/staff/ldapserver1`
   i. Click **Apply**.
   j. Click **Custom Properties**.
   k. For each of the required properties and for any optional properties that you want to set, click on the property's name, enter a value, and click **OK**.

      This table describes each property for the LDAP plug-in.

LDAP plug-in property	Required or optional	Comments
AuthenticationAlias	Optional	The authentication alias used to connect to LDAP, for example, `mycomputer/My LDAP Alias`. You must have defined this alias in the Administrative console. If this is not set, anonymous logon to the LDAP server is used.
AuthenticationType	Optional	If AuthenticationType is not set, the default logon is anonymous authentication. In all other cases, the default is simple authentication.

LDAP plug-in property	Required or optional	Comments
BaseDN	Required	The base distinguished name for all LDAP search operations, for example, `"o=mycompany, c=us"`
ContextFactory	Required	Sets the JNDI context factory, for example, `"com.sun.jndi.ldap.LdapCtxFactory"`
ProviderURL	Required	The LDAP directory server and port, for example, `"ldap://localhost:389"`
SearchScope	Required	The default search scope for all queries. Determines how deep to search beneath the baseDN. It must be one of: `"objectScope"`, `"onelevelScope"`, or `"subtreeScope"`

l. To apply the changes you have made, click **Save** in the Message(s) box.

3. To activate the plug-in, stop and start the server.

Results

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

What to do next

Depending on the queries you want to create and your directory structure, you might need to create your own transformations. For more information about this see Using the staff service in Process Choreographer

Using the staff service in Process Choreographer

Process Choreographer allows you to separate your business process logic from the staff resolution. Staff queries are resolved using a plug-in that is specific for the directory service. Some examples of using the staff service are descibed here. For detailed information on the staff resolution plug-ins, see the staff resolution white paper in the WebSphere Application Server Library at http://www.ibm.com/software/webservers/appserv/library.

Staff queries using the staff support service

The WebSphere Studio Application Developer Integration Edition allows you to define staff queries for the Staff Support Service. These queries are constructed using a set of predefined verbs and parameters.

The following example illustrates a Staff Support Service FDML snippet to retrieve the manager of the employee that started the process:

```
<wf:staff type="staffSupportService">
  <staff:verb>
    <staff:id>Manager of Employee by user ID</staff:id>
    <staff:name>Manager of Employee by user ID</staff:name>
    <staff:parameter id="EmployeeUserID">
```

```
      %wf:process.starter%
    </staff:parameter>
  </staff:verb>
</wf:staff>
```

When a process template is installed in the business process container, it is associated with the JNDI name of one staff plug-in, for example, `bpe/staff/ldapserver1`. Therefore, each process is associated with a particular staff plug-in, and any staff resolution queries from that process are translated into a form that is specific to the plug-in.

Staff query verb set

The staff support service accepts queries in an abstract form that is independent of the directory infrastructure being used. Not all plug-ins support all verbs. The following predefined set of verbs is available:

Users (parameter: Name, AlternativeName1, AlternativeName2)
: This verb allows you to define a staff query for a user who is known by name. It is not recommended that you hard code user names in process templates. This verb is useful for testing processes.

Users by user ID (parameter: User ID, AlternativeID1, AlternativeID2)
: This verb allows you to define a staff query for a user whose user ID is known. Even though it is not recommended that you hard code user IDs in process templates, this verb is useful in combination with context queries, for example:

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

Group Members (parameter: GroupName, IncludeSubgroups, Domain, AlternativeGroupName1, AlternativeGroupName2)
: This verb allows you to define a query to retrieve the members of a group. The `IncludeSubgroups` parameter defines whether group members are evaluated recursively. The `Domain` parameter allows to limit the search to a subset of the directory. In order to assign members of several groups in one query, the `AlternativeGroupName*` attributes can be used. This verb can be mapped to all specialized plug-ins except the System plug-in. However, not all plug-ins support the `IncludeSubgroups` and `Domain` parameters.

Department Members (parameters: DepartmentName, IncludeSubgroups, Domain, AlternativeDepartmentName1, AlternativeDepartmentName2)
: This verb allows you to define a query to retrieve the members of a department. The parameters are similar to the ones provided for the verb `Group Members`. This verb can be mapped using the LDAP plug-in, but the transformation requires knowledge about the schema that is used.

Role Members (parameters: RoleName, include nested roles, Domain, AlternativeRoleName1, AlternativeRoleName2)
: Retrieves the users associated with a business process role. The parameters are similar to the ones provided for the verb `Group Members`. This verb can be mapped using the LDAP plug-in, but the transformation must be customized to match your organization's schema.

Manager of Employee (parameters: EmployeeName, Domain)
: Retrieves the manager of a person. It can be mapped using the LDAP plug-in, but the transformation must be customized to match your organization's schema.

Manager of Employee by user ID (parameters: EmployeeUserID, Domain)
: Retrieves the manager of a person using the person's user ID. It can be

mapped using the LDAP plug-in, but the transformation must be customized to match your organization's schema. This verb allows you to use context variables, such as `%wf:process.starter%`.

Person Search (parameters: ID, Profile, Last Name, First Name, Middle Name, email, Company, Display Name, Secretary, Assistant, Manager, Department, Employee ID, Tax Payer ID, phone, fax, Gender, Timezone, Preferred Language)
Searches for a person based on an attribute match. It can be mapped using the LDAP plug-in, but the transformation requires knowledge about the XML schema that is used.

Group Search (parameters: ID, DisplayName, Type, Industry Type, Business Type, Geographic Location, Affiliates, Secretary, Assistant, Manager, BusinessCategory, Parent Company)
This verb allows you to search for a group based on an attribute match and to retrieve its users. It can be mapped using the LDAP plug-in, but the transformation requires knowledge about your organization's XML schema.

Native Search (parameters: Search Query, Query Template, and 5 optional additional parameters (AdditionalParameter1-5))
This verb allows you to define a native search based on directory specific parameters. It is not recommended to use this verb, as it creates directory dependencies in the process template. However, it may be required for specific process scenarios. It can be mapped using the UserRegistry plug-in, or the LDAP plug-in, but for LDAP, the transformation requires knowledge about your organization's LDAP schema.

Everybody
This verb allows you to assign a work item to every authenticated user. This verb has no parameters, and is supported by all staff plug-ins.

Nobody
This verb denies normal users access to the work item - only the process administrator and the Process Choreographer system administrator have access to it. This verb has no parameters, and is supported by all staff plug-ins.

Staff queries transformed for the User Registry staff plug-in

This plug-in allows staff queries to refer to users and groups that are known to the WebSphere Application Server. This plug-in requires no configuration parameters, and comes pre-installed and ready-to-use.

The following example FDML snippet resolves to the user IDs of all members of the `Administrators` group, and of all users whose name begins with 'Mi':

```
<wf:staff type="UserRegistry">
  <sur:staffQueries>
    <sur:usersOfGroup groupName="Administrators"/>
    <sur:search type="user" name="Mi*"/>
  </sur:staffQueries>
</wf:staff>
```

Note: This FDML is an example of the output of the transformation. You only need to understand this example if you modify the XSLT for the plug-in.

Explicit staff assignments using the System staff plug-in

This plug-in requires no configuration parameters, and comes pre-installed and ready-to-use. It allows you to hard code user names in your business process. This is not normally recommended, but it can be useful for testing, when using context variables, or special queries, as illustrated by the following four examples:

- `<staff:userID name="%wf:process.starter%">`
- `<staff:userID name="%wf:process.administrators%">`
- `<staff:everybody>`
- `<staff:nobody>`

The following FDML snippet explicitly resolves to the WebSphere user ID `smith`:

```
<wf:staff type="System">
  <staff:staffQueries>
    <staff:userID name="smith">
  </staff:staffQueries>
</wf:staff>
```

Modifying the transformations

If you want to use the LDAP plug-in, you will probably have to create a customized version of the LDAP XSL transformation to match your organization's LDAP schema. The standard XSLT provided for LDAP is located on UNIX in `$WAS_HOME/ProcessChoreographer/Staff/LDAPTransformation.xsl` and on Windows, in `%WAS_HOME%\ProcessChoreographer\Staff\LDAPTransformation.xsl`. Make your changes to a copy that has a different file name.

Staff queries transformed for the LDAP staff plug-in

The following FDML snippet illustrates the results of the LDAP transformation for a search. You only need to understand this example if you modify the XSLT for the LDAP plug-in.

```
<wf:staff type="LDAP">
  <sldap:search baseDN="search root"
              filter="valid LDAP filter"
              searchScope="subtreeScope"|"onelevelScope"|"objectScope"
              recursive="yes"|"no">
  attribute evaluation specification*
  </sldap:search>
</wf:staff>
```

Where:

baseDN
: is optional. It specifies the LDAP subtree to be searched. The default is the value of BaseDN specified when configuring the LDAP staff plug-in.

filter is mandatory.

attribute evaluation specification
: defines how to retrieve the query results. It can consist of one or more attribute elements:

```
<sldap:attribute name="attribute name"
                 objectclass="LDAP object class"
                 usage="simple"|"recursive"|/>
```

Where the three mandatory attributes:

name the attribute to evaluate, for example, `name = "sn"`.

objectclass
: the LDAP object class for this attribute, for example, `objectclass = "person"`.

usage determines how the attribute contents are used. Simple, adds the contents to the (intermediate) query result set. Recursive, uses the contents as DNs for recursive retrieval and evaluation.

Staff service settings

Use this page to enable or disable the staff service, which manages staff plugin resources used by the server.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Staff Service** .

Startup

Specifies whether the server will attempt to start the staff service.

Default	**Selected**
Range	**Selected** When the application server starts, it attempts to start the staff service automatically. **Cleared** The server does not try to start the staff service. If staff plugin resources are to be used on this server, the system administrator must start the staff service manually or select this startup property then restart the server.

Staff plugin provider collection

Use this page to manage staff plugin providers.

A staff plugin is responsible for the retrieval of user information.

To view this administrative console page, click **Resources > Staff Plugin Providers**.

Name

The name by which the staff plugin provider is known for administrative purposes.

Data type
String

Description

A description of the staff plugin provider.

Data type
String

Jar File

The file name, including the absolute path, of the Jar file containing the plugin.

Data type
String

Staff plugin provider settings

Use this page to modify staff plugin provider settings.

Staff Plugins are responsible for the retrieval of user information. Each staff plugin provider is registered with the run-time environment by specifying a name and a Jar file containing the plugin. A configuration file in the Jar file defines the class name, which represents the plugin as well as the properties for the plugin.

To view this administrative console page, click **Resources > Staff Plugin Providers** > *staffpluginprovider_name*.

Name

The name by which the staff plugin provider is known for administrative purposes.

Data type
String

Description

A description of the staff plugin provider.

Data type
String

Jar File

The file name, including the absolute path, of the Jar file containing the plugin.

Data type
String

Staff plugin configuration collection

Use this page to manage staff plugin configurations.

A staff plugin configuration is defined for a staff plugin provider. The staff plugin configuration can define any custom properties specified by the staff plugin provider. Each staff plugin provider can have multiple staff plugin configurations.

To view this administrative console page, click **Resources** > *Staff Plugin Providers* > **staffpluginprovider_name > Staff Plugin Configuration**.

Configuration tab

Name The name by which the staff plugin configuration is known for administrative purposes.

Data type
String

Description
A description of the staff plugin configuration.

Data type
String

JNDI Name
The JNDI name used to look up the staff plugin configuration in the namespace.

Data type
String

XSL Transform File
The file name, including absolute path, of the XSL transformation file.

Data type
String

Staff plugin configuration settings

Use this page to modify staff plugin configuration settings.

To view this administrative console page, click **Resources > Staff Plugin Providers >** *staffpluginprovider_name* **> Staff Plugin Configuration >** *staffpluginconfiguration_name*.

Configuration tab

Name The name by which the staff plugin configuration is known for administrative purposes.

Data type
String

Description
A description of the staff plugin configuration.

Data type
String

JNDI Name
The JNDI name used to look up the staff plugin configuration in the namespace.

Data type
String

XSL Transform File
The file name, including absolute path, of the XSL transformation file.

Data type
String

Using compensation in service choreography

You can use *compensation* in business processes to enable updates to be committed in several related transactions before the process has completed. If the process does not complete successfully, compensation is used to automatically perform actions that *compensate for* updates that have been committed. The compensation actions can be to back out committed updates or to take alternative actions.

For compensation, application developers can create a type of component service called a compensation pair. Each compensation pair contains a primary operation that performs the normal actions of the service and a compensation operation that can be used for the compensation actions.

You can use WebSphere Studio to define compensation properties of interruptible business processes.

Note: You cannot use compensation in non-interruptible processes.

When a process is started, it starts a *new compensation* sphere. Each compensation pair called as a node of the process registers with the compensation sphere, runs its primary operation, and stores the information needed to run its compensation operation if the process needs to be compensated. When the process ends it closes the compensation sphere, which checks if there is a need for compensation. The process decides whether to accept all of the updates made by primary operations or to run the compensation operation of each compensation pair. The compensation operation can access parts of both the input message and the primary operation's output message.

Troubleshooting Process Choreographer

Process Choreographer uses the WebSphere framework JRas for traces, messages, and audit logs. The following topics contain troubleshooting information that applies for Process Choreographer only.

Steps for this task

1. Using process-related trace information
2. Using process-related audit trail information
3. Using process-related messages
4. Troubleshooting the business process container

Using process-related trace information

Before you begin

To use tracing for process-related items, it must have been specified in the Flow Definition Markup Language (FDML) during modeling of the process. Accordingly, the options for tracing must also be set in the FDML.

Process Choreographer uses the WebSphere framework JRas for traces. How to use traces within WebSphere is described in Working with trace.

The following is process-specific information that is relevant for Process Choreographer:

- The Java package for Process Choreographer is **com.ibm.bpe**. Use com.ibm.bpe to trace process-related events only.
- The trace information is stored in the trace file that corresponds to the Java package you are using.

Using process-related audit trail information

Before you begin

To use the audit trail for Process Choreographer, it must have been enabled during the modeling of a process. It is specified in the FDML (Flow Definition Modeling Language) whether audit information is written at all, and which events are reported in the audit trail.

When a process instance is executed and audit trailing is enabled, Process Choreographer writes information about each significant event into an audit log, which is located in the corresponding database table. Process Choreographer provides a plug-in for the audit trail database to be used. In addition, you can clean up the audit log table according to your needs by using the cleanup utility.

- Using the BPEAuditLogDelete utility describes how to start the utility and lists the options that you can use.
- Process Choreographer - structure of the audit trail database table describes the structure of the audit trail database table.
- Process Choreographer - audit event types contains a list and description of all available event type codes that can occur during the processing of processes.

BPEAuditLogDelete utility for Process Choreographer

Syntax

You can use this utility to delete audit log entries for Process Choreographer from the database.

To run the utility, change to the `\bin` subdirectory of the WebSphere application server, and enter the command, using the following syntax:

```
BPEAuditLogDelete {-processtime processtime| -time time| -all [slice]}
 -dbs database_system -dbn database_name
 [-u userId] [-p password]
 [-host host] [-port port]
```

Parameters

Supported arguments include:

-processtime
: Deletes all audit log entries that belong to a process that has finished before the time specified in *processtime*. Use the same time format as for the -time option.

-time
: Deletes all audit log entries that are older than the time you specify in *time*. The time format must be as follows: `YYYY-MM-DD['T'HH:MM:SS]`. If you only specify year, month, and day, the hour, minutes, and seconds are set to `00:00:00`.

-all
: Deletes all audit log entries in the database. This is done in multiple transactions where each transaction deletes the specified amount of entries as specified in *slice*. The default size for *slice* is 250.

-dbs
: Specifies the database system. Allowed database systems and keywords are DB2, Cloudscape, Oracle, and Sybase.

-dbn
: Specifies the name of the database that contains the audit log table.

-u
: Specifies the user that is used to connect to the database. This option is not required for Cloudscape and DB2.

-p
: Specifies the password for the user you specified with the -u option. You can also enter the password after starting the utility.

-host
: Specifies the name of the database host. This option is only required for Oracle and Sybase, it is not supported for DB2 and Cloudscape.

-port
: Specifies the port of the database listener on the host. This option is only required for Oracle and Sybase, it is not supported for DB2 and Cloudscape.

Process Choreographer - structure of the audit trail database table

The following table describes the structure of the audit trail database table `AUDIT_LOG_T`. It lists the names of the columns, the audit event types, and gives you a short description for the table column.

To read the content of the audit trail, use SQL or any other administration tool that allows you to read database tables.

There are the following audit event types:
- Process instance events (PIE)
- Activity instance events (AIE)
- Events related to variables (VAR)
- Control link events (CLE)
- Process template events (PTE)

For a list of the possible audit event type codes refer to Process Choreographer - audit event types.

Structure of the Process Choreographer audit trail database table AUDIT_LOG_T

Name	Audit event type	Description
`ALID`	all	Internal ID and primary key for a row.
`EVENT_TIME`	all	Timestamp of when the event occurred (in UTC format).
`AUDIT_EVENT`	all	For a list of audit event codes refer to Using process-related audit trail information
`PTID`	all	Process template ID of the process that is related to the current event.
`PIID`	PIE, AIE, VAR, CLE	Process Instance ID of the process instance that is related to the current event.
`PROCESS_TEMPL_NAME`	PIE, AIE, VAR, CLE	Process template name of the process template that is related to the current event.
`PROCESS_INST_NAME`	PIE, AIE, VAR, CLE	Process instance name of the process template that is related to the current event.
`TOP_LEVEL_FI_NAME`	PIE, AIE, VAR, CLE	Name of the top-level process that is related to the current event.
`TOP_LEVEL_PIID`	PIE, AIE, VAR, CLE	Identifier of the top-level process that is related to the current event.
`PARENT_PI_NAME`	PIE, AIE, VAR, CLE	Name of the parent process instance.
`PARENT_PIID`	PIE, AIE, VAR, CLE	Process instance ID of the parent process, or null if no parent exists.
`VALID_FROM`	PIE, AIE, VAR, CLE	Valid-from date of the process template that is related to the current event.
`ACTIVITY NAME`	AIE	Name of the activity on which the event occurred.

Name	Audit event type	Description
ACTIVITY_KIND	AIE	Kind of the activity on which the activity occurred. It can have the following values: KIND_PROCESS_SUBPROCESS KIND_PROCESS_BLOCK KIND_EMPTY KIND_ELEMENTAL KIND_PERSON KIND_EVENT
ACTIVITY_STATE	AIE	State of the activity that is related to the event.
CONTROL_CONNECTOR_NAME	CLE	Name of the control link that is related to the current control link event.
IMPL_NAME	AIE	Name of the activity implementation. This is only applicable for elemental activities.
PRINCIPAL	AIE, PIE	Name of the principal that requested the API-call related to the event, applicable for these events (but only if directly called over the API): ACTIVITY_CLAIMED ACTIVITY_COMPLETED ACTIVITY_FAILED ACTIVITY_FAULT_MESSAGE_SET ACTIVITY_OUTPUT_MESSAGE_SET ACTIVITY_USERINPUT_SET PROCESS_DELETED PROCESS_STARTED PROCESS_STARTED PROCESS_TERMINATED
TERMINAL_NAME	AIE	Name of the fault terminal that has been set, applicable for SET_FAULT_MESSAGE.
VARIABLE_DATA	VAR	Data for variables for VARIABLE_UPDATED events.
EXCEPTION_TEXT	PIE, AIE	Exception message that caused an activity or process to fail. Applicable for: - PROCESS_FAILED - ACTIVITY_FAILED.

Process Choreographer - audit event types

The following tables list the types of audit events that can occur while processes are running, together with their corresponding codes. The ones in brackets are not implemented in this version.

Process instance events (PIE)

PROCESS_STARTED	21000
(PROCESS_SUSPENDED)	21001

(PROCESS_RESUMED)	21002
PROCESS_COMPLETED	21004
PROCESS_TERMINATED	21005
PROCESS_DELETED	21020
PROCESS_FAILED	42001
PROCESS_COMPENSATING	42003
PROCESS_COMPENSATED	42004
PROCESS_TERMINATING	42009
PROCESS_FAILING	42010

Activity events (AIE)

ACTIVITY_READY	21006
ACTIVITY_STARTED	21007
ACTIVITY_COMPLETED	21011
ACTIVITY_CLAIM_CANCELED	21021
ACTIVITY_CLAIMED	21022
ACTIVITY_TERMINATED	21027
ACTIVITY_RESTARTED_EXIT_CONDITION_FALSE	21041
ACTIVITY_FAILED	21080
ACTIVITY_EXPIRED	21081
ACTIVITY_LOOPED	42002
(ACTIVITY_SKIPPED)	42005
ACTIVITY_TERMINATING	42008
ACTIVITY_FAILING	42011
ACTIVITY_OUTPUT_MESSAGE_SET	42012
ACTIVITY_FAULT_MESSAGE_SET	42013
ACTIVITY_USERINPUT_SET	42014

Events related to variables (VAR)

VARIABLE_UPDATED	21090

Control link events (CLE)

CONTROL_LINK_EVALUATED_TO_TRUE	21034

Process template events (PTE)

PROCESS_INSTALLED	42006
PROCESS_UNINSTALLED	42007

Using process-related messages

Process Choreographer uses the WebSphere framework JRas for handling messages. To view Process Choreographer messages, check the activity.log file by using the WebSphere Log Analyzer.

The following is process-specific information that is relevant for Process Choreographer:

- The message catalog is **com.ibm.bpe.util**.
- The messages are stored in the com/ibm/BPE/Messages_<*language*><*country*>.properties file.
- The format for Process Choreographer messages is ″**BPE<Component><Number><TypeCode>**″. All messages that belong to Process Choreographer are prefixed with ″**BPE**″.

Process Choreographer: Resources for learning

Use the following links to find relevant supplemental information about Process Choreographer. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the Process Choreographer, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

View links to additional information about:

- “Planning, business scenarios, and IT architecture”
- “Programming instructions and examples”

Planning, business scenarios, and IT architecture

- White paper Process Choreographer Concepts and Architecture, by Matthias Kloppman and Gerhard Pfau, published in the WebSphere Developer Domain library , 2002.
- *Web services and business process management*, by F. Leymann, D. Roller, M.-T. Schmidt, IBM Systems Journal, Vol. 42, No. 2, 2002.

Programming instructions and examples

- Process Choreographer sample delivered in the Samples Gallery. Follow the instructions for accessing the samples on your local machine.

See also the set of links given in Enterprise beans: Resources for learning, which is located in the InfoCenter

Notices

programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

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

Trademarks

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

AIX
CICS
DB2
IBM
MQSeries
WebSphere

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

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

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

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