OpenText™ Documentum™ Content Management

Advanced Workflow User Guide

Design, create, and configure workflow processes.

EDCPKLPR250400-UGD-EN-1

OpenText™ Documentum™ Content Management
Advanced Workflow User Guide
EDCPKLPR250400-UGD-EN-1
Rev.: 2025-Oct-20

This documentation has been created for OpenText™ Documentum™ Content Management CE 25.4.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Patents may cover this product, see https://www.opentext.com/patents.

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Table of Contents

Chapter 1

Getting started

1.1 Introduction

OpenText Documentum CM Advanced Workflow is a unified set of tools for rapidly designing and deploying OpenText Documentum CM Advanced Workflow processes. It facilitates team collaboration by enabling process designers to build and reuse processes, reducing the time and cost to build processes.

Note: Place the Advanced Workflow installer under a root-level folder. For example, `C:\advancedwf\`

1.2 Designing an application

Designing an application is an iterative process. Each iteration involves planning, creating and modifying application components, and testing the application to verify that it satisfies business requirements. The complexity of the application determines the number of iterations and the scope of each iteration.

1.3 Planning

Thorough planning is important for processes because it can make the difference in whether a project succeeds or fails. This section addresses various aspects of process planning.

Requirements

Understanding business and system requirements is essential for designing an Advanced Workflow process that meets your needs. Collect the following information as a starting point for identifying Advanced Workflow process requirements.

- Current technical environment.
- High-level and system use cases.
- Estimated transaction volumes such as the number of scanned documents.
- Roles of users, user and group structures, and the policies for their maintenance.
- Relationships between information and how the relationships are maintained.
- Security requirements.
- Business process diagrams.
- System integration points such as email servers, databases, web services, and so on.

- Types of content and information that the system manages, including samples.
- Content for outbound information such as generated letters, emails, and so on.

Deployment

During the planning phase, make note of any aspects of the system that can change when the application is transferred from the development environment to production. For example, databases used by the application generally change when moving to the production environment. Users assigned to specific activities can also be different in the production environment. It is important to compile a complete inventory of these items to use during deployment.

1.4 Creating application and artifacts

After planning your application, create the application from the Home page. Advanced Workflow automatically creates an application project on your file system and the application model. As you build the application, the application model evolves to contain the design-time modeling for all the artifacts that comprise the application.

After creating the application, use the Application navigator to create application-level components. Application-level components are the OpenText Documentum CM repository artifacts that define functionality across the application. You can create application roles, parameters, endpoints, email templates, and security components such as permission sets.

After you create application-level models, use the Object Models navigator to create relationship models between the adopted or migrated data models. For more information, see Adopt Object Models from Repository and Migrating types from Composer.

1.5 Working in shared development environment

Shared development environment involves multiple resources working on different processes or artifacts of an Advanced Workflow application independently and simultaneously. For complete application testing and deployment, you must integrate the different processes or artifacts into the application. When you deploy an application from one of the individual development environment, the deployment will overwrite the existing processes within the same namespace in the OpenText Documentum CM repository. To avoid overwriting the processes, you must manually merge the artifacts from different applications into a central or main `Application/../../Artifacts` folder. After merging the artifacts from different applications, you can use the main application for complete deployment and integrated testing.

For example, if two resources are working in individual development environments for the same application and the first resource deploys the application into the shared repository, application will replace the existing artifacts or processes with new ones. Similarly, when the second resource deploys the application into the shared repository, all the artifacts or processes previously deployed by the first resource will be replaced by the artifacts from the second resource. To avoid this issue, you must manually copy all the artifacts from the individual development environments and merge into the `Artifacts` folder under the main application. Merging the artifacts will retain the existing artifacts and add new ones from the first resource. Similarly, the second resource also must copy the artifacts from the individual development environment and merge with the artifacts in the main application.

To merge artifacts from individual development environments:

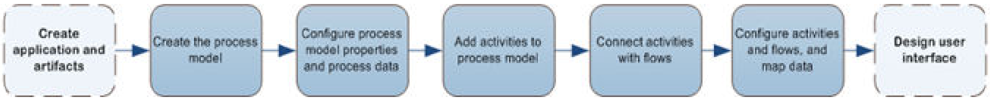
1. Copy the `Application\appName\appName\Artifacts` folder and merge its content with the `Artifacts` folder under the central repository.
2. In the Advanced Workflow designer, refresh the main application to view the merged artifacts.

Note: After merging the artifacts, you must verify that all the required artifacts are listed in the Advanced Workflow application before deployment.

1.6 Designing Advanced Workflow process models

After you create your application and artifacts, you design the process model. A process model defines the information flows through a series of activities. These can be automatically generated activities or activities requiring human interaction. For example, you can create a process model to be used for creating, reviewing, and approving loan applications.

The following graphic illustrates the steps for designing a process model:


Create application and artifacts
Create the process model
Configure process model properties and process data
Add activities to process model
Connect activities with flows
Configure activities and flows, and map data
Design user interface

When designing the process model, you define the properties of the process. This includes deciding if you want to run the process synchronously in a stateless mode or asynchronously in a stateful mode. It also includes determining who can start the process and access process information using permissions.

Process data is data that flows through the process at runtime. Process data includes packages, process variables, and application parameters. The process model properties enable you to specify packages and process variables. Using process data, a process can create or update business objects, content, and folders. You have the ability to restrict end-user access to process data at any point in the process.

A process creates and updates information through process activities that represent the tasks in your business process. Process activities enable you to:

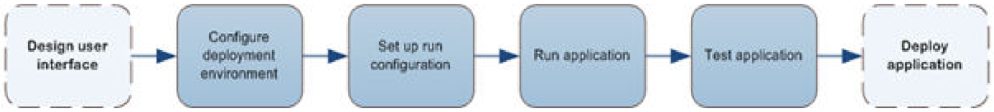
- Create and assign tasks to end users using business rules or logic.
- Send and receive information from external systems using endpoints.
- Extend the process using custom code or use predefined activities.

Information flows through activities in the process model using process flows. Flows connect the activities, determine the paths to take through the process, and ascertain when the activities start. Flows use transitions to determine which paths to take in the process. Flows use triggers to determine a path.

Once you have configured your process model, you can design the user interface for the end users.

1.7 Testing application

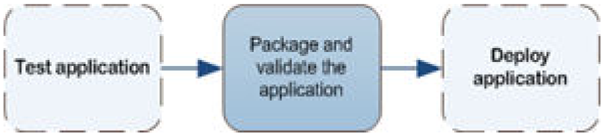
The following graphic illustrates the process for testing the application:



After you design the user interface, you must configure a deployment environment to test your application. By setting up a run configuration and associating it with a deployment environment, you do not have to specify deployment settings every time you test Advanced Workflow. By testing your application, you can verify that it meets the requirements established during the planning phase. If necessary, you can test different parameter and endpoint values by updating those values in the run configuration.

1.8 Packaging application

After you design your application and test it to verify it meets all business requirements, you package it for deployment:



When you package an application, Advanced Workflow does the following:

- Validates the application
- Compiles the application as a web application archive (WAR) file
- Copies the WAR file to the application folder structure on your file system

When the WAR file is available, you deliver it to the system administrator for deployment. The *OpenText Documentum xCelerated Composition Platform Deployment*

Guide and *OpenText Documentum xCelerated Management System Deployment Guide* provide more information on deploying applications.

Chapter 2

Creating applications and projects

2.1 Tasks

2.1.1 Creating an application

1. On the **Home** page, click **New Application**.
2. In the **New application** dialog box, configure basic application properties as described in the following table:

Field	Description
Root folder name	Type a more meaningful application root folder name if you do not want to use the application name. After you create the application, the root folder appears in the Applications folder. The maximum length for the root folder path is 128 characters. The following example shows the path to an application named Loan: `C:\advwfProcess\Applications\Loan`
Description	Type a description of the application.

3. Click **Next**.
4. Configure the application project as described in the following table:

Note: The new application is given a default name that is read-only. You must create only one application in each instance of OpenText Documentum CM Advanced Workflow. In the case of multiple applications, one application will override the other applications during deployment.

Field	Description
Project folder	By default, the system creates the application project folder in the application root folder. To use a different folder, select the option to change the project folders and click **Browse**. The following example shows the path to a local application project folder for an application named Advanced_Workflow: `C:\advwfProcess\Applications\Advanced_Workflow` To use a shared network drive, map a drive on your computer to the Uniform Naming Convention (UNC) path before you create the application. For example, map drive Z to `\\network_drive\advwfProcess_apps`. When you create the application, select the option to change project folders and select the mapped drive. If you use a source control system to share resources, add the application project folder to the source control repository. Refer to your source control system documentation for more information on adding resources to the source control repository.

5. Click **Finish**.

2.1.2 Exporting an application

1. On the **Home** page, select the application, click **Export**.
2. In the **Export application** dialog box, click **Browse**.
3. Navigate to the destination folder and click **OK**.
4. To include the projects on which the application project depends, select **Include all dependent projects**.
5. Click **Finish**.

2.1.3 Importing an application

Importing an application has two purposes:

- Reuse and modify an application developed using the current version of Advanced Workflow.
- Migrate an application developed using previous version.

1. On the **Home** page, click **Import Application**.
2. In the **Import application** dialog box, click **Browse** and navigate to the application root folder on your file system.
3. Click **OK**.
4. Select the application project from the Projects list.
5. To copy the selected projects to the root folder of your new application, select **Import a copy**.

   Do not select this option if you are importing an application that is managed by a source control system.
6. If necessary, type a new application root folder name.
7. Click **Finish**.

2.1.4 Working with a source control system

You can use a third-party source control system to manage application resources. All source control actions are initiated through the third-party source control client. The following high-level steps summarize how to use a third-party source control system to manage resources:

1. Install the source control repository and client.
2. Create an Advanced Workflow application on your file system.
3. Add the application root folder to the source control repository.

   Exclude the following folders (indicated with a forward slash) and files when adding the application root folder and its contents to the repository:

   - configuration/
   - gen/
   - META-INF/
   - target/
   - .index
   - .generatedresources
   - runapp.log

4. Verify that the application is under source control by confirming the following folders (indicated with a forward slash) and files are in the repository:

   - .settings/
   - Artifacts/
   - content/
   - lib/
   - src/
   - .classpath
   - .project
   - build.properties
   - pom.xml

5. To work with an application that another developer has added to the source control repository:

   a. Check out the application from the source control repository to the checkout directory on your file system.

   b. Import the application into your Advanced Workflow environment.

      Do not select the option to import a copy of the application.

   c. After you update shared resources, check them into the source control repository.

   d. To get changes that were checked in by other developers, use your source control client to synchronize your local file system with the changes from the source control repository and then click **Refresh** on the Advanced Workflow toolbar to pick up the latest changes.

Refer to the documentation for your source control system for more information on setting up a source control system and using the source control client to complete source control actions.

2.1.5 Invoking log view

To invoke Log View, perform the following steps:

1. Start Advanced Workflow.

2. When the deployment fails and you encounter with an error message, click **View Log** to invoke the Log view.

   Note: You can also use Ctrl + Alt + L key combination to invoke the Log view.

2.1.6 Customizing log view

1. On the toolbar, click **Preferences**.
2. Select **General > Log View**.
3. To modify the color code you wish to apply for a specific category, select one of the category and click **Edit**.
4. Select the desired color and click **OK**.
5. To retain the default color settings, click **Reset**.
6. If the classification for a log statement appears incorrectly, go to the **Logs** view, modify the classification of a log statement by selecting **Treat this log statement as** followed with an appropriate category. Press Ctrl + Alt + L to invoke Log view.
7. After you have reclassified the log statement, select the check box next to the log statement and click **OK** to reflect the changes.

2.2 Concepts

2.2.1 Application

An application is a collection of resources that represent a business solution. Resources consist of an application project and artifacts that contribute to specific functions within the application.

Every application has one application project that is the root project. The application project is automatically created when you create an application. The application project has an application model artifact that contains the design-time modeling for all the artifacts in the application. The application model also contains metadata that uniquely identifies instances of the application when it is deployed.

Use the Home page to create and import applications. Applications that you create and import appear in the application list on the page. Use the list to switch between applications that you want to work with.

Note: You can not import an application into Advanced Workflow 22.4.

2.2.2 Using references view

While developing an application using Advanced Workflow, you create multiple artifacts linked to each other. As you progressively build the enterprise application, you may not remember all references associated with an artifact. In this scenario, you can select an artifact in the navigator and view the references of the artifact in the References view. The References view lists all references associated with the selected artifact.

The name column lists the inbound references of the selected artifact, and the project column indicates the project to which the reference artifact belongs. To open a reference in the list, right-click the reference, and click **Open**.

When you want to delete an artifact, use the References view to preview all references associated with it.

2.2.3 Searching artifacts

While designing an enterprise application using Advanced Workflow, you can create multiple artifacts. From the pool of artifacts, locating a specific artifact using the Advanced Workflow navigator is a cumbersome task. In this scenario, you can use the global search toolbar in the Advanced Workflow.

As you type a search string in the Search toolbar, the system automatically lists artifacts matching the search string. When you click on the Search toolbar, the system lists all search queries you may have executed as previous choices. For example, you can search for process version artifacts that share the same label using the Search toolbar. The system displays the result as Process Name 1 (v1.0), Process Name 1 (v2.0).

In the search result, the system does not return any result for the hidden artifacts and the artifacts that reside inside libraries.

2.2.4 Log view

Use Log view to identify errors reported in the Advanced Workflow application. After a problem is identified, classified, displayed as log statement, the log viewer aids in quickly identifying the root cause of the problem. Log statements are classified and the problems can be filtered to help diagnose the problem with an appropriate resolution.

By default, the system shows the problem statement. However, you can apply filters to view other categories of errors described as follows:

Categories	Description
Maven goals	Identify execution phases related to the Maven environment.

Categories	Description
Result	Identify statements that may contain information about time taken by specific operations.
Environment settings	Identify log statements related to the current deployment environment.
Show empty and generic goals	When selected, the viewer shows the empty and generic goals.
General debug log	Identify general debug log statements.
Problems	Identify problems related to deployment. By default, the problems category is selected to quickly grab your attention towards cause of failure.

The system may occasionally classify a log statement incorrectly. You can classify the log statement correctly by right-clicking the log statement (**Treat this log statement as**) and then selecting the appropriate classification, as required.

Whenever the deployment fails, the errors now appear in the Log view instead of the system default text editor.

Use wildcard characters judiciously to search for terms in the Log View.

Chapter 3

Creating application-level components

3.1 Tasks

3.1.1 Creating a parameter

1. In the **Application** navigator, click **New Parameter**.
2. In the **New Parameter** dialog box, specify parameter properties as described in the following table:

Field	Description
Label	Type a label for the parameter.
Project	If the application contains projects, select the project for the parameter.
Type	Select the parameter type.

3. Click **Finish**.
4. On the parameter editor, specify parameter properties as described in the following table:

Field	Description
Label	Type a label to identify the parameter. If you change this label, the system generates a new system name.
System Name	Type a system name for internal use by the system to identify the parameter when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Type	If necessary, select a new parameter type.
Value	Type or select a default value for the parameter.
Description	Type a description of the parameter.
Source	Identifies the application project or project to which the parameter belongs.

5. Click **Save**.

3.1.2 Creating a role

1. In the **Application** navigator, click **New Role**.
2. Type a label to identify the role.
3. If the application contains projects, select the project for the role.
4. Click **Finish**.
5. In the **Role** editor, specify role properties as described in the following table:

Field	Description
Label	Type a label to identify the role. If you change this label, the system generates a new system name.
System Name	Type a system name for internal use by the system to identify the role when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the role.
Source	Identifies the project to which the application role belongs.

Do not delete roles created in Advanced Workflow using any other application, such as OpenText Documentum CM Administrator. If roles created in Advanced Workflow are deleted using any other application, the subsequent deployment of the application might fail.

3.1.3 Creating a permission set

1. In the **Application** navigator, click **New Permission Set**.
2. Type a label to identify the permission set.
3. Depending on your requirement, select the class for a permission set as follows:

Class	Description
Template	Creates a permission set template. Template permission sets are used to make applications, workflows, and lifecycles portable. For example, an application that uses a template permission set could be used by various departments within an enterprise.
Public	Creates a public permission set that anyone can use. Public permission sets are available for use by any user in the repository.
Regular	Creates a regular permission set that only the user or group that created it can use it.

4. Click **Finish**.

3.1.4 Configuring permission sets

1. In the **Application** navigator, navigate to the permission set and open it in the **Permission Set** editor.

2. To change the owner of any class of permission set, browse and select the required **Owner**.

3. To add permission set members:

   a. In the **All Users and Groups** area, click one of the following:

      - **Add Alias**: To add a new permission alias. It is applicable only to the permission set template.
      - **Add User**: To add a new permission user.
      - **Add Group**: To add a new permission group.

      By default, two default member entries appear:

      - **dm_owner**: Owner of the permission set. The default permissions for the member are Delete, Execute procedure, and Change Location.
      - **dm_world**: All repository users. The default permissions for the member are None, Execute procedure and Change Location.

   b. For the new Alias set, in the **Permission Details** dialog box, select one of the following:

      - **Relative**: To specify a relative alias name.
      - **Absolute**: To specify aliases from a specific alias set.

      Note: At runtime, the default value set for an alias set is not resolved for the permission set.

4. To specify basic and extended permission level for a permission set member:

   a. Select the member row from the **All Users and Groups** area.

   b. Select the basic **Permissions** from the list.

Level	Description
None	No access to the object.
Browse	View the attribute values of an object but not its content.
Read	View both the attribute values and content of the object. (includes Browse).

Level	Description
Relate	View both the attribute values and content of the object. and comment on the object (includes Browse and Read).
Version	Modify the content of the object and check in a new version of the object with a new version number (includes Browse, Read, and Relate).
Write	Edit the attribute values of the object and check in the object as the same version (includes Browse, Read, Relate and Version).
Delete	Delete the object (includes Browse, Read, Relate, Version, and Delete).

c. Click the **Extended Permissions** cell and select the permission level.

Level	Description
Execute Procedure	Superusers can change the owner of an item and use Run Procedure to run external procedures on certain object types.
Change Location	Move an object from one folder to another in the repository
Change State	Modify the state of an item that has a lifecycle applied to it.
Change Permissions	Modify the basic permissions of an item.
Change Ownership	Change the owner of an item.
Extended Delete	Allow to delete the object only.
Change Folder Links	Allow to bypass the folder security.

5. To remove a permission set member, select the member row and click **Remove**.

6. Click **Save**.

3.1.5 Creating an endpoint

1. In the **Application** navigator, click **New Endpoint**.
2. Type a label to identify the endpoint.
3. If the application contains projects, select the project for the endpoint.
4. In the **Type** list, select the endpoint type.
5. Click **Finish**.

3.1.6 Configuring a database endpoint

1. In the database endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the database endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the database endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the database endpoint.
Source	Identifies the application or project to which the endpoint belongs.

2. Specify connection details as described in the following table:

Field	Description
JDBC driver	Select a driver to connect to the database. Custom drivers as well as pre-installed drivers appear here.
Connection string	Type a JDBC connection string for your chosen database.

3. Specify authentication as described in the following table:

Field	Description
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.

Field	Description
Password	Type a password to authenticate the connection.

4. Click **Test Connection** to verify the connection.

5. Click **Save**.

3.1.7 Configuring an email endpoint

1. In the email endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the email endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the email endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of how to use the email endpoint.
Source	Identifies the application or project to which the endpoint belongs.
Server connection	Select whether you are creating an outgoing or incoming Email endpoint.

2. For an outgoing email endpoint, specify the **Outgoing Server Settings** as described in the following table:

Field	Description
Hostname	Type the hostname or IP address of the SMTP server used to send the email message.
Port number	Type the port number to use for the outgoing server connection.
Encryption	Select the encryption protocol to send the email message.

3. For an incoming email endpoint, specify the **Incoming Server Settings** as described in the following table:

Field	Description
Hostname	Type the hostname or IP address of the email server used to retrieve the email message.
Protocol	Select the protocol of the incoming email server to retrieve the email message.
Port number	Type the port number to use for the incoming email server connection.
Encryption	Select the encryption protocol to retrieve the email message.

4. Specify authentication as described in the following table:

Field	Description
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type the password used to authenticate the connection.

5. For outgoing email endpoints, click **Test Outgoing Connection** to test the outgoing SMTP connection to the email server.

6. For incoming email endpoints, click **Test Incoming Connection** to test the incoming IMAP or POP3 connection to the email server.

7. Click **Save**.

3.1.8 Configuring an FTP endpoint

1. In the FTP endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the FTP endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the FTP endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the FTP endpoint.

Field	Description
Source	Identifies the application or project to which the endpoint belongs.

2. Specify connection details as described in the following table:

Field	Description
Protocol	Select a protocol for the FTP transmissions.
Hostname	Type the hostname or IP address of the FTP server.
Port number	Type the port number for the protocol you selected.
Base folder path	Type the path to the folder to designate as source or target for file transfers.

3. Specify authentication information as described in the following table:

Field	Description
Basic authentication	Select to use basic authentication.
Public key authentication	Select to use public key authentication.
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type a password to authenticate the connection.

4. If available, click **Test Connection** to verify the connection to the endpoint.

5. Click **Save**.

3.1.9 Configuring an HTTP endpoint

1. In the HTTP endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the HTTP endpoint. If you change this label, the system generates a new system name.

Field	Description
System name	Type a system name for internal use by the system to identify the HTTP endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the HTTP endpoint.
Source	Identifies the application or project to which the endpoint belongs.

2. In the **URL** field, type the fully qualified URL of the site to which the activity posts content.

3. Specify HTTP basic authentication as described in the following table:

Field	Description
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type a password to authenticate the connection.

4. Click **Test Connection** to verify the connection.

5. Click **Save**.

3.1.10 Configuring a JMS endpoint

1. In the JMS endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the JMS endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the JMS endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the JMS endpoint.
Source	Identifies the application or project to which the endpoint belongs.

2. Specify initial context as described in the following table:

Field	Description
ContextFactory	Select the initial ContextFactory that the system uses to access the Java Naming and Directory Interface (JNDI) context of the messaging server.
Provider URL	Type the Provider URL of the messaging server.
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type a password to authenticate the connection.
Additional properties	Type any additional properties required to access messages from the JMS server. At runtime, InitialContextFactory uses these attributes to construct the InitialContext.

3. Specify queue/topic information as described in the following table:

Field	Description
Queue	Select this option to access a queue.
Topic	Select this option to access a topic.
Connection factory	Select the connection factory used to access the queue or topic.
Queue/Topic name	Type the name of the queue or topic to access.
Username	Type a user name to access the queue or topic.
Password	Type a password to access the queue or topic.

4. Click **Test Connection** to verify the connection.

5. Click **Save**.

3.1.11 Configuring a repository endpoint

1. In the repository endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the repository endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the repository endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of how to use the repository endpoint.
Source	Identifies the application or project to which the endpoint belongs.

2. Specify connection details as described in the following table:

Field	Description
Docbroker hostname	Type the name of the host that contains the doc broker where the repository is registered.
Dockbroker port	Type the port of host that contains the doc broker where the repository is registered.
Repository name	Click **Refresh** to get a list of repositories from the connection broker. Select the name of a repository that is configured for the runtime connection broker.

3. Specify authentication as described in the following table:

Field	Description
Username	Type the name used to authenticate to the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type the password used to authenticate to the connection.
Domain	Type the domain to authenticate to the repository.

4. Click **Test Connection** to verify the connection to the repository.

5. Click **Save**.
6. If you have multiple OpenText Documentum CM repositories to integrate into your application, create a repository endpoint for each repository.

3.1.12 Configuring a web service endpoint

1. In the web service endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the web service endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the web service endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the web service endpoint.
Source	Identifies the application or project to which the endpoint belongs.

2. In the **WSDL URL** field, type the fully qualified URL to the WSDL file for the operation you want to run.
3. Specify HTTP basic authentication as described in the following table:

Field	Description
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type a password to authenticate the connection.

4. Click **Test Connection** to verify the connection to the endpoint.
5. Click **Save**.

3.1.13 Accessing a web service through an HTTP proxy server

To access a web service that is located outside the firewall through an HTTP proxy server, modify the Advanced Workflow configuration file. The *OpenText Advaned Workflow Deployment Guide* provides information on how to configure an HTTP proxy server for web services at runtime.

1. Navigate to the directory where Advanced Workflow is installed and open the `advwfProcess.ini` file.

   For example, open `C:\advwfProcess\advwfprocess.ini`.

2. Type the following Java virtual machine arguments following -vmargs:

   ```
   -Dhttps.proxyHost=<proxy_host>
   -Dhttps.proxyPort=<port_number>
   -Dhttps.nonProxyHosts="<non_proxy_hosts>"
   ```

   Where:

   - <*proxy_host*> is the hostname of the proxy server.
   - <*port_number*> is the port number of the proxy server. The default value is 80 for HTTP Basic authentication and 443 for HTTP over SSL.
   - <*non_proxy_hosts*> is the list of hosts that can be reached directly, bypassing the proxy server. This list contains regular expressions separated by '`|`'. Any host matching one of these regular expressions is reached through a direct connection instead of through a proxy.

3.1.14 Configuring the xPression Server endpoint

1. In the xPression endpoint editor, specify basic information as described in the following table:

Field	Description
Label	Type a label to identify the xPression Server endpoint. If you change this label, the system generates a new system name.
System name	Type a system name for internal use by the system to identify the xPression Server endpoint when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the xPression Server endpoint.
Source	Identifies the application or project to which the endpoint belongs.

2. In the **xPression URL** field, type a URL for the xPression service.

3. Specify authentication as described in the following table:

Field	Description
Username	Type a user name to authenticate the connection. If you do not specify a user name and password, no credentials are supplied to the connection.
Password	Type the password used to authenticate the connection.

4. Click **Test Connection** to verify the connection.

5. Click **Save**.

3.1.15 Working with an application navigator

To delete an artifact, right-click the artifact and select **Delete** or select the artifact and press Delete.

3.1.16 Creating a Java extension

1. In the **Application** navigator, click **New Java Extension**.

2. In the **New Java Extension** dialog box, specify parameter properties as described in the following table:

Field	Description
Java Module/Java library	Depending on your requirement, you can specify either Java module or Java library as the Java extension.
Label	Type a label of the Java library.
Description	Type a description of the library.

3. Click **Finish**.

4. On the **Basics** tab in the editor, specify library properties as described in the following table:

Field	Description
Label	Identifies the label of the extension.
System Name	Identifies the system name for internal use by the system to identify the extension when used in the API. This name is based on the label. If you change the label, the system generates a new system name.
Description	Type a description of the extension.

Field	Description
Source	Identifies the application project or project to which the extension belongs.
File location	Identifies the path to the Java extension on your file system.

5. On the **Java Library** tab in the editor, specify library properties as described in the following table:

Field	Description
JAR Files	Add the JAR files from your local file system.
Property Files	Add the property files from your local file system.

   You can also perform delete or edit operation on any implementation JAR file or property file.

6. On the **Java Module** tab in the editor, specify Java module properties as described in the following table:

Field	Description
Implementation JARs	Add the implementation JAR files and classes from your local file system. Select the primary class from the selected JAR file. Once added, you can move the implementation JAR file to the Java library by right-clicking the selected JAR file.
Interface JARs	Add the interface JAR files from your local file system.
Java Libraries	Associate one or many Java libraries with the Java module. You can select any Java library you defined.

   You can perform delete or edit operation on any implementation JAR file, interface JAR file, or Java library.

7. Click **Save**.

3.1.17 Creating an alias set

1. In the **Application** navigator, click **New > Alias Sets**.
2. Type a label to identify the alias set.
3. If the application contains projects, select the project for the alias set.
4. To add one or more aliases that makes up the alias set:
   - In the **Aliases** area, click **Add**.
   - In the **New Alias** dialog box, specify the alias name.
   - In the **Type** list, select any alias type:
     - Unknown
     - User: Single application user.
     - Group: Group of application users.
     - User or Group: Either user or group.
     - Cabinet or Folder path: Path to a cabinet or folder in the OpenText Documentum CM repository.
     - Permission Set: Define the permission level that you can map to a permission set member.
5. Click **OK**.
6. In the **Alias Details** area, provide values for alias types as follows:

   Depending on the alias type you selected, various parameter and value options appear in the **Value** field.

Alias Type	Value	Description
Unknown	Value	Specify the value of the alias.
User	• Leave it Blank • Parameter	Specify the parameter of the alias. You can leave the value of the alias as blank.
Group	• Leave it Blank • Parameter • Value	Specify the parameter or value of the alias. You can leave the value of the alias as blank.
User or Group	• Leave it Blank • Parameter	Specify the parameter of the alias. You can leave the value of the alias as blank.
Cabinet or folder path	• Parameter • Value	Specify the parameter or value of the alias.

Alias Type	Value	Description
Permission Set	• Parameter • Value	Specify the parameter or value of the alias.

7. Click **Save**.

3.2 Concepts

3.2.1 Roles

A role represents a specific job function for a type of user within an application. For example, an application for processing mortgage loans could have separate roles for the loan applicant, loan originator, and loan processor. After you create the role, you can map it to a permission level to control access to application data. The permission level determines the level of access the user has to application data.

A user can be assigned to more than one role.

3.2.2 Permission sets

A permission set controls end-user access to object instances at application runtime. When you create a permission set in Advanced Workflow, you map a permission set member (application role, User parameter, or Group parameter) to a permission level. The user, alias, or group parameter values are resolved at runtime. The system resolves the alias while creating the instance of a permission from the template.

You can set the basic and extended permission levels and assign to a permission set member. When you select a permission level, consider if the level is appropriate for the object model. For example, a level of Browse is more appropriate for a business object because a business object does not have content.

The permission sets can be migrated from earlier version to the current version of Advanced Workflow. During migration, the system maps the permission sets as follows:

Previous Version	Current Version
No Access	None
Browse	Browse
Review	Relate
Edit	Write and Change Location
Administer	Delete and Change Location
Browse and Delete	Browse and Extended Delete

3.2.3 Connecting to external systems using endpoints

An endpoint is an artifact that represents an external system, making it easy for your application to connect to that system. When you create activities, you can select the pre-configured endpoints as sources or destinations for data that the activities consume or produce. Multiple activities can reuse the same endpoint. You can provide an endpoint with default configuration settings.

Endpoints enable you to deploy your application to several environments with different external system connections without having to change the process model definitions. For example, you can create a database endpoint and map it to different databases in your development, test, and production environments. The mappings override the default configuration settings of the endpoint.

3.2.4 Determining which endpoint to use

Use the following table to determine which endpoint to use:

Endpoint	Description
Database "Configuring a database endpoint" on page 29	Connects to an external SQL database.
Email "Configuring an email endpoint" on page 30	Connects to an external mail server.
FTP "Configuring an FTP endpoint" on page 31	Connects to a file transfer server that uses FTP, FTPS, or SFTP.
HTTP "Configuring an HTTP endpoint" on page 32	Connects to an external web server.
JMS "Configuring a JMS endpoint" on page 33	Connects to an external messaging server that uses Java Messaging Service.
Repository "Configuring a repository endpoint" on page 35	Connects to an external OpenText Documentum CM repository.
Web service (SOAP) "Configuring a web service endpoint" on page 36	Connects to a SOAP-based web service.
xPression "Configuring the xPression Server endpoint" on page 37	Connects to a Document Sciences xPression server.

3.2.5 Advanced Workflow log file

When you start Advanced Workflow, the system creates the advwfProcess.log file. This file contains all log messages except application deployment messages. The file is continually updated.

To view the file, navigate to `advwfProcess\logs` and open the `advwfProcess.log` file in a text editor.

Cleanup of temporary files

To activate the cleanup of temporary DAR files in the Windows environment, you must now set the JVM flag `com.emc.advwfProcess.builder.dar.autoclean` to true, in any of the following scenarios:

- For Advanced Workflow, add the JVM flag to `DocumentumAdvancedWorkflow.ini` file.
- For the application server, configure the JVM flag in server configuration.
- If you use a DFC JAVA code to fetch an object created by Advanced Workflow, then the JAVA command to execute that code should add the JVM flag. If you deploy multiple applications on the same application server, do not enable the JVM flag,

3.2.6 Java extensions

Java extensions allow you to add custom functions and business process capabilities to an Advanced Workflow application, such as integrating with external systems that do not provide a standard integration mechanism, triggering actions in external systems, removing attachments from email messages, or extracting names files from within a single zip file.

Java extensions primarily consists of these elements:

Java Module

Is an artifact file that aggregates multiple Java JAR archive files and is used to distribute Java classes and associated metadata.

A Java Module is composed of two types of JAR files: Interface JARs and Implementation JARs. Interface JARs contain Java interface classes and the implementation JARs contain the classes that implement the interface classes. Generally, the interface classes and the implementation classes are aggregated in separate JAR files.

A Java Module can also contain one or more Java Library.

Java Library

Is a set of Java libraries that can be attached to a Java module. It is an artifact file that aggregates multiple JAR files and associated property files.

A Java Library allows reuse of the common code across various Java modules without having to duplicate the JARs. For example, Apache file I/O library.

Note: Java Modules and Java Libraries support JARs compiled with Java 1.6, Java 1.7, and Java 1.8 only.

3.2.7 Alias set

Within Advanced Workflow applications, multiple references are made to many users, groups, permission sets, repository, cabinet or folders. Instead of referencing the actual user, group, cabinet, or folder name, you can assign an alias, a symbolic name. The symbolic name is called the alias name. The actual value is called the alias value. A collection of aliases is called an alias set.

3.2.8 Default alias set

The default alias set is an alias set whose values are resolved by the workflow initiator when starting the workflow. The aliases in the default alias set must not be mapped to any values. You can select an alias set as a default alias set or create a new default alias set using OpenText Documentum CM Advanced Workflow. For more information about default alias sets and alias sets, see the *OpenText Documentum Administrator User Guide*.

Chapter 4

Object model relationship

4.1 Concepts

4.1.1 Relationship example

An application has business objects named Company and Person. You want to configure a two-way relationship between the business objects. You want the Company business object to access the attributes of the Person business object and you want the Person business object to access the attributes of the Company business object.

Configure the Company business object to have a one-to-many relationship with the Person business object because one company can have many employees. Since you created the relationship through the Company business object, it is the source object. Person is the target object.

Because the company refers to a person as an employee, use Employee as the label for the Company-Person direction of the relationship. Because the person refers to the company as an employer, use Employer as the label for the Person-Company direction of the relationship. The following figures show the relationship when viewing it from each business object:

4.1.2 Behaviors of relationship configurations

The following table describes the behaviors of object model relationship configurations and the end results for customer to collateral business objects:

Relationship		Runtime relationship	Behavior	
Direction	Type			
One-way	Has one (With qualifying attributes)		1. Create a relationship between Customer1 and Collateral1 business object instances. 2. Create a relationship between Customer1 and Collateral2 business object instances. 3. Create a relationship between Customer2 and Collateral2 business object instances. 4. Create a relationship between Customer2 and Collateral1 business object instances. 5. Repeat step 1. 6. Repeat step 2.	1. Relationship between Customer1 and Collateral1 is created. 2. Relationship between Customer1 and Collateral1 is removed. Relationship between Customer1 and Collateral2 is created. 3. Relationship between Customer1 and Collateral2 is removed. Relationship between Customer2 and Collateral2 is created. 4. Relationship between Customer2 and Collateral2 is removed. Relationship between Customer2 and Collateral1 is created 5. Relationship between Customer2 and Collateral1 is removed. Relationship between

Relationship		Runtime relationship	Behavior	
Direction	Type			
				Customer1 and Collateral1 is created. 6. Relationship between Customer1 and Collateral1 is removed. Relationship between Customer1 and Collateral2 is created. Relationship(s) in the system: Customer1 and Collateral2

Relationship		**Runtime relationship**	**Behavior**	
Direction	**Type**			
Customer to Collateral	One-way	Has one (No qualifying attributes)	1. Create a relationship between Customer1 and Collateral1 business object instances. 2. Create a relationship between Customer1 and Collateral2 business object instances. 3. Create a relationship between Customer2 and Collateral2 business object instances. 4. Create a relationship between Customer2 and Collateral1 business object instances. 5. Repeat step 3. 6. Repeat step 4.	1. Relationship between Customer1 and Collateral1 is created. 2. Relationship between Customer1 and Collateral1 is removed. Relationship between Customer1 and Collateral2 is created. 3. Relationship between Customer1 and Collateral2 is removed. Relationship between Customer2 and Collateral2 is created. 4. Relationship between Customer2 and Collateral2 is removed. Relationship between Customer2 and Collateral1 is created 5. Relationship between Customer2 and Collateral1 is removed. Relationship between

Relationship		Runtime relationship	Behavior	
Direction	Type			
				Customer1 and Collateral1 is created. 6. Relationship between Customer1 and Collateral1 is removed. Relationship between Customer1 and Collateral2 is created. Relationship(s) in the system: Customer1 and Collateral2

Relationship		Runtime relationship	Behavior	
Direction	Type			
Customer to Company	One-way	Has many (With qualifying attributes)	1. Create a relationship between Customer1 and Company1 business object instances. 2. Create a relationship between Customer1 and Company2 business object instances. 3. Create a relationship between Customer2 and Company2 business object instances. 4. Create a relationship between Customer2 and Company1 business object instances. 5. Repeat step 1. 6. Repeat step 2.	1. Relationship between Customer1 and Company1 is created. 2. Relationship between Customer1 and Company2 is created. 3. Relationship between Customer1 and Company2 is removed. Relationship between Customer2 and Company2 is created. 4. Relationship between Customer1 and Company1 is removed. Relationship between Customer2 and Company1 is created 5. Relationship between Customer2 and Company1 is removed. Relationship between Customer1 and Company1 is created. 6. Relationship between

Relationship		Runtime relationship	Behavior	
Direction	Type			
				Customer2 and Company2 is removed. Relationship between Customer1 and Company2 is created. Relationship(s) in the system: Customer1 and Company1 Customer1 and Company2

Relationship		Runtime relationship	Behavior	
Direction	Type			
Customer to Company	One-way	Has many (No qualifying attributes)	1. Create a relationship between Customer1 and Company1 business object instances. 2. Create a relationship between Customer1 and Company2 business object instances. 3. Create a relationship between Customer2 and Company2 business object instances. 4. Create a relationship between Customer2 and Company1 business object instances. 5. Repeat step 1. 6. Repeat step 2.	1. Relationship between Customer1 and Company1 is created. 2. Relationship between Customer1 and Company2 is created. 3. Relationship between Customer1 and Company2 is removed. Relationship between Customer2 and Company2 is created. 4. Relationship between Customer1 and Company1 is removed. Relationship between Customer2 and Company1 is created 5. Relationship between Customer2 and Company1 is removed. Relationship between Customer1 and Company1 is created. 6. Relationship between

Relationship		Runtime relationship	Behavior	
Direction	Type			
				Customer2 and Company2 is removed. Relationship between Customer1 and Company2 is created. Relationship(s) in the system: Customer1 and Company1 Customer1 and Company2

Relationship		Runtime relationship	Behavior	
Direction	Type			
Customer to Insurance	Two-way	One-to-many (No qualifying attributes)	1. Create a relationship between Customer1 and Insurance1 business object instances. 2. Create a relationship between Customer1 and Insurance2 business object instances. 3. Create a relationship between Customer2 and Insurance2 business object instances. 4. Create a relationship between Customer2 and Insurance1 business object instances.	1. Relationship between Customer1 and Insurance1 is created. 2. Relationship between Customer1 and Insurance2 is created. 3. Relationship between Customer1 and Insurance2 is removed. Relationship between Customer2 and Insurance2 is created. 4. Relationship between Customer1 and Insurance1 is removed. Relationship between Customer2 and Insurance1 is created Relationship(s) in the system: Customer2 and Insurance2 Customer2 and Insurance1

Relationship		Runtime relationship	Behavior	
Direction	Type			
Insurance to Customer	Two-way	Many-to-one (No qualifying attributes) Read only	1. Create a relationship between Insurance1 and Customer1 business object instances. 2. Create a relationship between Insurance1 and Customer2 business object instances. 3. Create a relationship between Insurance2 and Customer2 business object instances. 4. Create a relationship between Insurance2 and Customer1 business object instances.	1. Relationship between Insurance1 and Customer1 is created. 2. Relationship betweem Insurance1 and Customer1 is removed. Relationship between Insurance1 and Customer2 is created. 3. Relationship between Insurance2 and Customer2 is created. 4. Relationship between Insurance1 and Customer1 is removed. Relationship between Insurance2 and Customer1 is created Relationship(s) in the system: Insurance1 and Customer2 Insurance2 and Customer1

Relationship		Runtime relationship	Behavior	
Direction	Type			
Customer to Loan	Two-way	Many-to-one (With qualifying attributes)	1. Create a relationship between Customer1 and Loan1 business object instances. 2. Create a relationship between Customer1 and Loan2 business object instances. 3. Create a relationship between Customer2 and Loan2 business object instances. 4. Create a relationship between Customer2 and Loan1 business object instances. 5. Repeat step 3. 6. Repeat step 4.	1. Relationship between Customer1 and Loan1 is created. 2. Relationship between Customer1 and Loan1 is removed. Relationship between Customer1 and Loan2 is created. 3. Relationship between Customer2 and Loan2 is created. 4. Relationship between Customer2 and Loan2 is removed. Relationship between Customer2 and Loan1 is created 5. Relationship between Customer1 and Loan2 is removed. Relationship between Customer1 and Loan1 is created. 6. Relationship between Customer1 and Loan1 is removed. Relationship between Customer1

Relationship		Runtime relationship	Behavior	
Direction	Type			
				and Loan2 is created. Relationship(s) in the system: Customer1 and Loan2 Customer2 and Loan1

Relationship		**Runtime relationship**	**Behavior**	
Direction	**Type**			
Customer to Bank	Two-way	Many-to-many (No qualifying attributes)	1. Create a relationship between Customer1 and Bank1 business object instances. 2. Create a relationship between Customer1 and Bank2 business object instances. 3. Create a relationship between Customer2 and Bank2 business object instances. 4. Create a relationship between Customer2 and Bank1 business object instances. 5. Repeat step 1. 6. Repeat step 2.	1. Relationship between Customer1 and Bank1 is created. 2. Relationship between Customer1 and Bank2 is created. 3. Relationship between Customer2 and Bank2 is created. 4. Relationship between Customer2 and Bank1 is created 5. Relationship between Customer1 and Bank1 is created. 6. Relationship between Customer1 and Bank2 is created. Relationship(s) in the system: Customer1 and Bank1 Customer1 and Bank2 Customer2 and Bank2 Customer2 and Bank1 Customer1 and Bank1

Relationship		Runtime relationship	Behavior	
Direction	Type			
				Customer1 and Bank2

Relationship		Runtime relationship	Behavior	
Direction	Type			
Bank to Customer	Two-way	Many-to-many (No qualifying attributes)	1. Create a relationship between Bank1 and Customer1 business object instances. 2. Create a relationship between Bank1 and Customer2 business object instances. 3. Create a relationship between Bank2 and Customer2 business object instances. 4. Create a relationship between Bank2 and Customer1 business object instances. 5. Repeat step 1. 6. Repeat step 2.	1. Relationship between Bank1 and Customer1 is created. 2. Relationship between Bank1 and Customer2 is created. 3. Relationship between Bank2 and Customer2 is created. 4. Relationship between Bank2 and Customer1 is created 5. Relationship between Bank1 and Customer1 is created. 6. Relationship between Bank1 and Customer2 is created. Relationship(s) in the system: Bank1 and Customer1 Bank2 and Customer1 Bank2 and Customer2 Bank2 and Customer1 Bank1 and Customer1

Relationship		Runtime relationship	Behavior	
Direction	Type			
				Bank1 and Customer2
Customer to Customer		Create a relationship between instances of the same business object	1. Create a relationship between Customer1 and Customer2 business object instances. You cannot create a relationship between an instance of a business object and itself, only between different instances of the business object.	1. Relationship between Customer1 and Customer2 is created. Relationship(s) in the system: Customer1 and Customer2

Relationship		Runtime relationship	Behavior	
Direction	Type			
Customer to Partner	Two-way	One-to-one (With qualifying attributes)	1. Create a relationship between Customer1 and Partner1 business object instances. 2. Create a relationship between Customer1 and Partner2 business object instances. 3. Create a relationship between Customer2 and Partner2 business object instances. 4. Create a relationship between Customer2 and Partner1 business object instances.	1. Relationship between Customer1 and Partner1 is created. 2. Relationship between Customer1 and Partner1 is removed. Relationship between Customer1 and Partner2 is created. 3. Relationship between Customer1 and Partner2 is removed. Relationship between Customer2 and Partner2 is created. 4. Relationship between Customer2 and Partner2 is removed. Relationship between Customer2 and Partner1 is created Relationship(s) in the system: Customer2 and Partner1

Relationship		Runtime relationship	Behavior	
Direction	Type			
Partner to Customer	Two-way	One-to-one (With qualifying attributes)	1. Create a relationship between Partner1 and Customer1 business object instances. 2. Create a relationship between Partner1 and Customer2 business object instances. 3. Create a relationship between Partner2 and Customer2 business object instances. 4. Create a relationship between Partner2 and Customer1 business object instances.	1. Relationship between Partner1 and Customer1 is created. 2. Relationship between Partner1 and Customer1 is removed. Relationship betweenPartner1 and Customer2 is created. 3. Relationship between Partner1 and Customer2 is removed. Relationship between Partner2 and Customer2 is created. 4. Relationship between Partner2 and Customer2 is removed. Relationship between Partner2 and Customer1 is created Relationship(s) in the system: Partner2 and Customer1

Chapter 5

Building process models

5.1 Tasks

5.1.1 Creating a process model

1. In the **Processes** navigator, click **New Process**.
2. In the **New Process** dialog box, specify project model properties as described in the following table:

Field	Description
Label	Type a label for the process model.
Process can run in stateless mode	Select the **Process can run in stateless mode** check box to enable this process model to run as a stateless process at runtime.

3. Select **Finish**.

5.1.2 Configuring general information for your process model

Use the **Process Properties** dialog box to provide information about the process model you are creating. These settings are applied to the process and the data used in that process.

1. Click **Process Properties**.
2. On the **General** tab, in the **Description** field, type a description for the process model.
3. In **Workflow Instruction**, type instructions for the performers of the activities.
4. In **Process Settings**, specify settings for the process as described in the following table:

Field	Description
Process can run in stateless mode	Specify whether to enable this process model to run as a stateless process at runtime.

Field	Description
Store content name to the package at runtime	Specify whether to make the names of routed content available for display to users. By default, the package routed through a process does not include the names of the content in the package for security reasons. Select this option to store the content names in the package to have the names used in the instructions passed to process participants.
Audit process events	Specify whether to enable event auditing for system events.

5. In the **Default Email Notifications** area, click **Add Event** to define the default email notifications and the default email template for the process activities. You can override these default email notifications in the activity configurations.

6. In the **Event** column, double-click the first row and select an event.

7. In the **Email Template** column, click the first row.

   a. In the **Notification Template** wizard, in the **Email Template** field, click **Select**.

   b. In the **Select Email Template** dialog box, select an email template and click **Finish**.

   c. In the **Preview** area, verify that the template is correct and click **Next**.

   d. Complete the **Input Message Mapping** and **Output Message Mapping** screens as described in "Configuring a Send Email activity" on page 110 and click **OK**.

8. On the **General** tab, click **Add Event** to add additional default email notifications.

9. Click **OK**.

5.1.3 Creating process variables

1. Click **Process Properties**.

2. On the **Data** tab, select the **Process Variables** node in the tree view and click **Add**.

3. Configure the variable definition as described in the following table:

Field	Description
Name	Type the name of the variable.
Type	Select a data type for the variable.

Field	Description
Single Value	Select if the variable has a single value. For example, a Ticker Symbol variable for a Stock Exchange process could be a string.
Multivalue	Select if the variable contains multiple values. For example, a Neighborhoods variable for a Real Estate Sales process could be a list of strings.
Default Value	For a single value variable, type a default value if desired.

To make a process variable not visible, configure it within an activity.

4. For a variable in a stateless process, configure data inputs and outputs for data mapping as defined in the following table:

Field	Description
Not used for input or output	Select to make the variable not visible.
Use for input only	Select to use the variable for data inputs.
Use for output only	Select to use the variable for data outputs.

5. Repeat step 2 through step 4steps 2 through 4 for each process variable you add to the process model and click **OK**.

   If a process has a large number of process variables, improve performance by replacing your process variables with a single business object package.

5.1.4 Creating packages

1. Click **Process Properties**.
2. On the **Data** tab, select the **Packages** node in the tree view and click **Add**.
3. Configure the package definition as described in the following table:

Field	Description
Name	Type a unique name to identify the package.
Type	Click **Select**, select the object type of the package, and click **Finish**.

Field	Description
Version	If you select a content package, select or type the version option that you want to use in the process: • Select **<Any>** if any version of the package can be used. • Type a specific version number, for example, 2.5 or 3.0. If you type a specific version number, the package contains that version of the content. • Type a symbolic version label, for example, Draft. If you type a version label, the label is case-sensitive and should match the version of the object in the repository. • Select **CURRENT** if you want the package to contain the version labeled CURRENT, which is the most current version of the object in the repository. The specified version appears as the default version used for each activity that handles the package. You can override the version when you configure the activity.
Visible across entire process	Sets the package as visible to every activity in the business process on the assumption that the package flows through the entire process. When you clear this option, the package is visible only in activities to which you explicitly add it. If you select a package that has different visibility settings at the global (Process Properties) and process activity levels, this option is unavailable. To reset all activities to the same value, click the **Change activity-level settings** link and in the dialog box that appears, click **Yes**.
This is a mandatory package	Sets the package as required by the process. If you select a package that has different mandatory package settings at the global (Process Properties) and process activity levels, this option is unavailable. To reset all activities to the same value, click the **Change activity-level settings** link and in the dialog box that appears, click **Yes**.

4. Repeat these steps for each package you add to the process model and click **OK**.

   If a process has a large number of process variables, improve performance by replacing your process variables with a single business object package.

5.1.5 Adding content to a process

1. In the **Processes** navigator, double-click the process model to open the process model editor.

2. Click **Process Properties**.

3. On the **Data** tab, select the **Packages** node in the tree view and click **Add**.

4. Configure the package definition as described in the following table:

Field	Description
Name	Type a name to refer to the content object at runtime.
Version	If the content object has a version, select the version that is referred to by the package.
Type	Select the content model to link to the package.
Visible across entire process	Select to make the package visible within all activities. You can override this selection in a specific activity configuration.
This is a mandatory package	Select to require the task to have the package before starting the task. You can override this selection in a specific activity configuration.

5. Click **OK**.

5.1.6 Specifying process permissions

1. Click **Process Properties**.

2. On the **Advanced** tab, specify process permissions as described in the following table:

Field	Description
Limit who can start this process and edit process variables	Enables you to restrict the number of end users who can start a process and edit process variables at runtime.
Permission Set	Select the permission set that includes the limited process permissions and click **Finish**.

3. Click **OK**.

5.1.7 Selecting a calendar for your process model

1. Click **Process Properties**.

2. On the **Advanced** tab, specify the default calendar to use for all activity timer calculations as described in the following table:

Field	Description
Use standard calendar	The system uses the standard calendar to calculate timer duration. The standard calendar is based on absolute time. It is a 24 x 7 calendar with no holidays or time off.
Use specific business calendar	The system uses a specified business calendar to calculate timer duration. A business calendar includes business hours and holidays. Select the business calendar from the application parameters and click **Finish**.

3. Click **OK**.

5.1.8 Creating a correlation set

1. Create process variables to use in the correlation set. Do not define a default value for these process variables. “Creating process variables” on page 66 provides instructions for adding the process variables.

2. Click **Process Properties**.

3. On the **Advanced** tab, click **Add correlation set**.

4. To name the correlation set, right-click the correlation set, select **Edit set name**, type a label, and click **OK**.

5. To add process variables to the correlation set:

    a. Select the correlation set and click **Add correlation set** once for each process variable that you want to include in the correlation set. For example, to add two process variables, click **Add correlation set** twice.

    b. For each new string, right-click the string and select **Edit correlation model**. In the **Correlation** dialog box, select a process variable from the tree and click **OK**.

6. Click **OK**.

5.1.9 Enabling inbound web service activities

If your process model contains the **Start from Web Service** or **Wait for Web Service** activities, follow this procedure to specify the target namespace Uniform Resource Indicator (URI) for the WSDL location. If there are multiple web service activities within the process model, they use the same target namespace.

1. Click **Process Properties** and then click **Advanced**.
2. In the **Target namespace URI** field, type the namespace used in the WSDL.
3. Click **OK**.

5.1.10 Configuring the default alias set

1. Select a process from the **Processes** list.
2. Click the **Process Properties** icon. The Process Properties dialog box is displayed.
3. On the **General** tab, select an alias from the **Default Alias Set** list to set it as the default alias set. The **Default Alias Set** list displays the alias sets from repository and those created in Advanced Workflow.
4. Click **OK** to mark an alias set as the default alias set.

5.1.11 Adding a performer from a default alias set to a task

1. Open the **Task Properties** to select a performer from the default alias set.
2. Navigate to the **Performers** tab and click **Add Performer**.
3. In the **Add Performer** dialog box, select **Group from Alias Set** or **User from Alias Set**.
4. From the **Select Alias Sets** list, select the alias set that is marked as **Default Alias Set**.
5. From the **Select Alias** list, select an alias set user and click **OK**.

Note: When you define an alias set category in OpenText Documentum CM Administrator, make sure that the value is not set for the alias set category. The default alias set category value must be mapped to a user or group during runtime only.

5.1.12 Changing default alias set for a process

You can also change the default alias set for a process in Advanced Workflow.

1. Select a process from the **Processes list**.
2. Click the **Process Properties** icon.
3. In the **Process Properties** dialog box, on the **General** tab, select a different alias set from the **Default Alias Set** list. When you change the default alias set, all the performers attached to the task are reset.
4. In the **Reset Performer Aliases** dialog box, click **Yes** to reset the existing performers for the activity or click **Discard** to cancel the process.

5.1.13 Debugging a process

1. If you debug a process that is configured with an endpoint or uses any of the object types as a package or application parameter, deploy the application before you begin debugging the process.
2. Open the process in the process model editor.
3. Select an activity to add a breakpoint and click `Toggle Breakpoint on selected activity.`
4. Click **Debug Process**.
5. In the **Debugger Login** dialog box, select a design-time environment. In the **Environment Details** area, the system populates all the fields as configured in the selected design-time environment. Click **OK**.
6. If the process workflow begins with a manual activity:

    a. On the **Debug Process** dialog box, click **Start a workflow using manual initiate** activity.

    b. To add a package to the workflow, click **Attach** and select an object from the repository.

    You can double-click an attribute to edit the attribute value.

    c. To edit the value of a variable, expand the **Variables** node, right-click, and select **Edit Value**. You can define multiple values as comma-separated values.

    d. To add an on-demand attachment to the workflow, click **Attach** and select an object.

    e. Click **Start Workflow** and go to Step 9.

7. If the process workflow begins with an inbound activity:

    a. On the **Debug Process** dialog box, click **Start a workflow using inbound initiate** activity.

b. Click **Start Listeners** to begin the debug process.

The process debugger begins listening for the message start request specific to the protocol of the inbound activity. The workflow begins when the process receives the appropriate message. Go to Step 9.

8. On the **Process Debug** window, perform the following steps as necessary:

To	Perform the following
Execute a manual activity	Click **Task Manager** and acquire, reject, or finish a task.
Execute an automatic activity	1. Click **Task Manager** to view protocol-specific input and output messages and any exceptions that occur. You view error messages, in the **Exception** text box of the **Task Manager** tab. 2. To edit the properties of an activity, click **Activity Inspector**. 3. To complete an activity that has errors, which you cannot correct, click **Force Complete Task**.
Add packages, attachments, and edit package attributes and variables	1. Click **Process Data**. 2. Right-click a process data and select **Edit value**. 3. Click **Refresh** to get the latest values.
View log messages for the execution of the workflow, timers, triggers, and email notifications	Click **Console** to troubleshoot a process.
Send a trigger event	1. Click **Manage Workflow**. 2. Type a signal name in the **Send Event Workflow** field. 3. Click **Send**.
Move to the next activity	Click **Step over to next activity**.
View details of an activity and perform troubleshooting functions	Click **Step into the execution of the current activity**.
Move to the next activity marked with a breakpoint	Click **Continue to next breakpoint**.
Initiate the debug process again from the beginning of the flow	Click **Rerun Debug**.

9. After debugging the process, select each activity marked with a breakpoint and click `Toggle Breakpoint on selected activity` to remove a breakpoint.

5.1.14 Creating a version of the process

1. Open a process in the process model editor.
2. Right-click the root node of the process and select **Create version**.
3. In the **New Version (Process)** dialog box, specify information as described in the following table:

Field	Description
Label	Specify a label for the new version of the process.
Minor Version	Select the minor version for a process. For example, v1.1.
Major Version	Select the major version for a process. For example, v2.0.
Description	Type a description for the version of the process.

4. Click **Finish**.

   The versioned process inherits the business logic and processes.

5.2 Concepts

5.2.1 Defining a business process model

A process model captures the definition of a business process, enabling users to perform the process repeatedly. For example, a model for processing mortgage loans could route an application from the loan applicant to the loan originator to the loan processor. Because the process model is separate from its runtime instantiation, multiple processes based on the same process model can run concurrently.

A process model represents the business process through which a given object or set of object flows. It consists of the activities (performed by people or the system), the performer selection, data to route, and transition conditions based on business data.

Each time you create a process model, you make design decisions such as what types of process data to use, which activities to include, and how to structure the process model.

Defining a process model includes the following tasks:

- Plan the process model.
- Create the process model.
- Set the process model properties.
  - Add process variables.

  - Add packages to include content such as business objects, content objects, and folders.
- Create endpoints to connect to external systems.
- Add manual or system activities to the process model editor.
- Connect the activities with flows.
- Define the properties of activities and flows, including data mapping.

5.2.2 Process model editor

The process model editor is the area where you design the process model. To define a business process, you can drag and drop predefined activities from the Activities palette onto the canvas area of the process model editor. You can connect the activities with flows and then define the properties of the activities and flows.

5.2.3 Processes

A process is an instance of a process model at runtime. Multiple processes based on the same model can run concurrently because the process model is separate from its runtime instantiation. You can use a ready-made process model or create your own.

A process consists of the following elements:

- **Activities:** Represent tasks within a business process. For example, these tasks can include sending or receiving a message, reviewing or approving content, and checking a file into a repository.
- **Flows:** Connect activities within the process and orchestrate the business logic.
- **Process data:** Refers to the different types of data that flow through a process. The two main types of process data that you can define in a process model are process variables and process packages.

5.2.4 Process data

Process data refers to the data that flows through the process at runtime. Process data consists of several types of data:

- **Packages:** Packages reference business object, content, and folder data. The package includes the object and its associated content and attributes. Packages are objects in the repository that persist after the process execution. Activities work on these packages and then pass them to other activities. For example, a process can contain a loan application within a package.
- **Process variables:** Process variables store transient data, which the system no longer needs when the process completes. For example, you can create process variables for part numbers and customer addresses.
- **Application parameters:** Application parameters contain default values defined at the application level. All processes within the application can access these values.

- **Execution data:** Execution data comes from the current process and work items, such as process creation date and workitem runtime state. The system discards this information when the process finishes.
- **Attachments:** Attachments are objects such as business objects and content objects that an end user attaches to a process or uncompleted work item. For example, a process manages engineering proposals under development. An end user can attach a research paper to support a proposal. Attachments can be added at any point in the process and can be removed when no longer needed. After an attachment is added, it is available to the performers of all subsequent activities.

Process data enables end users to see meaningful business data when viewing their list of tasks. For example, an end user can view the name of an applicant, the approval status of a request, and the loan amount. This information enables a task performer to work more efficiently on the tasks in their inbox.

Process packages, process variables, and application parameters are configurable. You define process packages and variables at the process level and use them in individual activities within a process. You configure application parameters at the application level, which enables application administrators to configure them at runtime.

5.2.5 Process variables

Process variables are instances of data that flow through your business process. For example, you can create a process variable `approved` as a Boolean value, and use it in a transition, to determine process flow. The value of `approved` can be updated in an upstream activity.

Process variables, which are created when the process is created, exist during the lifecycle of the process. If you configured the process variables to have default values, the system assigns these values. When the process completes, the system permanently removes the process variable values.

A process variable can be either a single value or a multivalue variable, and its data type must be specified as String, Integer, Float, Boolean, or Date-time.

5.2.6 Process packages

Packages are placeholders for the actual business object, content, and folder data that run through a process at runtime. Packages can be linked to the existing business objects, content objects, and folder objects listed in the Object Models navigator. The package references the object and its associated content and attributes. Content objects have content, but business objects do not. Once the package is linked, activities can use the content and attributes of the packages. For example, in a claims processing application, an approval process starts when a claim is created. The package is linked to a customer business object, which means it references the customer business object and its associated attributes. Any manual activities in the process can use that package to display customer information for the approver.

Packages can be configured in the properties of the process model and they are considered as process data. In an activity, you can:

- Reference an object and associate it with a package
- Update the attributes of a package
- Update the contents of a package

5.2.7 Limiting end-user process permissions

Process permissions determine the allowable process actions for an end user or group at runtime. For example, in order for an end user to work on a process task, the end user must have at least the Browse permission level.

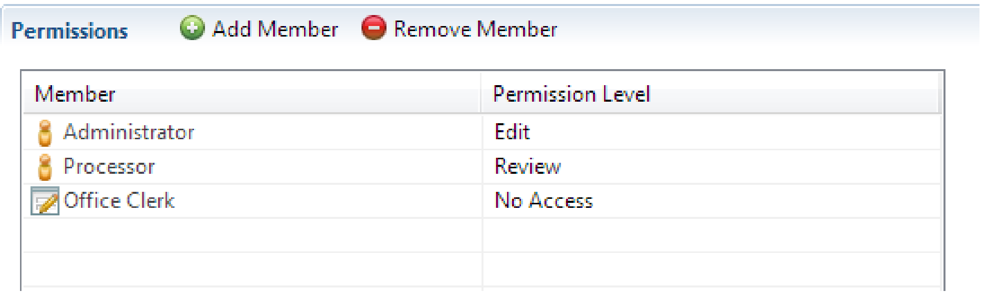
The following table describes the end-user process actions and the permission levels required to complete those actions:

Action	Minimum permission level
Work on a task for a process	Browse
Start a process	Review
Edit a process variable	Edit

At runtime, anyone can start a process and edit process variables by default. If your business processes require you to limit process permissions, you can create a permission set to restrict them. You can then specify this permission set in the process model properties.

You cannot limit the permissions of the workflow supervisor, the process creator, and the repository owner.

The following figure shows an example permission set that restricts process permissions:



Member	Permission Level
Administrator	Edit
Processor	Review
Office Clerk	No Access

In this example, the Administrator can start a process and edit process variables. The Processor can start a process and read process variables. The Office Clerk cannot start a process and cannot see the values of the process variables.

5.2.8 Stateless processes

A stateless process runs synchronously as a single transaction by session user. It is executed by the user logged into application and not by the performer configured on any activity. The synchronous transaction commits only at the end of the process. If an error occurs during the process, the entire process fails.

You can run a stateless process in the following ways:

- As an activity in a process
- From a timer activity

A stateless process uses process packages and process variables for both data inputs and data outputs. The process variables of a stateless process can have a combination of single value and multi-value variables.

You can use any combination of process packages and process variables for the data inputs and data outputs. You can define custom success or error messages, while the expression builder is not supported. By default, the system localizes these custom messages based on the Advanced Workflow locale and the availability of the Advanced Workflow language pack in the locale.

You can configure the visibility of a process package and process variable. For example, a variable that is used only for transitional logic can be configured as not visible.

In cases where process variable inputs are used to update or create an object, as a best practice, ensure that the process variable name matches the attribute system name. This is useful in showing meaningful error messages that are associated with the specific field having issues.

Benefits and limitations of stateless processes

The stateless process executes in memory and provides the following benefits:

- Improves performance since the stateless process executes in one thread and in one session.
- Enables you to use the outputs of the stateless process.

The stateless process returns user_login_name for the Execution Data.workflow.supervisor and Execution Data.task.performer attributes.

The following limitations apply to stateless processes:

- Stateless processes execute in one transaction and in one thread. While executing a process in one transaction leads to improved performance, it also increases the scope of the transaction when compared to a regular process. If another stateless or regular process modifies the same set of objects, there can be more version mismatch errors due to the increase in scope.

- A process model enabled to run in stateless mode can run either in a stateless mode or in a stateful mode at runtime. Processes run in a stateless mode do not generate business events related to the process. They do generate business events defined on the objects (business objects, folders, or content) that the process creates or updates.
- The executor of the stateless process runs all activities in the stateless process. It overrides the performer set in the Execution tab of the system activities.
- A stateless process cannot have manual activities, timer settings in activities, inbound Start From or Wait For activities, or trigger conditions (or events) in activities.
- Stateless processes do not support an input parameter named as `id` with empty or Null value.

Elevate session user permissions

Stateless process is frequently used as a data source to display results on the Advanced Workflow application interface. Usually, the logged in user is the owner of the session and in certain business scenarios, this user does not have sufficient permissions to perform some tasks in the stateless processes. Activities in a stateless process do not honor the performer setting in the **Execution** tab of the automatic activities as the entire process runs through the login user session.

To address such use cases, use the OOTB Java module, **ManageSessionUserPermissions** to elevate the session user permissions.

As a prerequisite, create dynamic groups and the users who are likely to log in and not have sufficient permissions should be the potential members for this dynamic group.

The **ManageSessionUserPermissions** Java module displays these methods:

- **addSessionUserToDynamicGroup**: Adds the session user to the dynamic group for this session. This allows the user to be treated as an active member of the dynamic group for the session until either the stateless process is terminated or the user is explicitly removed.
- **removeSessionUserFromDynamicGroup**: Removes the session user from the dynamic group for this session and hence the user is no longer treated as an active member of the dynamic group for this session.

Notes

- **ManageSessionUserPermissions** Java module should not be used in processes running on Java Method Server. When processes run on Java Method Server, they may get a shared session. Elevating permissions on shared sessions leads to security issues. For other use cases, stateful processes can be used.
- As a best practice, create a custom dynamic group instead of using OOTB OpenText Documentum CM Dynamic groups.

5.2.9 Message correlation

When the system receives an inbound message, it has to be able to route it to the correct process to handle the incoming data. The system uses correlation identifiers and correlation sets that contain unique data to match the incoming response to the original request.

For example, in a purchasing process, an activity sends a JMS message to a supplier requesting shipping information for an item on a purchase order. The message specifies the vendor ID number and the purchase order number, which the system uses to match the message to the process. The system of the supplier replies later with a shipping status message, which includes the vendor ID number and the purchase order number. When these unique data attributes are mapped to corresponding correlation set variables, the system can match the request to the response and continue the process.

In Wait For activities, you can use a correlation ID, a correlation set, or both. If a correlation ID is not available, the system looks for a correlation set mapping to match the incoming message to the process.

5.2.10 Correlation identifiers

Correlation identifiers are unique data values used to match incoming messages to the correct process. The system creates a unique Correlation ID attribute to identify each process. When a Wait For activity receives this identifier in a message, it can match the message to the appropriate process without using a correlation set.

The following table describes where to find the Correlation ID in the incoming message:

Activity	Correlation ID location
Wait for Email	Email header
Wait for FTP	File or directory name
Wait for HTTP	Query-string parameter
Wait for JMS	`correlationid` parameter of the JMS header
Wait for Web Service	SOAP header
Wait for Database	A column of data returned to the activity
Wait for Repository	An object attribute returned to the activity

5.2.11 Correlation sets

Correlation sets are sets of process variables whose values uniquely identify an incoming message for a particular process.

Correlation sets are defined at the process model level and then used within activities. For example, a correlation set named Banking Transactions consists of process variables for transaction number and bank routing number. If the process receives banking information within HTTP requests, a Wait for HTTP activity could receive the request. When configuring the activity, you map the incoming message attributes to the corresponding process variables in the correlation set. At runtime, the system uses the correlation set to route the incoming messages to the correct process.

You can only select one correlation set for mapping within an activity. Values for data in the correlation set must exist or be set before the process reaches the activity that uses the correlation set.

A correlation set can include more than one process variable and a process can use multiple correlation sets across multiple activities.

5.2.12 Process Debugger

You can test the design of a process model, detect, and diagnose errors before you deploy the process to a production environment. Debugging the process saves you time because you can resolve problems during the design phase of an end-user application, instead of troubleshooting problems at runtime.

When you debug a process, you can add a breakpoint to any activity in the process. The breakpoint allows you to interact with an activity in the process. The process debugger stops at each breakpoint so that you can verify or modify the properties of a process before executing it. For example, a breakpoint allows you to modify process data, which allows you to finish, reject, or complete the activity before executing a manual activity. The breakpoint also displays input, output, and exception messages associated with an automatic activity. You can add breakpoints for any manual or automatic activities.

You should debug the process model in a local environment. While debugging the process, the system does not create actual work items or queue items within a repository. However, the system saves the changes made to the objects referenced as a package or an attachment during the debug session and overwrites the existing attribute data. The system triggers all business events associated with such objects while you debug the process. The system also creates any objects needed for automatic activities within the process such as renditions or attachments created through mapping rules.

You must stop the bps while debugging a process having Start from or Wait for activities. When you deploy any application, the bps running in the application server starts the listener for the Start from and Wait for activities. All the messages may be consumed by the listener running in the bps and not by the listener running in the debugger.

The process debugger has these limitations:

- If the performer is set based on conditions, the performer is not resolved from the process debugger. This happens because the email notification does not go through in the process debugger, even though the performer is successfully resolved and the execution is completed.
- The the timer expires in process debugger after the configured time lapses, even if the current day is marked as a holiday in a business calendar and the calendar is used in the timer. The business calendar configuration in the timer is not supported in the process debugger.
- While debugging a process, the type fragment attributes and related objects are not shown for packages under **Process Data** tab in the process debugger.

5.2.13 Process versioning

A version is a copy of an existing process and is different from the original process.

You can create new versions of a process, so that multiple versions of the same process can coexist at runtime. When you create a version of a process, the process variables or packages of the process are versioned with the version of the process. At runtime, when you create an instance of a process, you can choose any version of the process to be instantiated. For example, consider that you have designed a loan model to process customer loans. Now, the government has introduced a new regulation that impacts your loan model. To change the process flow in the loan model, you can create a version of the existing process and add a new activity that complies with the newly introduced regulation, while retaining the existing process data and business logic intact. By using this mechanism, you do not need to stop or shut down any instances of the original process.

You can modify a process after the system has created a version of the process.

Use process versioning with caution to avoid creating inconsistencies in process data. It is recommended you create process versions while deploying a process to the production environment because creating different versions of a process can result in process data getting out of sync.

Note: As a best practice, it is recommended to change the process only by process versioning. Otherwise, the system breaks the contact between the process and running instance and this may lead to failure of running instances.

Chapter 6

Working with process activities

6.1 Tasks: Manual activities

6.1.1 Configuring manual activity tasks

Use this procedure to configure the manual and work queue tasks that processes create at runtime.

1. Drag and drop a manual activity (**Manual task** activity or **Work queue task** activity) onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Task** tab, configure the task as described in the following table:

Field	Description
Subject	Type a name for the task. To include process data, select a parameter for the subject and click **OK**. When the task is created at runtime, the process data is replaced with the value for the process data.
Instructions	Type instructions for the task performer. To include process data, select a parameter for the task instructions and click **OK**. When the task is created at runtime, the process data is replaced with the value for the process data.
Priority	Select a priority for the task. For **Manual task** activities, processors can use this priority to prioritize their tasks. For **Work queue task** activities, the system evaluates the priority for distributing tasks to end users. Queue managers can also use this priority to distribute tasks. To associate a custom priority module with a work queue task, select **Dynamic**.
Priority Module	If you select **Dynamic** as the priority for a work queue task, select the priority module to use for priority and aging logic.

4. Click **OK**.

6.1.2 Configuring the performers of a manual activity

1. Drag and drop the **Manual task** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Performer** tab, in the **Choose Performers** area, choose whether to select performers from a list or based on a condition.
4. To select a performer from a list, in the **Performer List** area, click **Add**.

   a. To select an individual performer, in the **Select Performer** dialog box, select the individual performer from the list and click **Finish**.

   b. To select other performers, “Determining which task performer to use” on page 151 provides additional details and links to performer selection procedures.

   c. To order the performers in the performer list for task distribution, click **Move Up** and **Move Down**.
5. To define conditions for a performer list, in the **Conditional Performers** area, click **Edit**.

   a. In the **Edit Rule** dialog box, in the **If** condition area, click **Select** to access the expression editor to add an If condition.

   b. In the **Then Performers List** area, click **Add** to add the performers that apply to that rule. “Determining which task performer to use” on page 151 provides additional details and links to performer selection procedures. Performer from Previous Activity is not available to use as a conditional performer.

   c. To order the performers in the performer list for task distribution, click **Move Up** and **Move Down**.

   d. Click **OK**.

   e. Add additional rules as required.
6. In the **Processed by** area, select how tasks are sent to the performers in the performer list as described in the following table:

Field	Description
Individual users and group member at the same time	Sends one task to each user and every user in each group.
Individual users and the first group member to accept the task, at the same time	Sends one task to each user and one task to each group. The group receives the task and it is assigned to the first member of this group who acquires the task.
Individual user or group member with fewest tasks	Sends one task to the performer who has the fewest pending tasks.

Field	Description
Individual users and the first group member to accept the task, in sequence	Sends one task to the first individual user listed in the performer list or the first group member to accept the task. The task is sent in the order of the list from top to bottom. After this task completes, another task is sent to the second individual user or the first group member to accept the task in the list. If you select this option, you have the additional option to select whether to allow performers to reject a task. The task returns to the performer who preceded them in the sequence.

7. In the **Task Options** area, select how to handle tasks as described in the following table:

Field	Description
Allow user to delegate task	If you select this option, select how to distribute the task if there is a problem with automatic delegation.
Allow performer to add additional performers to this task	This option enables the performer to add additional performers to complete the same task.
Performer sign-off is required	The activity requires a sign-off before it is complete. This option includes adding an electronic signature in the task.
Allow sequential task performers to reject to previous performer	If you select multiple sequential task performers, specify whether these performers can reject the task and return it to the previous performer in the sequence.

8. Click **OK**.

   At runtime, a manual task is created and assigned to users in the performer list as defined in the manual activity.

6.1.3 Selecting the performer of last activity to perform a manual activity

1. In the **Select Performer** dialog box, select **Performer of Last Activity** and click **Finish**.
2. Click **Move Up** and **Move Down** to order the performers for task distribution.
3. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.4 Selecting users from application parameters to perform a manual activity

1. In the **Select Performer** dialog box, select **User from Parameter** and click **Next**.
2. Use the **Search** field to search for users (application parameters of the User type) from the available parameters.
3. Select users in the list of available parameters and click the right arrow button to move them to the **Selected Performers** list.
4. Click **Finish**.
5. Click **Move Up** and **Move Down** to order the performers for task distribution.
6. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.5 Selecting a user from alias set to perform a manual activity

1. In the **Select Performer** dialog box, select **User from Alias Set** and click **Next**.
2. In the **Select a user from Alias Set** dialog box, select an alias from the alias set.
3. To select an alias from the alias set, select an alias set node in the upper area and select an alias from the lower area.
4. Click **Finish**.
5. Click **Move Up** and **Move Down** to order the performers for task distribution.
6. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.6 Selecting a user from process data to perform a manual activity

1. In the **Select Performer** dialog box, select **User from Process Data** and click **Next**.
2. In the **Select a user from process data** dialog box, select a process data string attribute from a package or select a process variable:
   - To select an attribute from a package, expand the **Packages** node in the upper area, select a package, and select an attribute from the lower area.
   - To select a process variable, select the **Variables** node in the upper area and select a variable from the lower area.

   Note: The first step activity (the activity that follows the Initiate activity) cannot have performers resolved from package attributes.
3. Click **Finish**.
4. Click **Move Up** and **Move Down** to order the performers for task distribution.
5. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.7 Selecting an application role to perform a manual activity

1. In the **Select Performer** dialog box, select **Application Role** and click **Next**.
2. Use the **Search** field to search for a role from the available application roles.
3. Select the application role in the available list and click the right arrow button to move the role to the **Selected Performers** list.
4. Click **Finish**.
5. Click **Move Up** and **Move Down** to order the performers for task distribution.
6. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.8 Selecting a group from application parameters to perform a manual activity

1. In the **Select Performer** dialog box, select **Group from Parameter** and click **Next**.
2. Use the **Search** field to search for a group (an application parameter of the Group type) from the available parameters.
3. Select the group in the list of available parameters and click the right arrow button to move the group to the **Selected Performers** list.
4. Click **Finish**.
5. Click **Move Up** and **Move Down** to order the performers for task distribution.
6. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.9 Selecting a group from process data to perform a manual activity

1. In the **Select Performer** dialog box, select **Group from Process Data** and click **Next**.
2. In the **Select a group from process data** dialog box, select a process data string attribute from a package or select a process variable:
   - To select an attribute from a package, expand the **Packages** node in the upper area, select a package, and select an attribute from the lower area.
   - To select a process variable, select the **Variables** node in the upper area and select a variable from the lower area.
3. Click **Finish**.
4. Click **Move Up** and **Move Down** to order the performers for task distribution.
5. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.10 Selecting a group from alias set to perform a manual activity

1. In the **Select Performer** dialog box, select **Group from Alias Set** and click **Next**.
2. In the **Select a group from Alias Set** dialog box, select an alias from the alias set.
3. To select an alias set for the group, select an alias set node in the upper area and select an alias from the lower area.
4. Click **Finish**.
5. Click **Move Up** and **Move Down** to order the performers for task distribution.
6. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.11 Selecting the performer from previous activity for a manual activity

1. In the **Select Performer** dialog box, select **Performer from Previous Activity** and click **Next**.
2. Configure the Performer from Previous Activity as described in the following table:

Field	Description
Activity	Select a previous activity from the drop-down list.
Multiple Performer Options	If the previous activity has multiple performers, select which set of performers to use for task distribution.

3. Click **Finish**.
4. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.12 Selecting the performer from previous activity to select next performer at runtime for a manual activity

1. In the **Select Performer** dialog box, select **Previous Activity Performer to Choose at Runtime** and click **Next**.

2. Configure the activity and user type for the performer as described in the following table:

Field	Description
Activity	Select a previous activity from the drop-down list.
User Type	Select a user type that performer can select at runtime from the drop-down list. You can select: • **Any User**: Performer can select any user at runtime. • **Any Group**: Performer can select any group of users at runtime • **Specific Group**: Performer can select a user from a specific group type at runtime. – **Group from Parameters**: You select application parameters to define placeholders for groups that are determined at runtime. A group selected from an application parameter can be filled in by any group in an environment. – **Group from Alias Set**: You select aliases from alias sets as placeholders for groups. The system retrieves the group from the alias value using alias set object in the repository.

3. Click **Finish**.

4. Configure the options for tasks and click **OK**. “Configuring the performers of a manual activity” on page 84 provides configuration details.

6.1.13 Defining work queues and skills in the repository

Before deploying your application, ensure the work queues and skills used in your process model are also defined in the repository. The *OpenText Documentum Server Administration and Configuration Guide* provides additional information.

1. In Documentum Administrator, define work queues with the same names as the work queues used in Advanced Workflow.

2. Within a process, if you configure skills in a **Work queue task** activity in Advanced Workflow, do the following in Documentum Administrator:

    a. Define a matching filter with the same skill names and values as the skills used in the **Work queue task** activity in Advanced Workflow.

    b. Associate the matching filter to the work queue.

    c. Add the skills for work assignment matching to the processor profiles.

    The application deploys even if the skill set names do not match or if they are not in the repository. In this case, the work queue behaves as if there are no skills enabled. Any of the work queue members can pull the task.

    If a process passes an incorrect skill value (or no value) to the work queue task skill set at runtime, work queue members cannot pull the task. Only the work queue administrator or the work queue manager can view and assign the task to the queue member.

6.1.14 Configuring a work queue for a task

1. Drag and drop the **Work queue task** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector** and click the **Work queue** tab.

3. In the **Work queue chosen** area, choose whether to select an individual work queue or to select a work queue based upon a condition.

4. If you select **Work queue**, click **Select**.

    "Selecting a work queue from a parameter" on page 92 and "Selecting a work queue from process data" on page 93 provide additional information on selecting a work queue and configuring work queue skill sets.

5. If you select **Based on a condition**, in the **Conditional Work Queue** area, click **Edit**.

    a. In the **Edit Rule** dialog box, in the **If condition** area, click **Select** to access the expression editor to add an If condition.

    b. In the **Work queue** area, click **Edit** to add the work queue that applies to that rule.

"Selecting a work queue from a parameter" on page 92 and "Selecting a work queue from process data" on page 93 provide additional information on selecting a work queue and configuring work queue skill sets.

c. Add additional rules as required, selecting only one work queue per rule.

6. In the **Task Options** area, select how to handle the task as described in the following table:

Field	Description
Allow performer to add additional performers to this task	This option enables the performer to add additional performers to complete the same task.
Performer sign-off is required	The activity requires a sign-off before it is complete. This option includes adding an electronic signature in the task.

7. Click **OK**.

   At runtime, a task is created and associated with the work queue as defined in the **Work queue task** activity.

6.1.15 Selecting a work queue from a parameter

1. In the **Select Work queue** dialog box, select **Work queue from a Parameter** and click **Next**.

2. In the **Select a work queue from a parameter** dialog box, use the **Search** field to search for a work queue (application parameters of the Work queue type) from the available parameters.

3. Select a parameter and click **Next**.

4. To enable skill-set matching, in the **Define work queue skills** dialog box, add the skills needed by the work queue performers:

   a. Click **Add Skill**.

   b. In the **Add Skill** dialog box, type the skill name, select the skill data type, and click **OK**.

   c. Select the skill and click **Select** to add a skill set value.

   d. Add additional skills as required.

5. Click **Finish**.

6. Configure the task options and click **OK**. "Configuring a work queue for a task" on page 91 provides configuration details.

6.1.16 Selecting a work queue from process data

1. In the **Select Work queue** dialog box, select **Work queue from Process Data** and click **Next**.
2. In the **Select a work queue from process data** dialog box, select the string attribute from a package or the process variable that contains the work queue name:

   - To select an attribute from a package, expand the **Packages** node in the upper area, select a package, and select an attribute from the lower area.
   - To select a process variable, select the **Variables** node in the upper area and select a variable from the lower area.

3. Click **Next**.
4. To enable skill-set matching, in the **Define work queue skills** dialog box, add the skills needed by the work queue performers:

   a. Click **Add Skill**.

   b. In the **Add Skill** dialog box, type the skill name, select the skill data type, and click **OK**.

   c. Select the skill and click **Select** to add a skill set value.

   d. Add additional skills as required.

5. Click **Finish**.
6. Configure the task options and click **OK**. “Configuring a work queue for a task” on page 91 provides configuration details.

6.1.17 Associating a priority module with a work queue task

1. Double-click a **Work queue task** activity to open the **Activity Inspector**.
2. On the **Task** tab, in the **Priority** field, select **Dynamic**.
3. In the **Priority Module** field, select the priority module to use for priority and aging logic.

6.2 Tasks: System activities

6.2.1 Configuring the performer and execution of a system activity

1. Drag and drop a system activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Execution** tab, configure the **Execution Settings** as described in the following table:

Field	Description
Perform as	Select one performer for the activity. This performer runs the activity execution logic and needs privileges for this activity. Click **Select**, select a performer in the **Select Performer** dialog box, and click **Finish**. “Selecting a user from application parameters to perform a system activity” on page 95 provides information on how to select a user from application parameters. **Best Practice:** To improve the performance of system activities in your process, make one user (or a small set of users) the performer of all the system activities. For example, by defining a user, such as auto_executor, and making that user the performer for all system activities, the runtime execution significantly increases for these activities.
Timeout (sec)	Type the timeout in seconds for the duration of the execution. **Best practice:** When configuring automatic activities, increase the Timeout (sec) field to greater than 60 seconds.
Priority	Select a priority for the system task. By default, the system executes system tasks in the order created. The system uses this priority only if enabled on the Documentum CM Server. The *OpenText Documentum Advanced Workflow Deployment Guide* provides instructions on enabling priority levels for system activities.

Field	Description
Enable troubleshooting logging	Select to save an execution log of the running service. The execution results appear in the `/Temp/BPM/Method Exec Results` repository folder. Activity failures appear in the `/Temp/BPM/Method Exceptions` repository folder.

4. Configure the **Exception Handling** as described in the following table:

Field	Description
Retry on timeout or exception	Select to rerun the activity when a failure occurs due to an exception or an execution taking longer than the specified timeout.
Maximum tries	Type the number of times to try before failing.
Interval (min)	Type the number of minutes to wait before rerunning the activity.
On failure	Select how to proceed when the exception cannot be resolved.

5. Click **OK**.

6.2.2 Selecting a user from application parameters to perform a system activity

1. In the **Select Performer** dialog box, select **User from Parameter** and click **Next**.

2. Use the **Search** field to search for a user (application parameter of the User type) from the available parameters.

3. Select a user in the list of available parameters and click **Finish**.

4. Configure the **Execution Settings** and **Exception Handling** for the system activity and click **OK**. “Configuring the performer and execution of a system activity” on page 94 provides configuration details.

6.2.3 Configuring a Create activity

1. Drag and drop the **Create** activity onto the canvas.

2. Double-click the activity to open **Activity Inspector**.

3. In the **Create Object** area of the **Create** tab, specify how to create the object as described in the following table:

Field	Description
From a model	Creates an object from a model. Browse and select the model of a business object, shareable business object, lightweight business object, folder, content, shareable content, or lightweight content and click **Finish**.
From a copy	Copies an object in the repository including the object attributes. If the object is a folder, it also copies subfolders and the objects contained within those subfolders. Creating an object from a copy does not copy relationships and permission sets. This is not supported for the Cabinet object model.
Propagate ACL to children	Determines whether the objects contained in the new folder inherit the access control list (ACL) of that folder. This option applies only to folders created from a copy.

4. In the **Get Destination Folder by** area, select how to identify the destination folder as described in the following table:

Field	Description
Use default location	Identifies the destination folder using the **Default location** property of the model.
Folder ID	Identifies the destination folder using a folder ID. This option requires the folder ID for the input mapping.
Folder Path	Identifies the destination folder using a path in the repository. This option requires the folder path for the input mapping. Use the following format for the folder path: `/<<cabinet name>>/<<folder name>>/<<folder name>>` This activity creates the objects in the last folder of the path.

5. Click **Next**.

6. On the **Input Message Mapping** screen, map process data (Source) to the object properties (Destination) and click **Next**.

7. On the **Output Message Mapping** screen, map the ID of the newly created object (Source) to the process data (Destination) and click **OK**.

6.2.4 Configuring a Create ACL activity

Use the create ACL activity template to create a OpenText Documentum CM ACL object from **Permission Set** and apply it to objects within a process instance. The permission set must have been created to include placeholders or aliases for users and groups. The system uses the **Create ACL** activity template to supply values for these placeholders and aliases based on process data and creates a new ACL object from the permission set after resolving the aliases.

1. Drag and drop the **Create ACL** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Create ACL** tab, select an existing Permission Set and click **Next**.

   Note: Only the Template class of the Permission Set is supported.

4. On the **Input Message Mapping** screen, map process data to the attributes of the ACL template:

Attributes	Description
ACL Name	Maps to the name of the ACL object.
ACL Domain	Maps to the owner of the ACL object.
ACL Class	Maps to the ACL class. Valid values are: • *0 (Private ACL)*: Are available only to the user who create them. Only the owner of a private ACL or a user with Superuser privileges can modify or delete the ACL. • *3 (Public ACL)*: Are available for anyone in the repository to use. Only the owner of a public ACL or a user with Sysadmin or Superuser privileges can modify or delete the ACL.

   Note: After you have configured the activity template with a permission set template, if new aliases are added to the permission set, ensure that values are mapped to the aliases in the Activity Inspector.

5. In the **Output Message Mapping** screen, map ACL parameters with the process data.

Attributes	Description
Object ID	Maps to the object ID of the newly created ACL object.
ACL Domain	Maps to the owner of the newly created ACL object.

Attributes	Description
ACL Name	Maps to the name of the newly created ACL object.

Note: Aliases belonging to categories such as permission set, cabinet path, and folder path are not supported for this action in the data mapper.

6. Click **OK**.

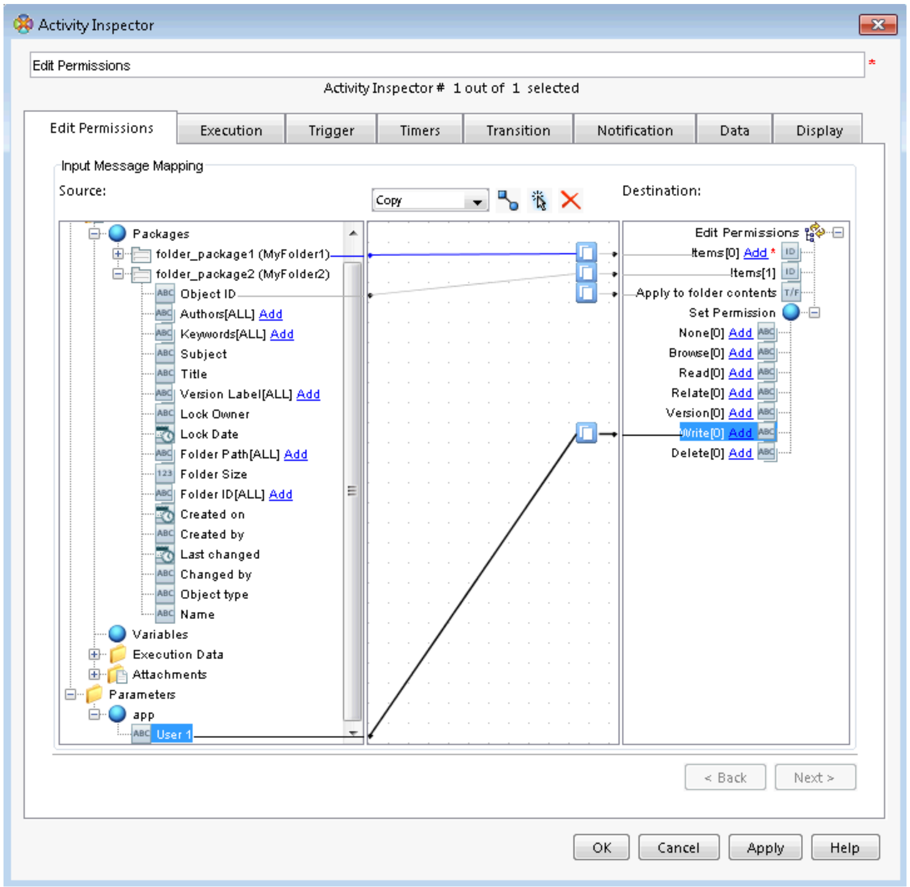
6.2.5 Configuring an Edit Permissions activity

1. Drag and drop the **Edit Permissions** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Edit Permissions** tab, map the process data (Source) to the edit permissions inputs (Destination):

    a. In the **Source** area of the **Input Message Mapping** screen, map the Object ID of the objects that you want to update with permissions to the **Items** attribute in the **Destination** area. To map multiple objects, click **Add** on the **Items** attribute to add additional index values for the items attribute (for example, Items[1], Items[2], and Items[3]). Map one Object ID to each index value.

    b. To apply permission updates to all objects contained within a folder, map a Boolean attribute with a value of True to the **Apply to folder contents** attribute in the **Destination** area.

    c. Map user or group names in the **Source** area to a specific permission level, such as Write, in the **Destination** area. Expand the **Set Permission** node to view the permission levels.



4. Click **OK**.

6.2.6 Configuring a Move activity

1. Drag and drop the **Move** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. In the **Get Destination Folder by** area of the **Move** tab, select how to identify the destination folder as described in the following table:

Field	Description
Folder ID	Identifies the destination folder using a folder ID. This option requires the folder ID for the input mapping.

Field	Description
Folder Path	Identifies the destination folder using a path in the repository. This option requires the folder path for the input mapping. Use the following format for the folder path: `/<<cabinet name>>/<<folder name>>/<<folder name>>` This activity moves or creates a shortcut in the last folder of the path.

4. In the **Operation** area, specify what to do with the set of objects as described in the following table:

Field	Description
Move	Moves the set of objects to the destination folder.
Create shortcut	Creates a shortcut in the destination folder to the set of objects.

5. On the **Input Message Mapping** screen, map process data (Source) to the parameters of the Move operation (Destination) and click **OK**.

6.2.7 Configuring a Set Process Data activity

1. Drag and drop the **Set Process Data** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Set Process Data** tab, map process data from the upstream activity to process data for the **Set Process Data** activity.
4. Click **OK**.

6.2.8 Configuring a Call Database Procedure activity

1. Drag and drop the **Call Database Procedure** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Call Database Procedure** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an endpoint for the activity.

Field	Description
Procedure name	Type the procedure name. The name of a stored database procedure (also known as a database function) is expressed in the following forms: • *fully qualified: <<catalog-name>>.<<schema-name>>.<<procedure-name>>* • *partly qualified: <<schema-name>>.<<procedure-name>>* • *unqualified: <<procedure-name>>* If you do not know the procedure name, click **Search** to search across all schemas and packages in the database.

4. Click **Next** to display the process data mapping tool.

   If the stored procedure takes no arguments, go to Step 6.

5. On the **Input Message Mapping** screen, map process data to procedure arguments and click **Next**.

   If the procedure has multiple input arguments and you map only one, the others are set to NULL or as defined in the procedure.

6. On the **Output Message Mapping** screen, map procedure return values to process data and click **OK**.

   If the procedure has multiple return values and you map only one, the others are set to NULL or as defined in the procedure.

6.2.9 Configuring a read from Database activity

1. Drag and drop the**Read from Database** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Read from Database** tab, select or create an endpoint for the activity.

4. In the **Query To Run** area, type a Select statement.

   For example:

   ```
   select * from dbo.TableadvwfProcess where Status='process'
   ```

   To create a query using parameters, type a question mark (?) where you want to substitute the value.

   The following example query contains parameters:

   ```
   select * from dbo.TableadvwfProcess where Status=?
   ```

   You can only add parameters on the *where* clause. The statement can include any number of parameters, each represented by a question mark. Do not include quotes around the question mark.

5. If the Select statement includes parameters, click **Set Parameter Type**.

    a. In the **Set Parameters Type** dialog box, select the data types for all parameters.

    b. Click **OK**.

6. Click **Validate Query** to verify that the query is valid SQL.

7. Click **Next** to display the process data mapping tool.

    If the statement has no parameters, go to Step 10.

8. On the **Input Message Mapping** screen, map process data to statement parameters.

9. Click **Next**.

10. On the **Output Message Mapping** screen, map statement return values to process data.

11. Click **OK**.

6.2.10 Configuring a start from Database activity

1. Drag and drop the **Start from Database** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Start from Database** tab, select or create an endpoint for the activity.

4. In the **Read Query** area, type a read query to run on the database and click **Validate Query** to verify that the query is valid SQL.

    For example:

    ```
    select * from dbo.TableadvwfProcess where Status='process'
    ```

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of simultaneous threads that the activity can use to process data returned by the read query.
Interval (sec)	Type the number of seconds to wait before running a new read query.
Maximum rows	Type the maximum number of result rows that the activity processes.

6. In the **Post-Processing Write Statement** area, specify a post-processing action to avoid retrieving the same data twice.

    Type a write statement that updates the data so that it no longer satisfies the query. The write statement can be a SQL update or delete statement.

To create a query using parameters, type a question mark (?) where you want to substitute the value.

The following example query contains parameters:

```
update dbo.TableadvwfProcess set Status='done' where Columnint=?
```

You can only add parameters on the *where* clause. The update statement can include any number of parameters, each represented by a question mark. Do not include quotes around the question mark.

Ensure that the row can be uniquely identified during post-processing so that the system only processes a single row.

7. If the write statement includes parameters, click **Set Parameters Type** to map the parameters to aliases, which are mapped to actual columns in the database.

   a. In the **Set Parameter Fields** dialog box, for each substitution parameter, select a database column name/alias. The database columns listed are the columns retrieved from the database using the read query.

   b. Click **OK**.

8. Click **Next** to display the process data mapping tool.

9. On the **Input Message Mapping** screen, map the retrieved database row columns (Source) to the process data (Destination).

10. Click **OK**.

6.2.11 Configuring a wait for Database activity

1. Drag and drop the **Wait for Database** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Wait for Database** tab, select or create an endpoint for the activity.

4. In the **Read Query** area, type a read query to run on the database and click **Validate Query** to verify that the query is valid SQL.

   For example:

   ```
   select * from dbo.TableadvwfProcess where Status='process'
   ```

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of simultaneous threads that the activity can use to process data returned by the read query.
Interval (sec)	Type the number of seconds to wait before running a new read query.

Field	Description
Maximum rows	Type the maximum number of result rows that the activity processes.

6. To use correlation identifiers, in the **Correlation ID** drop-down list, select a column that contains a correlation identifier. The system uses the correlation identifier to match the retrieved data to the workflow.

   You can use correlation identifiers, correlation sets, or both. Step 12 describes how to configure correlation sets.

7. In the **Post-Processing Write Statement** area, specify a post-processing action to avoid retrieving the same data twice.

   Type a write statement that updates the data so that it no longer satisfies the query. The write statement can be a SQL update or delete statement.

   To create a query using parameters, type a question mark (?) where you want to substitute the value.

   The following example query contains parameters:

   ```
   update dbo.TableadvwfProcess set Status='processed' where Columnint=?
   ```

   You can only add parameters on the *where* clause. The update statement can include any number of parameters, each represented by a question mark. Do not include quotes around the question mark.

   Ensure that the row can be uniquely identified during post-processing so that the system only processes a single row.

8. If the write statement includes parameters, click **Set Parameters Type** to map the parameters to aliases, which are mapped to actual columns in the database.

   a. In the **Set Parameter Fields** dialog box, for each substitution parameter, select a database column name/alias. The database columns listed are the columns retrieved from the database using the read query.

   b. Click **OK**.

9. Click **Next** to display the process data mapping tool.

10. On the **Input Message Mapping** screen, map the retrieved database row columns (Source) to the process data (Destination).

11. To use correlation sets, click **Next**. Otherwise go to step 13.

12. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map the retrieved database columns (Source) to correlation set variables.

13. Click **OK**.

6.2.12 Configuring a write to Database activity

1. Drag and drop the **Write to Database** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Write to Database** tab, select or create an endpoint for the activity.
4. In the **Query To Run** area, type an Insert, Update, or Delete statement.

   To create a query using parameters, type a question mark (?) where you want to substitute the value.

   The following example query contains parameters:

   ```
   update dbo.TableadvwfProcess set Status='done' where ID=?
   ```

   You can only add parameters on the *where* clause. The statement can include any number of parameters, each represented by a question mark. Do not include quotes around the question mark.
5. To set the parameters, click **Set Parameter Type**.

   a. In the **Set Parameters Type** dialog box, select the data types for all parameters.

   b. Click **OK**.
6. Click **Next** to display the process data mapping tool.

   If the statement has no parameters, go to Step 9.
7. On the **Input Message Mapping** screen, map process data to statement parameters.
8. Click **Next**.
9. On the **Output Message Mapping** screen, map statement return values to process data.
10. Click **OK**.

6.2.13 Configuring a read from Repository activity

1. Drag and drop the**Read from Repository** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Read from Repository** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create a repository endpoint for the activity.

Field	Description
Use application repository at runtime	Select to use the local application repository instead of the endpoint at runtime to improve performance. The endpoint and the application repository must be the same.

4. In the **Query To Run** area, type a DQL statement.

   For example:

   ```
   SELECT object_name, r_modify_date FROM dm_document WHERE object_name like
   '%loan%'
   ```

   To create a query using parameters, type a question mark (?) where you want to substitute the value.

   ```
   SELECT object_name, r_modify_date FROM dm_document WHERE object_name like
   ?
   ```

   You can only add parameters on the *where* clause. The statement can include any number of parameters, each represented by a question mark. Do not include quotes around the question mark.

   The following example query contains parameters:

   ```
   SELECT object_name, r_modify_date FROM dm_document WHERE object_name =?
   ```

   The following example query contains the **IN** clause and parameters:

   ```
   SELECT object_name, r_modify_date FROM dm_document WHERE object_name IN
   (?)
   ```

   To substitute multiple values of a parameter, go to Step 5.

5. If the statement includes parameters, click **Set Parameter Type**.

   a. In the **Set Parameters Type** dialog box, select the parameter types for all parameters.

   b. Select the **Multivalue** check box to substitute multiple values for a parameter.

   c. Click **OK**.

6. Click **Validate Query** to verify that the query is valid DQL.

7. Click **Next** to display the process data mapping tool.

   If the statement has no parameters, go to Step 9.

8. On the **Input Message Mapping** screen, map process data to statement parameters, and click **Next**.

9. On the **Output Message Mapping** screen, map statement return values to process data, and click **OK**.

6.2.14 Configuring a start from Repository activity

1. Drag and drop the **Start from Repository** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Start from Repository** tab, specify repository connection information as described in the following table:

Field	Description
Endpoint	Select or create a repository endpoint for the activity.
Use application repository at runtime	Select to use the local application repository instead of the endpoint at runtime to improve performance. The endpoint and the application repository must be the same.

4. In the **Read Query** area, type a read query to run on the repository.

   This read query is a DQL qualification and not a full query. For example:

   ```
   dm_folder where object_name='x'
   ```

5. Click **Validate Query** to verify that the query is valid DQL.
6. Specify processing information as described in the following table:

Field	Description
Interval (sec)	Type the number of seconds to wait before running a new read query.
Number of threads	Type the number of simultaneous threads that the activity can use to process objects returned by the read query.

7. Specify a post-processing action to avoid retrieving the same object twice. Post-processing actions are described in the following table:

Field	Description
Delete object	After the query has returned an object, delete the object from the repository.
Move object to folder	After the query has returned an object, move the object within the repository.
Execute update statement	After the query has returned an object, update its attributes within the repository. Type a statement that updates attributes so that the object no longer satisfies the query.

8. Click **Next** to display the process data mapping tool.

9. On the **Input Message Mapping** screen, map the object retrieved from the read query (Source) to the process data (Destination).

10. Click **OK**.

6.2.15 Configuring a wait for Repository activity

1. Drag and drop the **Wait for Repository** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Wait for Repository** tab, specify repository connection information as described in the following table:

Field	Description
Endpoint	Select or create a repository endpoint for the activity.
Use application repository at runtime	Select to use the local application repository instead of the endpoint at runtime to improve performance. The endpoint and the application repository must be the same.

4. In the **Read Query** area, type a read query to run on the repository.

   This read query is a DQL qualification and not a full query. For example:

   ```
   dm_folder where object_name='x'
   ```

5. Click **Validate Query** to verify that the query is valid DQL.

6. Specify processing information as described in the following table:

Field	Description
Interval (sec)	Type the number of seconds to wait before running a new read query.
Number of threads	Type the number of simultaneous threads that the activity can use to process objects returned by the read query.

7. To use correlation identifiers, in the **Correlation ID** drop-down list, select an attribute that contains a correlation identifier. The system uses the correlation identifier to match the retrieved object to the workflow.

   You can use correlation identifiers, correlation sets, or both. Step 11 describes how to configure Correlation sets.

8. Specify a post-processing action to avoid retrieving the same object twice. Post-processing actions are described in the following table:

Field	Description
Delete object	After the query has returned an object, delete the object from the repository.
Move object to folder	After the query has returned an object, move the object within the repository.
Execute update statement	After the query has returned an object, update its attributes within the repository. Type a statement that updates attributes so that the object no longer satisfies the query.

9. Click **Next** to display the process data mapping tool.
10. On the **Input Message Mapping** screen, map the object retrieved from the read query (Source) to the process data (Destination).
11. If you are using correlation sets for message correlation, click **Next**. Otherwise go to Step 13.
12. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map attributes of the retrieved object (Source) to correlation set variables.
13. Click **OK**.

6.2.16 Configuring a write to Repository activity

1. Drag and drop the**Write to Repository** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Write to Repository** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create a repository endpoint for the activity.
Use application repository at runtime	Select to use the local application repository instead of the endpoint at runtime to improve performance. The endpoint and the application repository must be the same.

4. In the **Query To Run** area, type a DQL statement.

   To create a query using parameters, type a question mark (?) where you want to substitute the value.

   You can only add parameters on the *where* clause. The statement can include any number of parameters, each represented by a question mark. Do not include quotes around the question mark.

The following example query contains parameters:

```
UPDATE dm_document SET title=? WHERE r_object_id=?
```

The following example query contains the **IN** clause and parameters:

```
UPDATE dm_document OBJECT SET title=? WHERE r_object_id IN (?)
```

To substitute multiple values of a parameter, go to Step 5.

5. To set the parameters, click **Set Parameter Type**.

    a. In the **Set Parameters Type** dialog box, select the parameter types for all parameters.

    b. Select the **Multivalue** check box to substitute multiple values for a parameter.

    c. Click **OK**.

6. Click **Next** to display the process data mapping tool.

    If the statement has no parameters, go to Step 8.

7. On the **Input Message Mapping** screen, map process data to statement parameters, and click **Next**.

8. On the **Output Message Mapping** screen, map statement return values to process data, and click **OK**.

In the Write to Repository activity template, the system support create, update, and delete object DQL queries.

6.2.17 Configuring a Send Email activity

For Secure Sockets Layer (SSL) or Transport Layer Security (TLS) encryption, you must have an SSL certificate in your environment. The *OpenText Documentum Advanced Workflow Deployment Guide* provides instructions on importing SSL certificates to support encrypted connections for activities.

1. Drag and drop the **Send Email** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Send Email** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an email endpoint for the activity.
Email Template	Select or create an email template for the activity.

Field	Description
Email Preview	View the Subject and Body of the email message defined in the email template to verify that it is correct.

4. Click **Next** to display the process data mapping tool.

5. On the **Input Message Mapping** screen, for the input message, map the process data (Source) to the email message attributes (Destination).

   a. Map the **To** recipient node in the **Destination** area.

   b. Map all placeholder nodes that reside under the main **Subject** and **Body** nodes.

   c. Use the attributes listed in the following table to create mappings that set values in email headers:

Attribute name	Email header name	Description	Action
Importance	X-Priority, Importance, Priority, X-MSMail-Priority	Status set by the sender that indicates the relative importance of the message. Valid values are: (1) High, (3) Normal, and (5) Low.	Set
Keywords	Keywords	Words or phrases that can be used when searching or sorting email.	Set
Expiry-Date	Expiry-Date	Time that the message is no longer valid.	Set
Sensitivity	Sensitivity	Guidance about disclosing message contents. Valid values are: `Personal`, `Private`, or `Company-Confidential`.	Set

The **Send Email** activity also enables you to add attributes to the **Additional Headers** node in the email message.

6. Map a process package or attachment to the email attachment. If you have multiple email attachments, you can click **Add** on the Attachment node to add more attachments.

7. Click **Next**.

8. On the **Output Message Mapping** screen, for the output message, map email message attributes (Source) to process data (Destination).

9. Click **OK**.

Note: If you are configuring Send Email activity with Microsoft Graph endpoint, set the following Java argument in the `\\Documentum\tomcat\bin\catalina.bat` file.

```
set "JAVA_OPTS=%JAVA_OPTS% -Dhttps.proxyHost=ban-prox01-l001.otxlab.net -
Dhttps.proxyPort=3128"
```

6.2.18 Configuring a Start from Email activity

The email client must have encoding set to UTF-8 while sending and reading messages that use multi-byte characters.

1. Drag and drop the **Start from Email** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Start from Email** tab, select or create an incoming email endpoint for the activity.

4. In the **Folder to monitor** field, type or select the folder, for example INBOX, on the email server to monitor for unread email messages.

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of threads used to process email messages.
Interval (sec)	Type the number of seconds that pass before the email listener checks for new messages from the email server.
Preferred content-type	Type or select the preferred content type to retrieve the data of the email message if the email message contains multiple content types. For example, an email message sent with multiple content types can contain both HTML and plain text data.

6. Specify a post-processing action to avoid processing the same email twice. Post-processing actions are described in the following table:

Field	Description
Delete email	Deletes the email after it is processed.

Field	Description
Mark email as read	Marks the email as read after it is processed.
Move email to	Moves the email after it is processed. Type or select the name of the archive folder to move the email (for example, `Archive`).

7. Click **Next** to display the process data mapping tool.

8. On the **Input Message Mapping** screen, map the message data (Source) to process data (Destination).

   Use the attributes listed in the following table to create mappings that retrieve values from email headers:

Attribute name	Email header name	Description	Action
Received-Date		Date that the system received the email message.	Retrieve
Importance	X-Priority, Importance, Priority, X-MSMail-Priority	Status set by the sender that indicates the relative importance of the message. Valid values are: (`1`) High, (`3`) Normal, and (`5`) Low. If this value is not mapped, the system uses the default value of 3, which is Normal.	Retrieve
Keywords	Keywords	Words or phrases that can be used when searching or sorting email.	Retrieve
Expiry-Date	Expiry-Date	Time that the message is no longer valid.	Retrieve
Sensitivity	Sensitivity	Guidance about disclosing message contents. Valid values are: `Personal`, `Private`, or `Company-Confidential`.	Retrieve

   If these attributes are not mapped, their values are not retrieved from the incoming message.

The **Start from Email** activity also enables you to add attributes to the **Additional Headers** node in the email message.

9. The **message** node enables you to map the incoming email message as an **eml** format file, consisting of content, metadata, and attachment, to a process package or attachment. While mapping the same in the process data (Destination), configure the format of the process package or attachment as eml.

10. Map the email attachment to a process package or attachment.

11. Click **OK**.

Note: If you are configuring Start for Email activity with Microsoft Graph endpoint, set the following Java argument in the `\\Documentum\tomcat\bin\catalina.bat` file.

```
set "JAVA_OPTS=%JAVA_OPTS% -Dhttps.proxyHost=ban-prox01-l001.otxlab.net -
Dhttps.proxyPort=3128"
```

6.2.19 Configuring a Wait for Email activity

The email client must have encoding set to UTF-8 while sending and reading messages that use multi-byte characters.

1. Drag and drop the **Wait for Email** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Wait for Email** tab, select or create an incoming email endpoint for the activity.

4. In the **Folder to monitor** field, type or select the folder, for example INBOX, on the email server to monitor for unread email messages.

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of threads used to process email messages.
Interval (sec)	Type the number of seconds that pass before the email listener checks for new messages from the email server.
Preferred content-type	Type or select the preferred content type to retrieve the data of the email message if the email message contains multiple content types. For example, an email message sent with multiple content types can contain both HTML and plain text data.

6. In the **Correlation header name** field, type a header name that includes a correlation ID to use for correlating the email to the correct instance of the waiting process.

You can use correlation identifiers, correlation sets, or both. Step 12 describes how to configure correlation sets.

7. Specify a post-processing action to avoid processing the same email twice. Post-processing actions are described in the following table:

Field	Description
Delete email	Deletes the email after it is processed.
Mark email as read	Marks the email as read after it is processed.
Move email to	Moves the email after it is processed. Type or select the name of the archive folder to move the email (for example, `Archive`).

8. Click **Next** to display the process data mapping tool.

9. On the **Input Message Mapping** screen, map the message data (Source) to process data (Destination).

   Use the attributes listed in the following table to create mappings that retrieve values from email headers:

Attribute name	Email header name	Description	Action
Received-Date		Date that the system received the email message.	Retrieve
Importance	X-Priority, Importance, Priority, X-MSMail-Priority	Status set by the sender that indicates the relative importance of the message. Valid values are: (1) High, (3) Normal, and (5) Low. If this value is not mapped, the system uses the default value of 3, which is Normal.	Retrieve
Keywords	Keywords	Words or phrases that can be used when searching or sorting email.	Retrieve
Expiry-Date	Expiry-Date	Time that the message is no longer valid.	Retrieve

Attribute name	Email header name	Description	Action
Sensitivity	Sensitivity	Guidance about disclosing message contents. Valid values are: `Personal`, `Private`, or `Company-Confidential`.	Retrieve

If these attributes are not mapped, their values are not retrieved from the incoming message.

The **Wait for Email** activity also enables you to add attributes to the **Additional Headers** node in the email message.

10. The **message** node enables you to map the incoming email message as an **eml** format file, consisting of content, metadata, and attachment, to a process package or attachment. While mapping the same in the process data (Destination), configure the format of the process package or attachment as EML file.

11. Map the email attachment to a process package or attachment.

12. If you are using correlation sets, click **Next**. Otherwise, go to Step 13.

13. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map attributes of the message (Source) to corresponding correlation set variables.

14. Click **OK**.

Note: If you are configuring Wait for Email activity with Microsoft Graph endpoint, ensure that you have copied the `msgraph.properties` file to the BPM and BPS installation locations:

- ..\bpm\WEB-INF\classes\msgraph.properties
- ..\bps\WEB-INF\classes\msgraph.properties

6.2.20 Configuring a Call Process activity

1. Drag and drop the **Call Process** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Call Process** tab, configure the activity as described in the following table:

Field	Description
Process Name	Click **Select**, select a subprocess for the activity to call, and click **Finish**.
Start current version of process	Select whether the activity calls current version or configured version of the subprocess at runtime. If the checbox is selected, the activity always call the current version of the subprocess at runtime, even though the process is configured with an older version of the process. If not selected, the activity calls the configured version of the subprocess at runtime.
Invoke Setting	Select whether the parent process waits for the invoked subprocess to complete before going to the next activity in the parent process. Select a setting for the child process.
Start Process Setting	Select whether the subprocess starts at the Initiate activity or at another activity in the process.

4. Click **Next**.

5. On the **Input Message Mapping** screen, map packages, variables, attachments, and execution data from the parent process (Source) to the process data of the subprocess (Destination), and click **Next**.

6. On the **Output Message Mapping** screen, map the available data from the subprocess to the parent process:

   - If the activity continues to the next activity before the subprocess completes, map the process ID of the subprocess (Source) to the parent process (Destination).
   - If the activity waits for the subprocess to complete, map the completed process data from the subprocess (Source) to the parent process (Destination).

7. Click **OK**.

6.2.21 Configuring a Call Stateless process activity

1. Drag and drop the **Call Stateless Process** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. In the **Process** field of the **Call Stateless Process** tab, click **Select**.
4. Select a stateless process model for the activity to call and click **Finish**. "Configuring general information for your process model" on page 65 details how to designate a process model to run in stateless mode.
5. Click **Next**.
6. On the **Input Message Mapping** screen, map the process data (packages and variables) from the parent process (Source) to the process data in the stateless process (Destination), and click **Next**.
7. On the **Output Message Mapping** screen, map the completed process data from the stateless process (Source) to the parent process (Destination).
8. Click **OK**.

6.2.22 Configuring a Signal Parent process activity

Use this activity in a process to signal a waiting parent process to continue.

1. Drag and drop the **Signal Parent Process** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Input Message Mapping** screen, map process data from the process (Source) to the **Signal name** attribute in the parent process (Destination).

   The Signal name is the name of the trigger signal. An activity in the parent process waits for this signal. The Signal name must match the name specified on the **Trigger** tab of the waiting activity.
4. Click **OK**.

6.2.23 Configuring a Download from FTP activity

1. Drag and drop the**Download from FTP** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Download from FTP** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an endpoint for the activity.

Field	Description
Message Type	Select the type of message to download, choosing either plain file, compressed (archived into a zip folder), or directory (organized in subfolders).

4. Click **Next** to display the process data mapping tool.
5. On the **Input Message Mapping** screen, specify a pattern or an exact file or directory name to fetch from the FTP server.
6. Click **Next**.
7. On the **Output Message Mapping** screen, map attributes of the downloaded file, compressed file, or directory to process data.
8. Click **OK**.

6.2.24 Configuring a Start from FTP activity

1. Drag and drop the **Start from FTP** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Start from FTP** tab, select or create an endpoint for the activity.
4. Specify monitoring information as described in the following table:

Field	Description
Message type	Select the type of message to receive, choosing either plain file, compressed file (archived into a zip folder), or directory (organized in subfolders).
Include name pattern	Specify a pattern or an exact file or directory name to fetch from the FTP server.
Exclude name pattern	Specify a pattern or an exact file or directory name to avoid fetching from the FTP server.

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of simultaneous threads that the activity can use to download content from the FTP server.
Interval (sec)	Type the number of seconds to wait before the activity checks for new incoming message files or folders from the FTP server.

6. Specify a post-processing action to avoid fetching the same file or directory twice. Post-processing actions are described in the following table:

Field	Description
Delete file/directory	Deletes the message file or directory from the FTP server after it has been fetched.
Move file/directory to	Moves the message file or directory after it has been fetched. Type a destination location on the FTP server (for example, `/archive`).

7. Click **Next** to display the process data mapping tool.

8. On the **Input Message Mapping** screen, map attributes of the FTP message (Source) to process data (Destination).

9. Click **OK**.

6.2.25 Configuring an Upload to FTP activity

1. Drag and drop the **Upload to FTP** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Upload to FTP** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an endpoint for the activity.
Message Type	Select the type of message to send, choosing either plain file, compressed file (archived into a zip folder), or directory (organized in subfolders).
Options	Select **Overwrite file/folder if exists** to overwrite files or folders on the endpoint system. Clear this option to create new files or folders instead of overwriting.

4. Click **Next** to display the process data mapping tool.

5. On the **Input Message Mapping** screen, map process data to file, compressed file or directory attributes.

   Zip folders and subfolders are named at runtime. Zip folder names are based on the compressed file name you configure. Subfolder names are based on the directory name you configure.

   Files whose **Content Type** attribute is mapped to string text are transferred as ASCII text. All other files are transferred in binary mode.

6. Click **OK**.

6.2.26 Configuring a Wait for FTP activity

1. Drag and drop the **Wait for FTP** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Wait for FTP** tab, select or create an endpoint for the activity.
4. Specify monitoring information as described in the following table:

Field	Description
Message type	Select the type of message to receive, choosing either plain file, compressed file (archived into a zip folder), or directory (organized in subfolders).
Include name pattern	Specify a pattern or an exact file or directory name to fetch from the FTP server.
Exclude name pattern	Specify a pattern or an exact file or directory name to avoid fetching from the FTP server.

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of simultaneous threads that the activity can use to download content from the FTP server.
Interval (sec)	Type the number of seconds to wait before the activity checks for new incoming message files or folders from the FTP server.

6. To use correlation identifiers for message correlation, in the **Correlation pattern** field, type a correlation pattern to show how to determine the correlation ID from the filename. The FTP listener uses this pattern to correlate the message with the workflow.

   Use `$id$` as a placeholder for the correlation identifier. Placeholder values are alphanumeric characters without spaces.

   For example, the file *sam_956d77f734b6d7fb97771be.xml* can be represented as *sam_id.xml*. The placeholder value `$id$` represents the correlation ID *956d77f734b6d7fb97771be*.

   You can use correlation identifiers, correlation sets, or both. Step 11 describes how to configure correlation sets.

7. Specify a post-processing action to avoid fetching the same file or directory twice. Post-processing actions are described in the following table:

Field	Description
Delete file/directory	Deletes the message file or directory from the FTP server after it has been fetched.
Move file/directory to	Moves the message file or directory after it has been fetched. Type a destination location on the FTP server (for example, `/archive`).

8. Click **Next** to display the process data mapping tool.
9. On the **Input Message Mapping** screen, map attributes of the FTP message (Source) to process data (Destination).
10. If you are using correlation sets, click **Next**. Otherwise, go to Step 12.
11. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map attributes of the FTP message (Source) to corresponding correlation set variables.
12. Click **OK**.

6.2.27 Configuring a Send HTTP or RESTful Request activity

1. Drag and drop the **Send HTTP/RESTful Request** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Send HTTP/RESTful Request** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an endpoint for the activity.
Request Method	Select one of following HTTP methods for the request: GET, POST, PUT, and DELETE.
Connection Timeout (mins)	Displays the number of minutes allowed to lapse before the system terminates an idle connection.
Allow multipart request body	Allows you to associate multiple attachments with the HTTP request. This option is valid for the POST and PUT requests only.

Field	Description
Handle HTTP Errors	Allows you to choose if the activity has to be completed or paused in case of HTTP service failure. Usually, the HTTP response codes not between 200 and 206 are considered as failures. So, if the HTTP response code is not within this range of 200 and 206, the check box value determines if the activity has to be paused or completed.

4. Click **Next** to display the process data mapping tool.

5. On the **Input Message Mapping** screen, map process data to the request message, and click **Next**.

6. On the **Output Message Mapping** screen, map the response message to process data.

   If required, add attributes or elements to the process data to complete the mappings.

Note: You must select a HTTP Service Endpoint (URL, username, and password) to configure the Send HTTP/RESTful Request activity.

The endpoint (URL, username and password) for the HTTP/REST call can configured in the following two ways:

- Using HTTP endpoint when it is constant for all requests.
- Using URL, username and password nodes in the data mapper when the endpoint changes for every request.

Either the endpoint or data mapping should be present to complete the Send HTTP/RESTful Request activity configuration. Both endpoint and data mapping can be configured. If both are configured, then during runtime the data mappings to URL, username, and password nodes takes precedence over the nodes specified in the endpoint. There is no validation for URL specified in the endpoint except for **Test Connection** used on the endpoint.

6.2.28 Configuring a Start from HTTP activity

The browser must have encoding set to UTF-8 when sending the request to the inbound HTTP listener when multi-byte characters are used.

1. Drag and drop the **Start from HTTP** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Start from HTTP** tab, specify request information as described in the following table:

Field	Description
URL Suffix	Type a suffix for the local URL to be monitored for incoming HTTP requests.
HTTP Method	Select an HTTP method for the incoming request. If you expect the request URL to include query-string parameters, map these to process data in Step 7. If you expect the request to include a message body, select **With Attachments**. For example, a content or image file could be the message body. If the message body is an XML document that needs validation, select **Validate XML Documents**.

4. Specify authentication information as described in the following table:

Field	Description
Require Authentication	Select this option to require authentication credentials from the web server sending the HTTP request.
Username	Type a username to authenticate the request.
Password	Type a password to authenticate the request.

5. To create an HTML response message to return from the request, select **Use custom response template**.

   a. In the **Response Template** text box, type a template for the HTTP response to send when an HTTP request has been received.

   b. Include a substitution variable for each item of process data to insert in the template at runtime. Map the substitution variables to process data in Step 9. These variables have the same syntax as email templates.

   In the following example, a message is sent to indicate that a specific purchase order was processed successfully:

   ```
   Purchase Order $orderno successfully processed.
   ```

   In this example, *orderno* is a placeholder that appears in the **Output Message Mapping** screen and it can be mapped to a package or a process variable. An actual order number replaces the substitution variable from the response data mapping at runtime:

   ```
   Purchase Order 0896523 successfully processed.
   ```

6. Click **Next** to display the process data mapping tool.

7. On the **Input Message Mapping** screen, map the input request message (Source) to process data (Destination).
8. Click **Next**.
9. On the **Output Message Mapping** screen, map the process data (Source) to the response message (Destination).
10. Click **OK**.

6.2.29 Configuring a Wait for HTTP activity

The browser must have encoding set to UTF-8 when sending the request to the inbound HTTP listener when multi-byte characters are used.

1. Drag and drop the **Wait for HTTP** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Wait for HTTP** tab. specify request information as described in the following table:

Field	Description
URL Suffix	Type a suffix for the local URL to be monitored for incoming HTTP requests.
HTTP Method	Select an HTTP method for the incoming request. If you expect the request URL to include query-string parameters, map these to process data in Step 8. If you expect the request to include a message body, select **With Attachments**. For example, a content or image file could be the message body. If the message body is an XML document that needs validation, select **Validate XML Documents**.

4. If you want to use a correlation ID, in the **Correlation header name** field, type the name of the header that contains the correlation ID. The HTTP listener uses the correlation ID to match the incoming HTTP request with the process.

   You can use correlation identifiers, correlation sets, or both. Step 12 describes how to configure correlation sets.

5. Specify authentication information as described in the following table:

Field	Description
Require Authentication	Select this option to require authentication credentials from the web server sending the HTTP request.
Username	Type a username to authenticate the request.
Password	Type a password to authenticate the request.

6. To create an HTML response message to return from the request, select **Use custom response template**.

   a. In the **Response Template** text box, type a template for the HTTP response to send when an HTTP request has been received.

   b. Include a substitution variable for each item of process data to insert in the template at runtime. Map the substitution variables to process data in Step 10. These variables have the same syntax as email templates.

   In the following example, a message is sent to indicate that an account was processed successfully:

   ```
   Account number $acctno successfully processed.
   ```

   In this example, *acctno* is a placeholder that appears in the **Output Message Mapping** screen and it can be mapped to a package or a process variable. An actual account number replaces the substitution variable from the response data mapping at runtime:

   ```
   Account number 0453323 successfully processed.
   ```

7. Click **Next** to display the process data mapping tool.

8. On the **Input Message Mapping** screen, map the input request message (Source) to process data (Destination).

9. Click **Next**.

10. On the **Output Message Mapping** screen, map the process data (Source) to the response message (Destination).

11. If you are using correlation sets, click **Next**.

12. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map the attributes of the HTTP message (Source) to corresponding correlation set variables.

13. Click **OK**.

6.2.30 Configuring an Execute Java method activity

1. Drag and drop the **Execute Java Method** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Execute Java Method** tab, configure the activity as described in the following table:

Field	Description
Method Name	Enter the object_name of the dm_method that is created in the docbase. **Notes** - To run a method using this activity, the attribute *a_special_app* must be set. *a_special_app* is a *dm_sysobject* attribute reserved for use by OpenText Documentum CM products. This attribute must have the value Workflow. The value of this attribute is not validated. - If the custom methods are dependent on libraries present in `bpm.war/WEB-INF/lib`, include the custom logic and its dependency JARs as part of a custom BOF module and invoke the BOF module from the custom method.

4. Click **OK**.

6.2.31 Configuring an Execute Java Service activity

1. Drag and drop the **Execute Java Service** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Execute Java Service** tab, configure the activity as described in the following table:

Field	Description
Java service	Select the Java service you want the process to execute.

Field	Description
Method	Select one of the methods that the Java service makes available. Each method corresponds to one function that the Java service can perform.

4. Click **Next** to display the process data mapping tool.
5. On the **Input Message Mapping** screen, map the process data to method arguments.
6. Click **Next**.
7. On the **Output Message Mapping** screen, map the response to process data.
8. Click **OK**.

6.2.32 Example of extending process functionality

An example of extending default process functionality is the ability to generate a unique ID for use in a Customer profile. To obtain this functionality, a developer can write custom Java code and integrate it into your application as a Java module.

In this example, the Java service name is **JavaGetUniqID** and the method is **generateUniqueID(String Prefix)**. This method has an input of type String and its value is used as a prefix for the unique ID. The method output (the unique ID) is of type String. The method generates a unique ID and adds the prefix to that unique ID and returns it.

```
import java.util.UUID;

public class UniqueIDGenerator
{
  public static String generateUniqueId(String prefix)
  {
     int hashCode = UUID.randomUUID().hashCode();
      return prefix + Integer.toString(hashCode);
  }
}
```

1. Create a business object named `Customer` with the attributes: `first name`, `last name` and `CustomerID`. All of them are of type String. CustomerID represents a unique ID.
2. Create a process model.
3. Create a package called `CustomerPkg` of type Customer.
4. Add an **Execute Java Service** activity to your process model.
5. When you configure the activity:
   a. Select the **JavaGetUniqID** service.
   b. Select the **generateUniqueID(String Prefix)** method.

c. Map process data to the method input. Select the **last name** attribute of CustomerPkg as the data to map to **Prefix** on the generateUniqueID.

d. Map the output of the method to the **CustomerID** attribute on CustomerPkg.

At runtime, when the process is started with a Customer instance, the system populates the **CustomerID** attribute with a unique ID generated by the Java service.

6.2.33 Configuring a Send JMS Message activity

1. Drag and drop the**Send JMS Message** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Send JMS Message** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an endpoint for the activity.
Message Type	Select a type of JMS message to send. For example, there are types for text, binary, and streaming data.

4. Click **Next** to display the process data mapping tool.
5. On the **Input Message Mapping** screen, map process data to message data.
6. Click **Next**.
7. On the **Output Message Mapping** screen, map message attributes to process data.
8. Click **OK**.

   If necessary to complete the mappings, add attributes or elements to data.

6.2.34 Configuring a Start from JMS activity

1. Drag and drop the **Start from JMS** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Start from JMS** tab, select or create an endpoint for the activity.
4. Specify monitoring information as described in the following table:

Field	Description
Message selector	Type a JMS message selector to filter messages appropriately for your process.

Field	Description
Message type	Select a message body format (message type) defined by the JMS API.

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of threads that the activity can use simultaneously to download content from the JMS server.
Validate XML	Select this option to validate JMS messages using an XML schema.

6. Click **Next** to display the process data mapping tool.

7. On the **Input Message Mapping** screen, map message data (Source) to process data (Destination). If you are using an XML schema on the JMS server to validate messages, add the XML schema to the **Data** node in the message context tree. “Using an XML schema to represent a content structure” on page 181 provides step-by-step instructions.

8. Click **OK**.

6.2.35 Configuring a Wait for JMS activity

1. Drag and drop the **Wait for JMS** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Wait for JMS** tab, select or create an endpoint for the activity.

4. Specify monitoring information as described in the following table:

Field	Description
Message selector	Type a JMS message selector to filter messages appropriately for your process.
Message type	Select a message body format (message type) defined by the JMS API.

5. Specify processing information as described in the following table:

Field	Description
Number of threads	Type the number of threads that the activity can use simultaneously to download content from the JMS server.
Validate XML	Select this option to validate JMS messages using an XML schema.

6. Click **Next** to display the process data mapping tool.

7. On the **Input Message Mapping** screen, map message data (Source) to process data (Destination). If you are using an XML schema on the JMS server to validate messages, add the XML schema to the **Data** node in the message context tree. "Using an XML schema to represent a content structure" on page 181 provides step-by-step instructions.

8. If you are using correlation sets for message correlation, click **Next**. Otherwise go to Step 10.

9. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map message data (Source) to correlation set variables.

10. Click **OK**.

6.2.36 Configuring a Call Web Service activity

1. Drag and drop the **Call Web Service** activity onto the canvas.

2. Double-click the activity to open the **Activity Inspector**.

3. On the **Call Web Service** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an endpoint for the activity.
Port Type	Select the port for the operation you intend to choose.
Operation	Select an operation supported by the web service you intend to call.
Use XML-binary Optimization Packaging	Select to enable serialization of XML Information Sets that contain binary data.

4. Click **Next** to display the process data mapping tool.

   If the operation has no input parameters, go to Step 7.

5. On the **Input Message Mapping** screen, map process data to operation input parameters.

6. Click **Next**.

7. On the **Output Message Mapping** screen, map output values to process data.

   If the operation has no output parameters, go to Step 8.

8. Click **OK**.

6.2.37 Registering the repository for chained services

This procedure uses the ContextRegistryService provided by DFS to register the repository. The *OpenText Documentum Foundation Services Development Guide* provides more information on the ContextRegistryService.

1. Create a web service endpoint using the fully qualified URL path to the WSDL.

   For example:

   ```
   http://<hostname>:<port>/services/core/runtime/ContextRegistryService?wsdl
   ```

   The WSDL can also exist as a local file.

2. Double-click the **Call Web Service** activity to open the Activity Inspector.

3. On the **Call Web Service** tab, select the web service endpoint that you created in step 1.

4. In the **Port Type** field, select **ContextRegistryServicePort**.

5. In the **Operation** field, select **register**.

6. Click **Next**.

7. On the **Input Message Mapping** screen, create the following mappings:

   - Map **RepositoryName** to:

     ```
     SOAPEnvelope/SOAPBody/parameters/register/context/RepositoryIdentity
     [0]/@repositoryName
     ```

   - Map **Username** to:

     ```
     SOAPEnvelope/SOAPBody/parameters/register/context/RepositoryIdentity
     [0]/@userName
     ```

   - Map **Password** to:

     ```
     SOAPEnvelope/SOAPBody/parameters/register/context/RepositoryIdentity
     [0]/@password
     ```

   - Map **Domain** to:

     ```
     SOAPEnvelope/SOAPBody/parameters/register/context/RepositoryIdentity
     [0]/@domain
     ```

   RepositoryIdentity is a repeating node. Multiple identities can be registered to a single token.

8. Click **Next**.

9. On the **Output Message Mapping** screen, the generated token is available in

   ```
   SOAPEnvelope/SOAPBody/parameters/registerResponse/return
   ```

6.2.38 Unregistering the repository for chained services

This procedure uses the ContextRegistryService provided by DFS to unregister the repository. The *OpenText Documentum Foundation Services Development Guide* provides more information on the ContextRegistryService.

1. Create a web service endpoint using the fully qualified URL path to the WSDL.

   For example:

   ```
   http://<hostname>:<port>/services/core/runtime/ContextRegistryService?wsdl
   ```

   The WSDL can also exist as a local file.

2. Double-click the **Call Web Service** activity to open the Activity Inspector.

3. On the **Call Web Service** tab, select the web service endpoint that you created in step 1.

4. In the **Port Type** field, select **ContextRegistryServicePort**.

5. In the **Operation** field, select **unregister**.

6. Click **Next**.

7. On the **Input Message Mapping** screen, add the token that you want to unregister into `SOAPEnvelope/SOAPBody/parameters/unregister/token`.

6.2.39 Invoking typical DFS services (single or chained)

1. Create a web service endpoint using the fully qualified URL path to the WSDL.

   For example:

   ```
   http://<hostname>:<port>/services/core/runtime/ContextRegistryService?wsdl
   ```

   The WSDL can also exist as a local file.

2. Double-click the **Call Web Service** activity to open the Activity Inspector.

3. On the **Call Web Service** tab, select the web service endpoint that you created in step 1.

4. Select the **Port Type** and **Operation** based on the values available in the WSDL file.

5. Click **Next**.

6. On the **Input Message Mapping** screen, add the following data mappings for authenticating any of the DFS services.

   Use the concatenate function to construct the string for mapping.

   The following mappings are only used to authenticate the DFS services. The *OpenText Documentum Foundation Services Development Guide* provides more details on DFS functionality.

   For a chained service, map the following:

```
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd"><wsse:BinarySecurityToken
QualificationValueType="http://schemas.emc.com/documentum#Resource
AccessToken" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd"
 wsu:Id="RAD">someToken</wsse:BinarySecurityToken></wsse:Security>
```

to the SOAPHeader

```
(SOAPEnvelope/SOAPHeader/SOAPHeaderElement[0])
```

For a single service, map the following:

```
    <ServiceContext
xmlns="http://context.core.datamodel.fs.documentum.emc.com/"
xmlns:ns2="http://properties.core.datamodel.fs.documentum.emc.com/"
xmlns:ns3="http://profiles.core.datamodel.fs.documentum.emc.com/">
<Identities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
repositoryName="docbaseName" password="docbasePassword"
userName="docbaseUsername"
xsi:type="RepositoryIdentity"/></ServiceContext>
```

to the SOAPHeader

```
(SOAPEnvelope/SOAPHeader/SOAPHeaderElement[0])
```

Use multiple <Identities/> when more than one repository is used for authentication.

6.2.40 Configuring a Start from Web Service activity

1. In the **Process Properties** of the process model, on the **Advanced** tab, configure the **Target Namespace URI**.

2. Drag and drop the **Start from Web Service** activity onto the canvas.

3. Double-click the activity to open the **Activity Inspector**.

4. On the **Start from Web Service** tab, specify web service input information as described in the following table:

Field	Description
Define based on	Select whether to define web service inputs from **Process data** or an **Existing web service**. If you select **Process data**, the system does not require you to map the data since the mappings are implicit.
WSDL URL	If you select **Existing web service**, type the URL for the Web Services Description Language (WSDL) used for the activity and click **Read** to retrieve the WSDL from the specified location.
Port type	If you select **Process data**, type the name for the WSDL port type. Otherwise, select a port type from the drop-down list.

Field	Description
Operation	If you select **Process data**, type a name for the WSDL operation. Otherwise, select an operation from the drop-down list.

5. Specify authentication information as described in the following table:

Field	Description
Enable WS-Security Authentication	Select this option to require authentication credentials from the web service sending the SOAP message.
Username	Type a username to authenticate the transmission.
Password	Type a password to authenticate the transmission.

Once enabled, all client requests to this activity must contain the same credentials in the request header (defined by the Web Services Security UsernameToken Profile) to obtain access to the activity or process:

```
<SOAPENV:header>
    <wsse:Security xmlns:wsse="{WS-Security Schema}">
        <wsse:UsernameToken>
            <wsse:Username>{username}</wsse:Username>
            <wsse:Password Type="wsse:PasswordText">{password}
</wsse:Password>
        </wsse:UsernameToken>
    </wsse:Security>
</SOAPENV:header>
```

6. Select **Enable MTOM** (Message Transmission and Optimization Mechanism) to enable the system to optimize any attachments while sending the response.

7. If you defined your web service inputs from **Process data**, go to Step 10. Otherwise, click **Next**.

8. On the **Input Message Mapping** screen, map the incoming SOAP message attributes (Source) to process data (Destination) and click **Next**.

   Map the content and file format for attachments. If you have multiple attachments, you can click **Add** on the Attachment node to add more attachments.

9. On the **Output Message Mapping** screen, map process data (Source) to the outbound SOAP response (Destination).

10. Click **OK**.

Once the application is deployed, Process Integrator creates the web service inbound listeners. The following are the valid URL formats for accessing the WSDL:

```
http://<hostname>:<port>/bps/webservice/<processID>?WSDL
http://<hostname>:<port>/bps/webservice/<processName>?WSDL
http://<hostname>:<port>/bps/webservice/<processName>/<version>?WSDL
```

6.2.41 Configuring a Wait for Web Service activity

1. In the **Process Properties** of the process model, on the **Advanced** tab, configure the **Target Namespace URI**.
2. Drag and drop the **Wait for Web Service** activity onto the canvas.
3. Double-click the activity to open the **Activity Inspector**.
4. On the **Wait for Web Service** tab, specify web service input information as described in the following table:

Field	Description
Define based on	Select whether to define web service inputs from **Process data** or an **Existing web service**. If you select **Process data**, the system does not require you to map the data and correlation set since the mappings are implicit.
WSDL URL	If you select **Existing web service**, type the URL for the Web Services Description Language (WSDL) used for the activity and click **Read** to retrieve the WSDL from the specified location.
Port type	If you select **Process data**, type the name for the WSDL port type. Otherwise, select a port type from the drop-down list.
Operation	If you select **Process data**, type a name for the WSDL operation. Otherwise, select an operation from the drop-down list.
Correlation Set	If you select **Process data**, you can select this option to use correlation sets for message correlation. If you select an **Existing web service**, you can configure correlation sets in Step 11. You can use correlation IDs, correlation sets, or both. Step 13 details how to add a correlation ID.

5. Specify authentication information as described in the following table:

Field	Description
Enable WS-Security Authentication	Select this option to require authentication credentials from the web service sending the SOAP message.
Username	Type a username to authenticate the transmission.

Field	Description
Password	Type a password to authenticate the transmission.

Once enabled, all client requests to this activity must contain the same credentials in the request header (defined by the Web Services Security UsernameToken Profile) to obtain access to the activity or process:

```
<SOAPENV:header>
    <wsse:Security xmlns:wsse="{WS-Security Schema}">
        <wsse:UsernameToken>
            <wsse:Username>{username}</wsse:Username>
            <wsse:Password Type="wsse:PasswordText">{password}
</wsse:Password>
        </wsse:UsernameToken>
    </wsse:Security>
</SOAPENV:header>
```

6. Select **Enable MTOM** (Message Transmission and Optimization Mechanism) to enable the system to optimize any attachments while sending the response.

7. If you defined your web service inputs from **Process data**, go to Step 12. Otherwise, click **Next** to display the process data mapping tool.

8. On the **Input Message Mapping** screen, map the incoming SOAP message attributes (Source) to process data (Destination) and click **Next**.

   Map the content and file format for attachments. If you have multiple attachments, you can click **Add** on the Attachment node to add more attachments.

9. On the **Output Message Mapping** screen, map process data (Source) to the outbound SOAP response (Destination).

10. If you are using correlation sets, click **Next**. Otherwise, go to Step 13.

11. On the **Correlation Set Mapping** screen, in the **Select set** field, select a correlation set from the drop-down list. Using the Copy function, map attributes of the incoming SOAP message (Source) to correlation set variables.

12. Click **OK**.

13. To enable message correlation using a correlation ID, add the correlation ID to the SOAP header in a format defined by the Web Services Addressing standard, as in the following example:

```
<SOAPENV:header>
    <wsa:RelatesTo xmlns:wsa="{WS-AddressingSchema}">{correlationid}
</wsa:RelatesTo>
</SOAPENV:header>
```

Once the application is deployed, Process Integrator creates the web service inbound listeners. The following are the valid URL formats for accessing the WSDL:

```
http://<hostname>:<port>/bps/webservice/<processID>?WSDL
http://<hostname>:<port>/bps/webservice/<processName>?WSDL
http://<hostname>:<port>/bps/webservice/<processName>/<version>?WSDL
```

6.2.42 Configuring a Publish Document activity

1. Drag and drop the **Publish Document** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **xPression Configuration** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an xPression server endpoint for the activity.
Category	Select an xPression document category.
Document	Select a content object that belongs to the category you selected.
Output profile	Select an output profile to publish the content.

4. Click **Next**.
5. On the **Input Message Mapping** screen, map the process data to the input document variables:

   - Map all variables that the xPression server requires to complete the publish service. At runtime, the variables are replaced with the process data in the generated document.
   - With xDesign documents, only variables that are used in the document appear in the mapper. Ensure that the primary key is mapped.
   - If necessary, add attributes or elements to some of the data to complete the mappings.

   Data validation is not supported in the Publish Document activity template. Ensure that all the runtime data is valid.

6. Click **Next**.
7. On the **Output Message Mapping** screen, map the output document to a package or an attachment when the document profile provides an output document.

   The following attributes must be mapped to process data:

   - **format:** This is the format type set in the output profile. For example, if the output profile is defined to publish a PDF file, this attribute must be mapped to the corresponding package or attachment format.
   - **data:** This is the generated document. It must be mapped to the corresponding attribute in either a package or an attachment.

8. Click **OK**.

6.2.43 Configuring an Advanced Publish Document activity

1. Drag and drop the **Advanced Publish Document** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **xPression Configuration** tab, configure the activity as described in the following table:

Field	Description
Endpoint	Select or create an xPression server endpoint for the activity.
Category	Select an xPression document category.
Document	Select a content object that belongs to the category you selected.
Output profile	Select an output profile to publish the content.

Field	Description
Customer Data XML	Type the XML that creates the mappings for constants and adds the variables for dynamic values. This XML contains customer information used by the xPression document to personalize the content during the publish process. The XML in this text box should consist of static text as well as variables for the dynamic content that you map. • Variables are prefixed with a dollar sign **$** (for example, $streetaddress). • Use two dollar signs **$$** to create multivalued variables. • To use a literal dollar sign in the text, use **\$** (for example, five hundred dollars would be expressed as **\$500.00**). Variables must be alphanumeric characters without spaces. The variable starts after the dollar sign **$** and ends with first non-alphanumeric character. Variables must be string types, so date and time values may have to be converted using the DateToString data mapper function. The date format that comes from the process data and the XSD date format that is expected by xPression server may be different. The system requires the following formats for XSD date and time conversions: • **date:** yyyy-MM-dd • **dateTime:** yyyy-MM-dd'T'HH:mm:ss.SSSZ • **time:** HH:mm:ss.SSS “Customer Data XML sample code” on page 165 provides an example. Data validation is not supported in the Advanced Publish Document activity template. Ensure that all the runtime data is valid.

4. Click **Next**.

5. On the **Input Message Mapping** screen, map the process data to the input document variables that were placeholder values in the XML:

   - Create mappings for variables provided in the **Customer Data XML** text box.

- If necessary, add attributes or elements to some of the data to complete the mappings.

The constant values that you add using the Customer Data XML text box do not appear in the data mapper.

6. Click **Next**.
7. On the **Output Message Mapping** screen, map the output document to a package or an attachment when the document profile provides an output document.

   The following attributes must be mapped to process data:

   - **format:** This is the format type set in the output profile. For example, if the output profile is defined to publish a PDF file, this attribute must be mapped to the corresponding package or attachment format.
   - **data:** This is the generated document. It must be mapped to the corresponding attribute in either a package or an attachment.
8. Click **OK**.

6.2.44 Configuring a Lifecycle activity

1. Drag and drop the **Lifecycle** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Lifecycle** tab, specify lifecycle information as described in the following table:

Field	Description
Lifecycle Operation	Select one of the following: • Promote: Advances the object to the state specified in the input mapper. • Demote: Demotes the object from its current normal state to the previous normal state. • Suspend: Stops an object's progression through assigned lifecycle states temporarily. • Resume: Resumes the object that is marked with a paused lifecycle state.

4. Click **Next**.
5. On the **Lifecycle** tab, map process data from the process (Source) to the lifecycle attribute (Destination):

   a. In the Source area of the **Input Message Mapping** screen, map the Object ID of the objects that you want to update with Lifecyles to the **Items** attribute in the **Destination** area.

b. To apply lifecycle to the object, map a Boolean attribute with a value of True to the **State** attribute in the Destination area.

c. Set **Override entry checks** to True to force the object to promote, regardless of other conditions.

6. Click **OK**.

6.2.45 Configuring a Lifecycle Apply activity

1. Drag and drop the **Lifecycle Apply** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Lifecycle** tab, specify lifecycle information as described in the following table:

Field	Description
Lifecycle	Select application parameter for a lifecycle.
Scope	Indicate the scope of lifecycle.
Initial State	Identifies the position of the state or state of the lifecycle associated with the object.

4. Click **Next**.
5. On the **Lifecycle** tab, map process data from the process (Source) to the lifecycle attribute (Destination):

   a. In the Source area of the **Input Message Mapping** screen, map the Object ID of the objects that you want to update with lifecyles to the **Items** attribute in the **Destination** area.

6. Click **OK**.

6.2.46 Configuring a Dematerialize activity

1. Drag and drop the **Dematerialize** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Input Message Mapping** screen, map the Object ID of the Lightweight SysObjects that you want to dematerialize with the **Items** attribute in the **Destination** area.
4. Click **Apply** and then click **OK**.

6.2.47 Configuring a Materialize activity

1. Drag and drop the **Materialize** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Input Message Mapping** screen, map the Object ID of the Lightweight SysObjects to materialize with the **Items** attribute in the **Destination** area.
4. Click **Apply** and then click **OK**.

6.2.48 Configuring a Predict activity

1. Drag and drop the **Predict** activity onto the canvas.
2. Double-click the activity to open the **Activity Inspector**.
3. On the **Predict** tab, configure the activity as described in the following table:

Field	Description
HTTP/RESTful endpoint	Select the HTTP/RESTful endpoint for the activity.
Model Name	Specify the model name to use for the Predict activity.
Model Version	Specify the model version to use for the Predict activity.

4. Click **Test Connection** to verify the connection with the model.
5. Click **Next**.
6. On the **Input Message Mapping** screen, map the packages, variables, attachments, and execution data from the parent process (Source) to the process data of the Predict request (Destination) and click **Next**.
7. On the Output Message Mapping screen, map the available data from the predict response to the parent process.
8. Click **OK**.

Note: The response from Predict activity can be inferred based on business case to make a decision.

6.3 Concepts

6.3.1 Process activities

Process activities represent the tasks performed in a business process. A process model contains the following activities:

- **Start activities:** The activities that start the process.
- **End activity:** The last activity in the process. A process model can have only one end activity.
- **Step activities:** The Intermediate activities between the start and the end. A process model can have any number of step activities.

Every process model consists of at least one start activity, at least one step activity, and one end activity. When you create a process model, the system adds start and end activities. You can add additional start and step activities to the process model.

Manual and system activities

An activity can be either manual or system. One or more users perform a manual activity. The system performs a system activity on behalf of a user.

Use manual activities to create tasks (work items) for end users. You can place these tasks in the inbox of an end user or group. You can also place tasks in a work queue. Users can select tasks to perform from the work queue. These tasks contain the packages and instructions necessary to complete the task.

The manual activities are described in the following table:

Icon	Manual activity	Description
👤	Manual task	Use the Manual task activity to create a task for one person or multiple performers.
👤	Work queue task	Use the Work queue task activity to create a task and associate it with a work queue.

Integration activities

Integration activities communicate with external systems using a specific protocol.

The integration activities are described in the following table:

Icon+	Integration activity	Description
	Activity that calls integration logic	These activities synchronously call integration logic and continue the process flow.
✉	Activity that starts a process based on an integration communication	These activities wait for a message from an integration. A new process starts when the activity receives the message.
✉	Activity that halts a process and waits until it receives an integration communication	These activities wait for a message from an integration. When the activity receives the message, it is correlated with an existing process and the process is triggered to continue its current flow. "Message correlation" on page 80 provides additional information.

Activity definition

The activity definition is located in the Activity Inspector. It describes the characteristics of the activity, including:

- How the activity is executed
- Who performs the work
- How the performer is assigned
- When an activity with multiple incoming flows starts
- What triggers are necessary
- What is the transition behavior when the activity completes

Activities can also have characteristics that are specific to the type of task they represent. For example, an activity that sends content to an external vendor can include an attribute containing the email address of the vendor.

The BPM framework internally handles completion of the work item when it executes automatic activities in a process. If the custom workflow method or Java service has the logic to handle work item completion or clean up, you do not have to handle work item transitions or workflow operations. Otherwise, this may cause exceptions on execution.

6.3.2 Determining which activities to use

Use the following tables to determine which activities to add to your process model.

To start a process, use the Start activities described in the following table:

Activity	Description
Initiate	Starts a process. The system adds this start activity to your process model, but you can add other start activities.
Start from Database "Configuring a start from Database activity" on page 102	Starts a process when a SQL read query on the database returns results. After receiving the results, the activity processes the data as specified in the activity.
Start from Repository "Configuring a start from Repository activity" on page 107	Starts a process when a DQL read query on the repository returns results. After receiving the results, the activity processes the objects as specified in the activity.
Start from Email "Configuring a Start from Email activity" on page 112	Starts a process when an email arrives in an inbox. "Start a process when an email is received" on page 161 provides additional information.
Start from FTP "Configuring a Start from FTP activity" on page 119	Starts a process when a file or directory appears in a monitored folder on an FTP server.
Start from HTTP "Configuring a Start from HTTP activity" on page 123	Starts a process upon receiving an HTTP request. This activity can also send an HTTP response to the sender of the incoming request. "Start a process when an HTTP request is received" on page 162 provides an example.
Start from JMS "Configuring a Start from JMS activity" on page 129	Starts a process when a JMS queue or topic receives a JMS message. "Start a process when a JMS message is received" on page 163 provides an example.
Start from Web Service "Configuring a Start from Web Service activity" on page 134	Starts a process when a web service in the deployment environment receives a Simple Object Access Protocol (SOAP) message. "Start a process when a SOAP message is received" on page 165 provides an example.

To interact with the incoming flow, add Step activities as described in the following table:

Activity	Description
Create "Configuring a Create activity" on page 95	Creates an object in the repository, such as a folder, business object, or content object. You can also create an object by copying an existing object.
Create ACL "Configuring a Create ACL activity" on page 97	Use the Create ACL activity to create a OpenText Documentum CM ACL object from a Permission Set.
Edit Permissions "Configuring an Edit Permissions activity" on page 98	Use the Edit Permissions activity to change access to objects for an end user or group at runtime. "Changing end-user access to objects" on page 159 provides additional information.
Move "Configuring a Move activity" on page 99	Moves a set of objects to a folder or create a shortcut in that folder.
Set Process Data "Configuring a Set Process Data activity" on page 100	Modifies process data as it flows between two other activities.
Call Database Procedure "Configuring a Call Database Procedure activity" on page 100	Connects to an external database and runs a stored procedure or function. "Call Database Procedure" on page 160 provides additional information.
Read from Database "Configuring a read from Database activity" on page 101	Connects to an external database and returns the results of a SQL Select statement. It also supports reading content from a database to a package, with either BLOB type data in Oracle or Bytes type data in Microsoft SQL Server. "Read from Database" on page 160 provides additional information.
Wait for Database "Configuring a wait for Database activity" on page 103	Halts a process until a SQL read query on the database returns results. The activity queries the database at specified intervals. After receiving the results, the activity processes the data as specified in the activity.

Activity	Description
Write to Database “Configuring a write to Database activity” on page 105	Connects to an external database and runs a SQL Insert, Update, or Delete statement. It also supports the exchange of content from packages to database parameters. That is, you can insert or update binary or BLOB data from a package into the database. “Write to Database” on page 161 provides additional information.
Read from Repository “Configuring a read from Repository activity” on page 105	Reads from a OpenText Documentum CM repository.
Wait for Repository “Configuring a wait for Repository activity” on page 108	Halts a process until a DQL read query on the repository returns results. The activity queries the database at specified intervals. After receiving the results, the activity processes the objects as specified in the activity.
Write to Repository “Configuring a write to Repository activity” on page 109	Writes to a OpenText Documentum CM repository.
Send Email “Configuring a Send Email activity” on page 110	Sends email messages to lists of one or more users. “Send Email activity” on page 161 provides additional information.
Wait for Email “Configuring a Wait for Email activity” on page 114	Halts a process until an email arrives in an inbox.
Download from FTP “Configuring a Download from FTP activity” on page 118	Transfers files from an FTP server to your application.
Upload to FTP “Configuring an Upload to FTP activity” on page 120	Transfers files to an FTP server.
Wait for FTP “Configuring a Wait for FTP activity” on page 121	Halts a process until a file or directory appears in a monitored folder on an FTP server.
Send HTTP Request “Configuring a Send HTTP or RESTful Request activity” on page 122	Sends HTTP GET, PUT, or POST requests to, and receives the responses from, a web server.

Activity	Description
Wait for HTTP "Configuring a Wait for HTTP activity" on page 125	Halts a process until the activity receives an HTTP request. This activity can also send an HTTP response to the sender of the incoming request.
Execute Java Method "Configuring an Execute Java method activity" on page 127	Use this activity to run workflow methods.
Execute Java Service "Configuring an Execute Java Service activity" on page 127	Adds custom code to your process model. "Extending process functionality" on page 163 and "Example of extending process functionality" on page 128 provide additional information.
Send JMS Message "Configuring a Send JMS Message activity" on page 129	Sends text, binary, or streaming data to a Java Message Service (JMS) server.
Wait for JMS "Configuring a Wait for JMS activity" on page 130	Halts a process until a JMS queue or topic receives a JMS message.
Call Web Service "Configuring a Call Web Service activity" on page 131	Obtains data from a SOAP-based web service. This activity supports both secure and non-secure web services. "Call Web Service" on page 163 provides additional information.
Wait for Web Service "Configuring a Wait for Web Service activity" on page 136	Halts a process until a web service in the deployment environment receives a SOAP message.
Publish Document "Configuring a Publish Document activity" on page 138	Generates a new Document Sciences xPression document through xPression Server. "Using Document Sciences xPression" on page 165 provides additional information.
Advanced Publish Document "Configuring an Advanced Publish Document activity" on page 139	Provides the same functionality as Publish Document, but instead of using the data mapper to map each variable, you can use customer data XML. "Using Document Sciences xPression" on page 165 provides additional information.

Activity	Description
Lifecycle “Configuring a Lifecycle activity” on page 141	Allows you to apply the lifecycle operation.
Materialize “Configuring a Materialize activity” on page 143	Allows you to materialize the Lightweight SysObjects.
Dematerialize “Configuring a Dematerialize activity” on page 142	Allows you to dematerialize the Lightweight SysObjects.

To create tasks for end users, add Manual activities as described in the following table:

Activity	Description
Manual task “Configuring manual activity tasks” on page 83	Creates a task for one person or multiple performers. This activity is a manual activity. “Configuring the performers of a manual activity” on page 84 and “Determining which task performer to use” on page 151 provide additional information.
Work queue task “Configuring manual activity tasks” on page 83	Creates a task and associates it with a work queue. This activity is a manual activity. “Configuring a work queue for a task” on page 91 provides additional information.

To orchestrate the execution of the process flows, use the activities described in the following table:

Activity	Description
Call Process “Configuring a Call Process activity” on page 116	Launches a subprocess. “Calling a process” on page 162 provides additional information.
Call Stateless Process “Configuring a Call Stateless process activity” on page 118	Launches a stateless process. The performer of the Call Stateless Process activity runs all activities in the stateless process. It overrides the performer set in the Execution tab of the automatic activities. “Stateless processes” on page 78 provides additional information.

Activity	Description
Decision Split	Displays decision points explicitly in your business process model. This activity splits the flow of the process into multiple flows.
Group "Grouping process activities" on page 168	Adds a group to the process model. Groups enable you to organize and simplify complicated process models. "Organizing process activities into groups" on page 186 provides additional information.
Join	Merges two flows in a process into a single flow. "Parallel Flows" on page 184 provides information on intelligent Smart Joins.
Parallel Flow "Configuring a Parallel Flow activity" on page 168	Adds a parallel flow to the process model. Use a parallel flow when you are unable to determine the exact number of input flows for a Join activity until runtime. "Parallel Flows" on page 184 provides additional information.
Signal Parent Process "Configuring a Signal Parent process activity" on page 118	Sends a signal to a parent process to trigger an activity. "Using a signal to trigger a parent process" on page 162 provides additional information.

6.3.3 Determining which task performer to use

An activity definition includes the information that lets the system determine the performer of the activity. Manual activities have several different choices for performers while system activities must identify a user whose permissions are used to run the script or program.

When a manual activity starts at runtime, the system adds a task to the inbox of the end user or users designated as the performer of that activity. For high-volume processing, you can add the task to a work queue used by multiple workers.

Participants in a process have the option to mark themselves as unavailable for the tasks. If the end user selected as the performer is unavailable, the process engine attempts to give the work item to the delegated user for that end user.

The following table lists the categories from which you can choose a performer. Only the first four options are available for system activities.

Performer	How system retrieves performers
Workflow Supervisor	The system selects the user designated as the workflow supervisor when the activity starts. By default, the user who starts the process is the workflow supervisor.
Repository Owner	The system selects the user identified as the owner of the active repository.
Performer of Last Activity	The system selects the performer from the previous finished activity that satisfied the trigger condition of the current activity. The Performer of Last Activity is an alias for the user that performed the last activity. It can include multiple performers and users from other previous activities. This performer is not known until runtime since the performer can come from a group. "Selecting the performer of last activity to perform a manual activity" on page 86 provides additional information for manual activities.
User from Parameter	You select application parameters to define placeholders for users that are determined at runtime. Users selected from application parameters can be filled in by any user in an environment. "Selecting a user from application parameters to perform a system activity" on page 95 provides additional information for system activities. "Selecting users from application parameters to perform a manual activity" on page 86 provides additional information for manual activities.
User from Process Data	You select users from process data using package attributes or variables. The system retrieves the user from the dynamic variables or attributes that you select in the current manual activity. "Selecting a user from process data to perform a manual activity" on page 87 provides additional information.

Performer	How system retrieves performers
User from Alias Set	You select aliases from alias sets as placeholders for users. The system retrieves the user from alias value using alias set object in repository. “Selecting a user from alias set to perform a manual activity” on page 86 provides additional information.
Application Role	The system retrieves the role that you select and considers it as a single entity when sending tasks. Application roles are groups of users in an application that perform the same business function. “Selecting an application role to perform a manual activity” on page 87 provides additional information.
Group from Parameter	You select application parameters to define placeholders for groups that are determined at runtime. A group selected from an application parameter can be filled in by any group in an environment. “Selecting a group from application parameters to perform a manual activity” on page 88 provides additional information.
Group from Process Data	You select a group from process data using package attributes or variables. The system retrieves the group from the dynamic variables or attributes that you select in the manual activity. “Selecting a group from process data to perform a manual activity” on page 88 provides additional information.
Group from Alias Set	You select aliases from alias sets as placeholders for groups. The system retrieves the group from alias value using alias set object in repository. “Selecting a group from alias set to perform a manual activity” on page 89 provides additional information.

Performer	How system retrieves performers
Performer from Previous Activity	The system retrieves the performer or performers of any previous activity that you select in the current manual activity. You can select any manual activity with an inbound connection to the manual activity. Performer from Previous Activity is not available to use as a conditional performer. "Selecting the performer from previous activity for a manual activity" on page 89 provides additional information.
Previous Activity Performer to Choose at Runtime	This option indicates that the performer of the previous activity can select the performer for the next activity at runtime. "Selecting the performer from previous activity to select next performer at runtime for a manual activity" on page 90 provides additional information.

6.3.4 Using work queues for task processing

A work queue is a task list that multiple end users can access. Available processors assigned to a work queue perform the tasks.

A work queue enables work to flow more efficiently in situations where the volume of tasks and the need for quick processing are high. For example, a loan-processing center that has applications for several types of loans with different processing requirements is a good fit for work queues. You can create queues for different purposes, enabling you to manage and balance the workload in your organization.

The system assigns a task to a work queue based upon the task and work queue properties. Processors assigned to work on that queue receive tasks in their inbox in priority order. The queue manager can also manually assign a task to a particular user. Processors with the queue_advance_processor role can selectively pull tasks from the queue regardless of the priority. They do not have to wait to receive tasks in their inbox.

6.3.5 Work queue roles

Work queue end users are referred to as processors. There are two roles for processors: queue_processor and queue_advance_processor.

Processors with the queue_processor role cannot browse work queues. They can elect to have the system automatically deliver the next task when their queue is empty or they can request the next task manually. The system then assigns the processor the next highest priority task from among the work queues of that processor.

Only processors with the queue_advance_processor role can view and access work queue tasks in a work queue task list. A queue manager can directly assign a task to any processor.

The following table describes the work queue roles:

Role	Description
Queue_admin	Creates work queues and queue policies, assigns processors to work on queue tasks, and defines skill profiles. Skill profiles enable the application to assign tasks to the appropriate processor. Queue administrators can reassign and suspend tasks. They can also add, edit, or assign skill profiles to the individual work queue processors. Queue administrators can view work queue tasks. They can view the name of the processor assigned to the task, the task status, the task received time, and the current task priority. Members of the queue_admin role do not have the administrator role by default. Queue administrators who have CREATE_GROUP privileges can create work queues.
Queue_manager	Monitors work queues to determine which queues have overdue tasks or too many tasks. Queue managers assign roles to queues and assign users to work on queue tasks. Queue managers can reassign and suspend tasks. They can also add, edit, and assign skill profiles to individual work queue processors. Queue managers can view work queue tasks. They can view the name of the processor assigned to the task, the task status, the task received time, and the current task priority. Queue managers who have CREATE_GROUP privileges can create work queues.
Queue_processor	Works on tasks that the system assigns from one or more work queues. Queue processors can request work, suspend and resume work, and complete work.
Queue_advance_processor	Works on tasks that the system assigns from one or more work queues. Queue advanced processors can also select tasks to work on from these work queues.

- Unassign task can be performed by all the roles.
- Work queue tasks cannot be delegated.

6.3.6 End-user tasks

A task is work that requires an end user to perform some action. The process creates tasks through a Manual task activity or a Work queue task activity. Actions available for tasks depend upon the work queue role of the end user and the properties of the task.

Manual task actions

Manual task actions are the actions that an end user can perform on a manual task at the runtime using Smart view, OpenText Documentum CM clients, and custom clients.

The following table describes the manual task actions:

Task action	Description
Acquire	Accepts a group task that is not assigned. Once acquired, another end user cannot complete it.
Complete	Completes the task.
Reject	Completes the task and sends the task to the reject flow. This action is only available when enabled in the Manual task activity.
Delegate	Saves the task data and sends the task to another end user. This action is only available when enabled in the Manual task activity.
Hold	Pauses a task.
Unhold	Removes the hold from a paused task and returns it to the correct state.
Halt	Pauses a task indefinitely.
Resume	Resumes a halted task.
Repeat	Completes the task and adds additional performers to work on the current task. This action is only available when enabled in the Manual task activity.

Work queue task actions

Work queue task actions are the actions that a work queue member can perform on a work queue task at runtime. The actions they can perform depend upon their work queue role. *Work queue roles* describes each of the work queue roles.

These work queue task actions are only relevant in the context of a work queue task list. An end user with the queue_processor role cannot view a work queue task list. Queue processors only see tasks in their individual inboxes.

The following table describes the work queue task actions available in a work queue task list and the work queue roles required to complete the actions:

Task action	Description	Work queue role
Acquire	Accepts a task that is not assigned.	Queue advanced processor, queue manager, or queue administrator
Update	Updates the task metadata without completing the task.	Queue advanced processor, queue manager, or queue administrator
Complete	Completes the task.	Queue advanced processor, queue manager, or queue administrator
Reject	Completes the task and sends the task to the reject flow.	Queue advanced processor, queue manager, or queue administrator
Delegate	Saves the task data and sends the task to another end user.	Queue advanced processor, queue manager, or queue administrator
Hold	Pauses a task.	Queue manager or queue administrator
Unhold	Removes the hold from a paused task.	Queue manager or queue administrator
Halt	Pauses a task indefinitely.	Queue manager or queue administrator
Resume	Resumes a halted task.	Queue manager or queue administrator
Repeat	Completes the task and adds additional performers to work on the current task. This action is only available when enabled in the Work queue task activity.	Queue advanced processor, queue manager, or queue administrator
Assign	Assigns a task to a member of the work queue.	Queue manager or queue administrator
Unassign	Puts a task back into the work queue.	Queue manager or queue administrator
Reassign	Assigns the task to a different user in the work queue.	Queue manager or queue administrator
Get next task	Gets the highest priority task from all of the assigned work queues of the processor that match the skills of the processor.	Queue advanced processor, queue manager, or queue administrator
Return to Queue	Returns a task to the work queue.	Queue advanced processor, queue manager, or queue administrator
Change work queue	Changes the work queue of a task.	Queue administrator

6.3.7 Matching end-user skill sets to work queue tasks

Work queue skill sets qualify who can work on a task in a work queue. When you create a work queue, you can define the skills or expertise necessary to process tasks in the work queue. The processor profile in Documentum Administrator enables you to assign those skills to work queue processors. When a processor gets the next task or when a manager assigns a task, the system qualifies the processor based upon the skills required for the task. A particular task associated with a queue can require one or more skills to complete. A processor can have several skills related to a work queue.

For example, the work queue loan_underwriter_queue has three required skills defined for it: auto loans, commercial loans, and home loans. When an auto loan application enters the process, the system evaluates the skills defined in the process activity and resolves the skill value for an auto loan. It then sends the loan application to the loan_underwriter_queue. When a supervisor assigns a task or when a processor tries to get a task, the system ensures that this processor has the auto loans skill before allowing the processor to acquire the task.

It is not necessary to enable skill-set matching. If you do not define skills for a work queue, the system does not qualify the processors based on skills. In this case, any queue processor in the work queue can use the Get Next Task action to work on the tasks regardless of qualifications.

It is a best practice to calculate a baseline timing of a task list before implementing skill sets. This baseline calculation enables you to determine how skill-set matching affects performance. It is important to realize that performance costs grow as the number of tasks grows in a task list.

If you use work queues and skill sets in your process model, they must exist in the repository before deploying the application. Use Documentum Administrator to define the work queues and skills in the repository. The *OpenText Documentum Server Administration and Configuration Guide* provides additional information.

6.3.8 Understanding task priority levels

System and manual activity tasks have priority levels that can be set within the activity. These priority levels are low, medium, and high. Depending on the type of task, these levels have different functions.

Work queue tasks can also have an additional priority level: dynamic. Dynamic priority uses a custom task priority module that enables you to specify the initial task priority and how the priority can change within a process.

The following table describes the priority levels available for each type of task:

Task	Priority level description
System	By default, the system executes system tasks in the order created. The system uses the priority levels only if the Documentum CM Server is configured to use priorities for system activities. The *OpenText Documentum Advanced Workflow Deployment Guide* provides instructions on enabling priority levels for system activities.
Manual	Processors can use the priority levels to prioritize their tasks.
Work queue	The system evaluates the low, medium, and high priorities for distributing tasks to end users. Queue managers can also use these priorities to distribute tasks. The system can use a dynamic custom priority module to determine priority instead of using the low, medium, and high priority levels. It enables you to use complex priority and aging scenarios to set and update the priority levels. Dynamic priority is set in the process at runtime rather than in the activity properties.

6.3.9 Custom task priority modules

A custom task priority module enables you to create complex initialization and aging scenarios for your work queue tasks. You can specify the initial task priority and the frequency and percentage at which it increments based on different values in the priority module.

A priority module contains logic to calculate and update the priority dynamically based on process data or other attributes that belong to that process. For example, if a loan application from a preferred customer arrives in a work queue, it can receive a higher priority than applications from other customers. If the request is from a preferred loan broker or for a high loan amount, the priority can increase at a higher rate. You can also increase the priority of a task as it nears a deadline or some other time restriction. This priority increase pushes the task up the queue at a higher rate.

6.3.10 Changing end-user access to objects

Use the Edit Permissions activity to change access to objects for an end user or group at runtime. The following table describes the Access Control List (ACL) basic permissions you can set:

Basic permission	Access
None	No access to the object.
Browse	View the properties of the object but not the object content.
Read	View both the properties and content of the object.
Relate	Add annotations to the object (includes Browse and Read permissions).

Basic permission	Access
Version	Modify the content of the object and check it in as a new object version with a new version number (includes Browse, Read, and Relate permissions). End users cannot overwrite an existing version or edit the object properties.
Write	Edit object properties and check in the object as the same version (includes Browse, Read, Relate, and Version permissions).
Delete	Delete objects (includes Browse, Read, Relate, Version, and Write permissions).

These basic permissions grant or revoke the ability to access and manipulate the attributes and the content of an object. The basic permission levels are hierarchical and each higher access level includes the capabilities of the preceding access levels. For example, an end user with the Relate permission also has Read and Browse permissions.

When changing permissions, if the permission exists for that end user, the new permission replaces the existing permission for the object. If you add two permissions for the same end user, the system applies the highest permission level.

6.3.11 Read from Database

Use the Read from Database activity to connect to an external database and return the results of a SQL Select statement. It also supports reading content from a database to a package, with either BLOB type data in Oracle or Bytes type data in Microsoft SQL Server.

The supported data types for database parameters are: CHAR, VARCHAR, NVARCHAR, BIGINT, INTEGER, SMALLINT, TINYINT, BIT, DOUBLE, FLOAT, NUMERIC, DECIMAL, REAL, DATE, TIME, TIMESTAMP, BOOLEAN, BINARY, BLOB, LONGVARBINARY, VARBINARY.

6.3.12 Call Database Procedure

Use the Database Procedure activity to connect your system to an external database and run a stored procedure or function. The activity returns the values for OUT and IN/OUT parameters or the result set of the stored procedure.

The supported data types for IN, OUT, and IN/OUT parameters are: CHAR, VARCHAR, NVARCHAR, BIGINT, INTEGER, SMALLINT, TINYINT, BIT, DOUBLE, FLOAT, NUMERIC, DECIMAL, REAL, DATE, TIME, TIMESTAMP, BOOLEAN, BINARY, BLOB, LONGVARBINARY, and VARBINARY.

6.3.13 Write to Database

Use the Write to Database activity to connect to an external database and run a SQL Insert, Update, or Delete statement. It also supports exchange of content from packages to database parameters. That is, you can insert or update binary or BLOB data from a package into the database.

The supported data types for database parameters are: CHAR, VARCHAR, NVARCHAR, BIGINT, INTEGER, SMALLINT, TINYINT, BIT, DOUBLE, FLOAT, NUMERIC, DECIMAL, REAL, DATE, TIME, TIMESTAMP, BOOLEAN, BINARY, BLOB, LONGVARBINARY, VARBINARY.

6.3.14 Send Email activity

Use the Send Email activity to send email messages to lists of one or more users. For example, you can send an email response to a customer complaint or send an expense report for approval by including it in the body of an email message. The Send Email activity enables you to add attachments and has attributes which you can map to set values in email headers.

The activity defines the following parameters:

- Content of the message
- Address to which it is delivered
- SMTP server used to send the email
- Action to take with the email response

For added flexibility, the subject and body of the message can contain static text as well as placeholder values that can be mapped to process data. In addition, the body of the email can contain HTML that is copied in from a third-party HTML editor.

After configuring the subject and body, use the data mapper to configure remaining details. Map information from the context of the process to the attributes of the message such as the sender and recipients. Then map the process artifacts such as packages and variables to the output message ID.

6.3.15 Start a process when an email is received

Use the Start from Email activity to start a process when an email arrives in an inbox. For example, in a customer service process, this activity polls an email server and starts a process when it receives a customer email message. The system can route the email through a manual activity to a performer who reviews the message to assist the customer.

The output of the Start from Email activity is email message data. The email message data is mapped within the activity to the data model of the process.

6.3.16 Calling a process

Use the Call Process activity to call a process. You can decide whether this activity waits for the process to complete before continuing to the next activity within the parent process.

You have the following options:

- The Call Process activity invokes the process and waits for it to complete. After the process finishes by either completing or terminating, the parent process continues to the next activity. The completed process data can be mapped back to the parent process. The execution data includes the completed_sucessfully attribute. This attribute is true if the process returns data correctly and false if the process aborts or terminates.
- The Call Process activity invokes the process and then immediately continues to the next activity. Since the activity completes before the process, the only data that can be mapped back to the parent process is the process ID of the process.

6.3.17 Using a signal to trigger a parent process

You can use a signal to trigger an activity in a parent process. You can configure the activity in the parent process to wait for a signal to arrive from a subprocess before it starts. For example, a parent process can orchestrate multiple subprocesses in different states of completion. Each subprocess can send a signal to the parent process with a status update. These signals trigger an activity in the parent process and provide status information on all of the subprocesses.

A parent process calls a subprocess using a Call Process activity. Any step activity in the parent process can pause and wait for a signal from the subprocess before the activity starts. The subprocess uses a Signal Parent Process activity to signal the parent process, which enables the parent process to continue.

6.3.18 Start a process when an HTTP request is received

Use the Start from HTTP activity to start a process upon receiving an HTTP request. For example, in a real estate application, this activity could start a process after receiving an HTTP request with a mortgage document in the message-body. The system can route the contents of the HTTP request through a manual activity to a performer who reviews the mortgage document.

This activity can also send an HTTP response to the sender of the incoming request. For example, the process could send an HTTP response to confirm that the loan application was received.

6.3.19 Extending process functionality

Use the Execute Java Service activity to extend the functionality of your process beyond the standard process activities provided by default. You can work with a developer who creates custom Java code to provide the extended functionality. The developer integrates the code into the application as a Java module and the extended functionality then becomes available through the Execute Java Service activity.

A process can use multiple Execute Java Service activities to access relevant methods from any Java services integrated into your application. Before you configure the Execute Java Service activity, ask the developer for the Java service name and the name of the method that provides the functionality.

6.3.20 Start a process when a JMS message is received

Use the Start from JMS activity to start a process when a JMS queue or topic receives a JMS message. For example, in a claims processing process, you can configure a Start from JMS activity to listen to a claims queue. The activity starts the process when a message reaches the queue. The system then routes the data from the message to the intended destination.

6.3.21 Call Web Service

This activity supports both secure and non-secure web services.

Operations listed in the WSDL do not appear in this activity under the following conditions:

- Operations defined in the binding that are not supported in SOAP 1.1 or SOAP 1.2.
- Operations defined in the WSDL that have a mismatch between the name attribute of the `input` and `output` message tags of the `binding` and `port`. For example: (/binding/operation/input and /binding/operation/output) and (/portType/operation/input and /portType/operation/output).

6.3.22 Invoking a secure web service

The Call Web Service activity in Advanced Workflow supports three kinds of security:

- HTTP proxy support
- HTTP basic authentication
- SOAP header-based authentication

Before configuring a secure web service, you can configure the HTTP proxy server. The *OpenText Documentum Advanced Workflow Deployment Guide* provides information on how to configure an HTTP proxy server for web services.

If the web service being invoked is protected by HTTP authentication, use HTTP basic authentication in the web service endpoint to pass the username and password. “Configuring a web service endpoint” on page 36 provides additional information.

If the web service being invoked is protected by SOAP Header-based authentication, map the security token and SOAP-secured parameters in the SOAP Header node. The SOAP Header can be found in the input mapping (SOAP Envelope/SOAP Header).

To invoke a web service secured by Secure Socket Layer (SSL), you must have an SSL certificate in your environment. The *OpenText Documentum Advanced Workflow Deployment Guide* provides instructions on importing SSL certificates to support encrypted connections for activities.

6.3.23 Invoking Foundation SOAP API services from Advanced Workflow

You can invoke Foundation SOAP API services in one of two ways from Advanced Workflow:

- As a single service, meaning that a process contains one web service activity that performs a single invocation of a Foundation SOAP API web service.
- As a chained service, meaning that a process contains multiple web service activities that invoke Foundation SOAP API web services. The repository identity information is registered to a token that each activity uses.

Web service activities support base64 and MTOM for content transfers.

6.3.24 Web Service implicit data mappings

The Start from Web Service and Wait for Web Service activities enable you to define the web service inputs based on Process Data or an Existing Web Service. If you select Process Data, you do not have to map the data. You also do not have to map the correlation set in the Wait for Web Service activity. The mappings are implicit and the system completes the mappings.

The SOAP message expected by the operation depends upon the visible process variables and business objects for that activity. During input mapping, the system maps the variables and business objects received from the SOAP message to the variables and business objects of the process.

During output mapping for the Start from Web Service activity, the SOAP message returned by the operation contains the workflow ID. During output mapping for the Wait for Web Service activity, the SOAP message returned by the operation contains the workflow ID and workitem ID.

In correlation mapping, if the correlation set uses a variable, the system checks whether the variable value in the message matches the value in the process.

6.3.25 Start a process when a SOAP message is received

Use the Start from Web Service activity to start a process when a web service in the deployment environment receives a Simple Object Access Protocol (SOAP) message. For example, in a resource planning application, this activity could start a process when it receives a SOAP message ordering supplies for regional offices. The system could route the contents of the SOAP message through a manual activity to a performer who supervises shipping.

The web service that receives SOAP messages is defined when you configure the Start from Web Service activity.

6.3.26 Using Document Sciences xPression

Use the Publish Document activity to generate a new Document Sciences xPression document through xPression server. As part of the process, document variables are replaced in the document when the document is published. xPression Server produces various customer documents and distributes them in outputs such as print, email, archive, or web-based formats.

Use the Advanced Publish Document activity when there are multiple constant values to map to process data. This activity provides the same functionality as Publish Document, but instead of using the data mapper to map each variable, you can use customer data XML. You would use the customer data XML to create the mappings for constants and add placeholders for dynamic data. You would also use the data mapper to map only the placeholder values that the XML creates.

6.3.27 Customer Data XML sample code

The following sample code shows the Customer Data XML for a medical records release request. The request number, request date, and the telephone number are all variables that must be mapped in the data mapper.

```
<dataroot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <CUSTOMER_DATA>
   <MedicalRecordReleaseRequest>
    <reqNo>$RequestNo</reqNo>
    <requestDate>$ReqDate</requestDate>
    <patientName>John Doe</patientName>
    <patientAddress_street1>1234 Cherry Circle</patientAddress_street1>
    <patientAddress_street2></patientAddress_street2>
    <patientAddress_city>San Jose</patientAddress_city>
    <patientAddress_state>California</patientAddress_state>
    <patientAddress_zipCode>95131</patientAddress_zipCode>
    <patientDOB>1976-01-01</patientDOB>
    <patientSSN>123-12-1234</patientSSN>
    <patientHospitalName>Mercy Hospital</patientHospitalName>
    <orgName>Mercy Medical Foundation</orgName>
    <orgTelNo>$$TelephoneNo</orgTelNo>
   </MedicalRecordReleaseRequest>
  </CUSTOMER_DATA>
</dataroot>
```

In this example, if requestDate is of type XSD dateTime, use the following format string:

```
yyyy-MM-dd'T'HH:mm:ss.SSSZ
```

6.3.28 Lifecycle

A lifecycle defines a sequence of states an object can encounter. Typically, lifecycle is designed for document to describe a review process. For example, when a end user creates a document, sends it across to other users for review and approval. The lifecycle defines the state of the object at every stage in the process.

Lifecycle contains:

- States: A lifecycle can be in one of a normal progression of states or in an exception state. The defined states are Promote, Demote, Suspend, and Resume.
- Object types that can be attached to the lifecycle: Any system object or subtype can have an attached lifecycle.
- Validation procedures: A lifecycle can trigger custom behavior in the repository when an object enters or leaves a lifecycle state.

For more information about Lifecycles, see *OpenText Documentum Server Fundamentals Guide*.

6.3.29 Predict Activity

Predict activity helps in improving decision making for business processes by analyzing the business data and comparing it with historical data. It helps business analysts to detect patterns and make selections that might be hard to find or take a long time to infer. Predict activity is helpful in domains such as sales forecasts, consumer demand analysis, and fraud detections. It uses Google TensorFlow for analyzing and interpreting the business data.

Chapter 7

Working with flows and data mapping

7.1 Tasks

7.1.1 Connecting activities with flows

1. Use one of the following activity flows to connect the activities in a process model:

   - To advance process data from one activity to the next, add a forward flow. Click **Straight flow** to add a forward flow with a single line. Click **Multi-Segment flow** to add a forward flow with a line consisting of multiple segments.
   - To direct process data in a backward loop or in an alternate forward flow, click **Create Reject Flow**.
   - To connect to a fault handler activity, click **Use this flow to assign Fault Handlers**.

2. Drag and drop the flow from the edge of one activity to the next to connect the activities.

7.1.2 Changing the appearance of flows

Follow this procedure to change how the flow lines appear and to add flow labels to make your process model more descriptive.

1. On the process model editor canvas, double-click an activity flow line to open the **Flow Inspector**.

2. Configure how the flow appears on the process model as described in the following table:

Field	Description
Flow visual style	Select whether to use a single line or multi-segment lines. Multi-segment lines can be easier for users to follow.
Label font	Select a font to use for the flow label.
Point size	Select the size of the label text.
Bold	Select to bold the label text.
Italic	Select to italicize the label text.
Show label	Select to display a label for the flow. Clear this field to hide the flow label.

Field	Description
Show visible packages at destination activity	Select to label the flow with the names of the packages that the following activity handles.
Custom label	Select to add your own flow label. Type the text for the label.

3. Click **OK**.

7.1.3 Configuring a Parallel Flow activity

1. Drag and drop the **Parallel Flow** activity onto the canvas and position it within the main process model.
2. Adjust the Parallel Flow as required for your process model:
   - To resize the Parallel Flow, select an edge of the Parallel Flow container and drag it.
   - To replace an activity, drag and drop a replacement activity from the **Activities** palette onto the first activity or any step activity. Click **Yes** to confirm the replacement.
   - To configure the first activity instead of replacing it, double-click the **Decision Split** activity to open the **Activity Inspector**. Change the activity name in the field at the top of the **Activity Inspector**. Click the **Transition** tab and adjust the transition configuration. Complete the configuration of this activity as you would do for any other activity and click **OK**.
   - To add additional activities and flows, add them between the first activity and the **Smart Join** activity.
   - To change the background color, double-click the background of the Parallel Flow, select a background color from the drop-down list, and click **OK**.

7.1.4 Grouping process activities

1. Drag and drop a **Group** activity onto the canvas.
2. Double-click the activity to open the **Group Inspector**.
3. Type the label for the group.
4. Select the label font and the background color of the group.
5. Click **OK**.
6. To make the group larger, drag an edge to enlarge the group boundary.
7. Drag and drop activities that you want to include in the group into the group container.
8. Connect the activities in the process model with process flows.
9. To hide the activities in the group, click the minus sign. To expand the group, click the plus sign.

7.1.5 Annotating the process model

You can add text to the visual layout of the process model using notes. Notes annotate the process model and provide clarification for application designers. They do not change the processes.

1. On the toolbar, click **Create Note** and then click the process model editor canvas to place the note on the canvas.
2. Drag and drop the note to reposition it.
3. Double-click the note to open the **Note Inspector**.
4. On the **Note Content** tab, type the text of the note.
5. On the **Display** tab, in the **Font** area, configure the font and style of the note text.

   Changes to the note settings appear in the **Preview** area.
6. In the **Format** area, configure the alignment and color of the note as described in the following table:

Field	Description
Format	Specify how to justify the text in the note.
Transparency	Set the level of transparency using the slider control. If you set the transparency level to 100 percent, the note is opaque and hides any objects behind it. If you set the transparency level to 0 percent, the note is transparent.
Text Color	Select the color of the note text.
Background Color	Select the color of the note background.

7. In the **Preview** area, specify the border of the note as described in the following table:

Field	Description
Border	Displays a yellow border.
BPMN style	Displays the note in the Business Process Modeling Notation (BPMN) format for text annotations.

8. Click **OK**.
9. To resize the note, click and drag a black handle on the edge of the note.

7.1.6 Changing a process activity image

Use this procedure to change the image that appears in the process model for an activity. Your custom image file should be in the **Resources** navigator.

1. Double-click an activity to open the **Activity Inspector**.

2. On the **Display** tab, configure the activity image as described in the following table:

Field	Description
Use custom image for activity	Select to replace the default image for this activity with your own image.
Image source	Click **Select** to locate your image. In the **Select Resources** dialog box, select your custom image file and click **Finish**.
Image size	Select a percentage to adjust the image size.

3. Configure the activity label as described in the following table:

Field	Description
Show label	Select to include the activity label. Clear this field to hide the activity label.
Point size	Select the size of the label text.
Bold	Select to bold the label text.
Italic	Select to italicize the label text.

4. Click **OK**.

7.1.7 Configuring trigger conditions to start an activity

1. Double-click an activity to open the **Activity Inspector**.

2. On the **Trigger** tab, specify when to start this activity as described in the following table:

Field	Description
All input flows are selected	Select to start this activity when all immediately preceding activities are complete.
This number of input flows selected	Select to specify how many input flows must complete before this activity starts.
And when this signal arrives	Select to wait until a specific action occurs before this activity starts. Type the name of the trigger signal that should arrive before this activity starts.
This activity can run more than once in a workflow	Select to enable this activity to run more than once in a process. Structure your process model so that only one instance of the activity can be active at any time.

3. Click **OK**.

   Best Practice: Verify that the **Trigger** tab of each activity is defined with the correct number of incoming activities before you run the application and start the process.

7.1.8 Setting activity timers

1. Double-click an activity to open the **Activity Inspector**.
2. On the **Timers** tab, click **Add** to add a timer to the activity.
3. In the **Timer Configuration** dialog box, configure the timer as described in the following table:

Field	Description
Label	Type a label to identify the timer.
Timer starts on	Select whether the timer starts when the process starts or when the activity starts.
Timer removed on	Displays the event that removes the timer.

4. In the **Duration of Timeout** area, type an integer or an expression to calculate the expiration date of the timer. The result of the expression must be an integer. At runtime, if the system evaluates the expression as a negative value, then the action for the timer executes immediately.
5. In the **Duration in** field, select the unit to use for the duration.
6. In **Actions on Timeout** dialog box, click **Add** to add an action to run when the timer expires, and then on the **Add Action** screen, select an available action as described in the following table:

Action	Procedure
Call Process	On the **Call Process** screen, select a process and click **Next**. On the **Call Process <<*process name*>>** screen, define the input values from the current context to call the process and click **Finish**.
Call Stateless Process	On the **Call Stateless Process** screen, select a stateless process and click **Next**. On the **Call Stateless Process <<*process name*>>** screen, define the input values from the current context to call the process and click **Finish**.
Set Value	In the context data tree, update the process data value.
Delegate Task	On the **Delegate Task** screen, in the **Delegate to** field, click **Select**. In the **Select Performer** dialog box, select the performer of the delegated task.
Complete Task	Click **Finish**.

7. Click **Add** to add additional actions.

8. To change the order of the actions, select an action and click **Move Up** or **Move Down**.

9. In the **Reoccurrence** area, select **Actions on timeout will reoccur every duration** to enable the actions to be repeated and type an integer or an expression to calculate how often the actions repeat. The result of the expression must be an integer.

   Reoccurrence is not supported in process debugger.

10. In the **Duration in** field, select the unit to use for the duration.

11. Click **Finish**.

12. If your timer starts when the activity starts, on the **Timers** tab, in the **Due Date** column, select the due date that the system uses to calculate overdue tasks.

13. In the **Calendar** area, select the calendar to use to convert the duration to a trigger date as described in the following table:

Field	Description
Use standard calendar	Specifies the standard calendar to calculate the timer duration.
Use process business calendar	Specifies the calendar defined in the process properties of the process model to calculate the timer duration.
Use specific business calendar	Select a business calendar to use to calculate the timer duration.

14. Click **OK**.

7.1.9 Configuring transitions between activities

1. Double-click an activity to open the **Activity Inspector**.

2. On the **Transition** tab, specify a transition to determine which activities come next in the process flow.

   - To route the process flow to all the subsequent activities, select **Select all connected activities**. If the activity has multiple performers, specify the number of performers that must complete the task before the activity completes. Go to Step 4.
   - To let the performer determine the next activities in the process flow, select **Let performer select the next activities** and go to Step 3.
   - To route the process flow to the next activities based on a set of conditions, select **Select next activities based on these conditions**. "Configuring a conditional transition between activities" on page 173 provides information for configuring a conditional transition.

3. If you let the performer select the next activities in the process, specify the transition rules:

a. In the **Select up to [<*n*>] activities** field, select the maximum number of activities the performer can select.

b. If the activity has multiple performers, specify the number of performers that must complete the task before the activity completes.

c. If you let multiple performers select the next activities, select when to start the next activities as described in the following table:

Field	Description
Any performer rejects	Starts selected reject activities immediately without waiting for other responses.
Any performer forwards	Starts selected forward activities immediately without waiting for other responses.
All performers complete the task	Starts the selected next activities only after the number of performers specified complete the task. Select which activities to start if the performers select both forward and reject activities.

4. Click **OK**.

7.1.10 Configuring a conditional transition between activities

This procedure only applies where flows connect one upstream activity to one or more activities downstream.

You can specify rules that determine which flows are taken when the upstream activity completes. The use of rules makes this a conditional transition.

1. Double-click the upstream activity to open the **Activity Inspector**.

2. On the **Transition** tab, select **Select next activities based on conditions** to display the **Transitions** area.

3. Configure one or more rules.

   a. Double-click an If-Then-Else rule to view it in the rule editor.

   b. Specify a condition using the expression editor.

   c. Select one or more activities to occur with that condition.

   d. Click **OK**.

   e. On the **Transition** tab, continue to configure rules, clicking **Add** to add new rules and **Apply** to save changes.

4. Click **OK**.

   Clicking **OK** and **Apply** check the rules for errors.

7.1.11 Configuring activity email notifications

1. Double-click an activity to open the **Activity Inspector**.
2. Click **Notification**.
3. For each notification event listed, select whether to send a system email notification or a custom email notification.
4. To send a custom email notification:

   a. Select **Send custom email** and click **Edit**.

   b. In the **Email Template** field of the **Notification Template Wizard**, select the email template to use for the notification and click **Finish**.

   c. Click **Next**.

   d. On the **Input Message Mapping** screen, for the input message, map the process data (Source) to the email message attributes (Destination).

   Map the **To** recipient node in the **Destination** area. Map all placeholder nodes that reside under the **Subject** and **Body** nodes.

   e. Map the email attachment in the process data to the email message attachment. If you have multiple email attachments, you can click **Add** on the Attachment node to add more attachments.

   f. Click **Next**.

   g. On the **Output Message Mapping** screen, map the Message ID of the sent message (Source) to process data (Destination) and click **OK**.

5. When you finish your selections for the notification events, click **OK**.

7.1.12 Configuring process data within an activity

1. Double-click an activity to open the **Activity Inspector**.
2. On the **Data** tab, select each package or process variable separately to configure them.

   - To select a package, expand the **Packages** node and select the package.
   - To select a process variable, expand the **Process Variables** node and select a process variable.

3. Configure the packages and process variables as described in the following table:

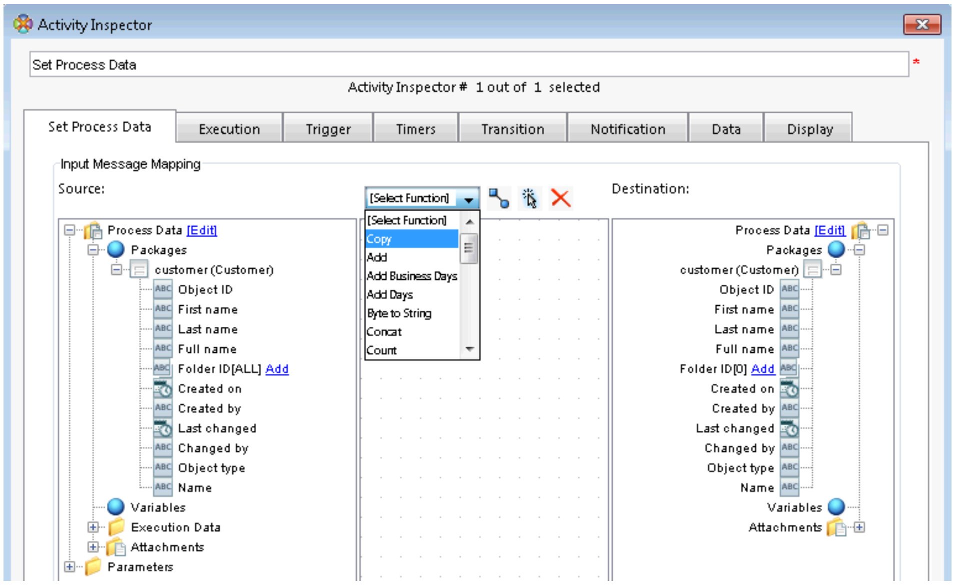
Field	Description
Version	For a content package, select or type the version of the package that applies to this activity. You can type a specific version, for example, 2.0 or 3.0. You can also type a symbolic version, for example, Draft. This version overrides the version configured in the process model.

Field	Description
Visible at this activity	Select this option to make the package or process variable available to the performer of this activity. Clear this option if you do not want the performer of this activity to see the package or process variable at runtime.
This is a mandatory package	Select to require filling the package before the activity completes.

4. Click **OK**.

7.1.13 Mapping process data

1. Double-click an integration activity to open the Activity Inspector.
2. Navigate to the process data mapping tool.
3. On a data mapping screen (Input Message Mapping, Output Message Mapping, or Correlation Set Mapping), select a process data mapping function from the drop-down list.

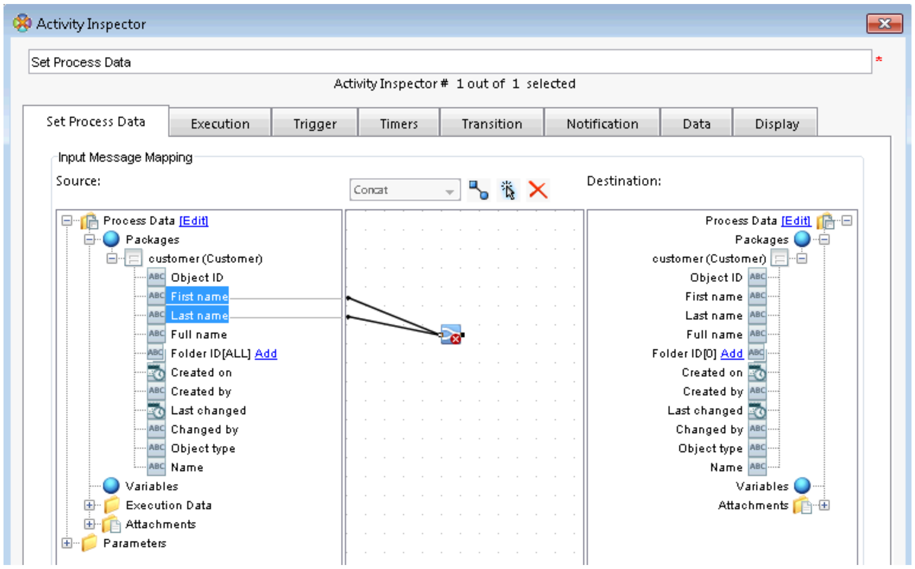


For example, to copy values from the source to the destination, select the **Copy** function.

4. Drag the function icon to the location where you want it to appear. Its position does not affect the order of execution.
5. In the **Source** area, select the attributes whose values are the input data for the function.

   - To select an attribute, click the name of the attribute. A line appears in the mapping area, connecting the selected attribute to the current function icon.

If the current function accepts multiple input values (for example, the **Concat** function), you can select multiple attributes from the **Source** area.



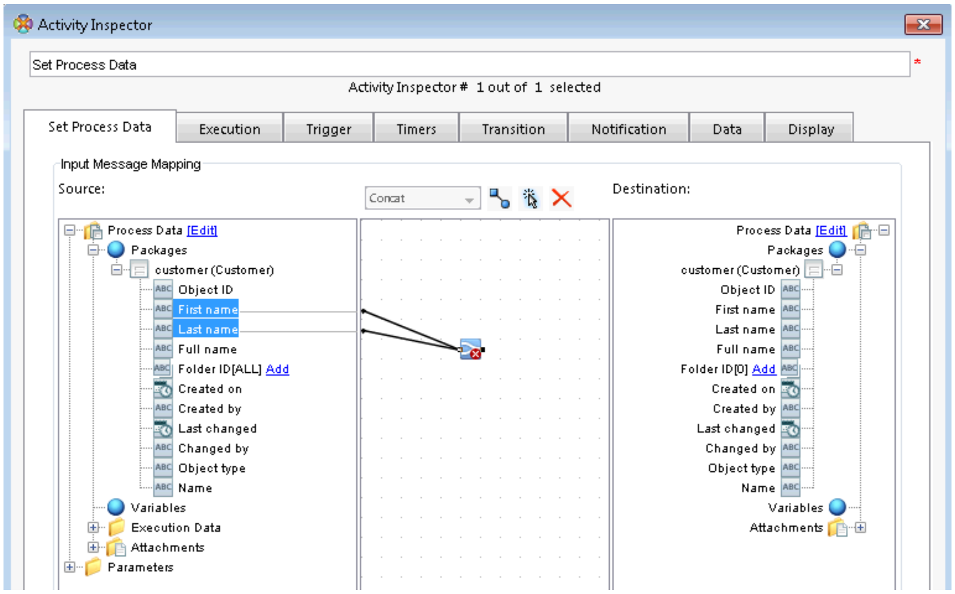
Set Process Data

Activity Inspector # 1 out of 1 selected

Concat

- To clear the attribute and remove the line, click the name of the attribute a second time.
- To select an attribute that is already mapped to another function, click **Line drawing mode**. Draw a line from the connecting point on the previous selection line to the function icon.
- To define a function that uses constants instead of attributes as input data, skip this step and type the constant values in step 7.

6. In the **Destination** area, select the attribute that receives the result of the function.



Set Process Data

Activity Inspector # 1 out of 1 selected

Concat

7. To modify the function properties, double-click the function icon to open the **Function Editor**. Make any necessary changes and click **OK**.

   - To add a constant as additional input value, in the **Function Parameters** area, click **Add a constant parameter**. Type the constant in the field. For example, Reviewed.
   - To change the order of the attributes, use the arrows to move them up or down in the list.

8. Repeat these steps for each data mapping function that you want to create.

7.1.14 Configuring a data mapping function

1. Double-click an integration activity to open the Activity Inspector.
2. Navigate to the process data mapping tool.
3. On a data mapping screen, select a process data mapping function from the list box.
4. Double-click the function icon to open the **Function Editor**.

   - To add a constant as additional input value, in the **Function Parameters** area, click **Add a constant parameter**. Type the constant in the field.
   - To remove a constant, select the constant and click **Remove a constant parameter**.
   - To change the order of the attributes, use the arrows to move them up or down in the list.

5. Click **OK**.

6. To add a package attribute to the function, select the package from the Source area.

7.1.15 Adding multivalue attributes

1. Double-click an integration activity to open the **Activity Inspector**.
2. In the process data mapping tool, locate a multi-value attribute.
3. Click the **Add** link next to the attribute name to create a node with a default index value.
4. To change the index value, double-click the index value.
    a. In the **Repeating Index** dialog box, select **FIRST**, **LAST**, or type a numeric value.
    b. Click **OK**.
5. Repeat these steps to specify additional values for the attribute.

7.1.16 Setting the context for a multi-value attribute

Use this procedure to set the context for a multi-value attribute with an index value of [ALL].

1. Double-click an integration activity to open the **Activity Inspector**.
2. Navigate to the process data mapping tool.
3. Select a function that can input multiple values and double-click the function icon.
4. In the **Set Repeating Context** area of the **Function Editor**, select the value for the **Input Context** as described in the following table:

Field	Description
All	Performs the function for all of the attributes at the same time and creates a single-valued attribute output. For example, the multi-value attribute IntArray is an Integer type with values [1, 2, 3, 4, 5]. The mapping rule function is Add(IntArray, 1). The result of this function is the multi-value attribute values + 1. This option performs the **Add** function against IntArray as an argument, creating a single-valued output. In this case, the output in the **Destination** area is 16 = (1+2+3+4+5) + 1.
For-each	Performs the function for each of the attributes individually and creates a multi-value attribute output. Using the previous example, this option performs the **Add** function for each individual value in IntArray, creating a multivalued attribute in the **Destination** area. In this case, the output is [2, 3, 4, 5, 6].

5. Select the value for the **Output Context** as described in the following table:

Field	Description
Over-write	Overwrites existing values.
Insert After	Inserts values after the index value.
Insert Before	Inserts values before the index value.

6. In the **Source** area, select the multi-value attribute node representing all values (the index value for this node is **[ALL]**).

7. In the **Destination** area, create a node representing a specific index value of the multi-value attribute. Select this node to complete the data mapping.

7.1.17 Adding a node to a message

Use this procedure to add an additional message node in order to complete a data mapping.

1. Double-click an email, JMS, or HTTP integration activity to open the **Activity Inspector**.

2. Configure the activity and navigate to the process data mapping tool.

3. To add a message node:

   - In an HTTP activity template, to add a URL Parameter, right-click **URL Parameters** and select **Add**.
   - In an email activity template, to add a custom email header, right-click **Additional Headers** and select **Add**.
   - In a JMS activity template, to add a JMS message property, right-click **Property** and select **Add**.

4. In the **Data Mapper Parameter** dialog box, click **Add mapping parameter**.

5. Type the **Display Name** and the **Full Name** of the property.

6. Select the **Type** of property.

7. Click **OK**.

   Ensure that any message properties you add are appropriate for the message protocol, since there is no validation for added message properties.

8. Use the added message node in your data mapping.

7.1.18 Adding a conditional node to process data

1. Right-click a multi-value item node () and select **Show Condition Builder**.
2. In the data tree, select an attribute for the expression.
3. Select an operator and type a value for the expression.
4. Click **Insert Expression**.
5. Click **OK**.

   A conditional node appears in the data tree along with the text of the expression. If the text of the expression is not visible, right-click the conditional node and select **Edit Condition** to view the complete text of the expression.

7.1.19 Mapping data from one to many nodes

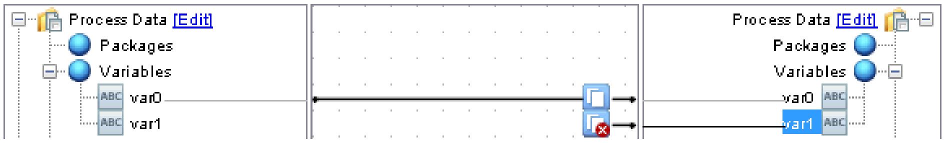
One source variable can provide input to multiple functions whose output then sets values for multiple destination variables. You can map it with a single data mapping.

For example, a date-time variable could be the source for both of the following functions:

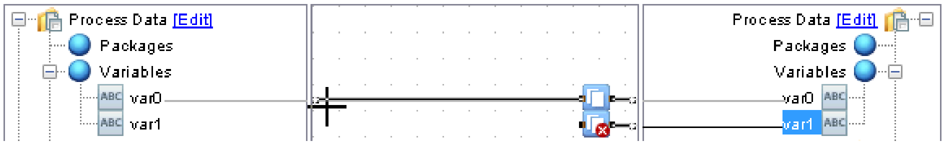
- A copy function that sets the value of a date-time destination variable
- A date-to-string function that sets the value of a string variable

The following example shows the required techniques for mapping data from one source to many destinations.

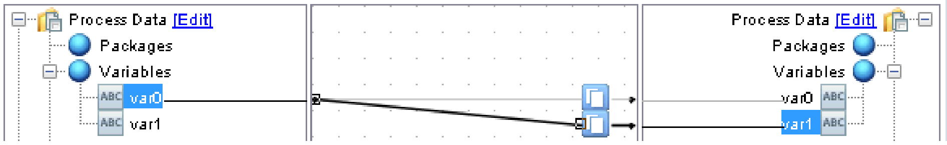
1. Create the copy function for the first item.
2. Add a second copy function and click the destination variable.



3. Click **Line drawing mode** on the toolbar.
4. Place your cursor over the connecting point for the source variable and click once.



5. Click the handle next to the copy function to complete the mapping.



7.1.20 Using an XML schema to represent a content structure

You can use an XML schema to represent the structure of content for a package or an attachment. Before completing this procedure, place the XML schema (XSD file) in the application Resources folder.

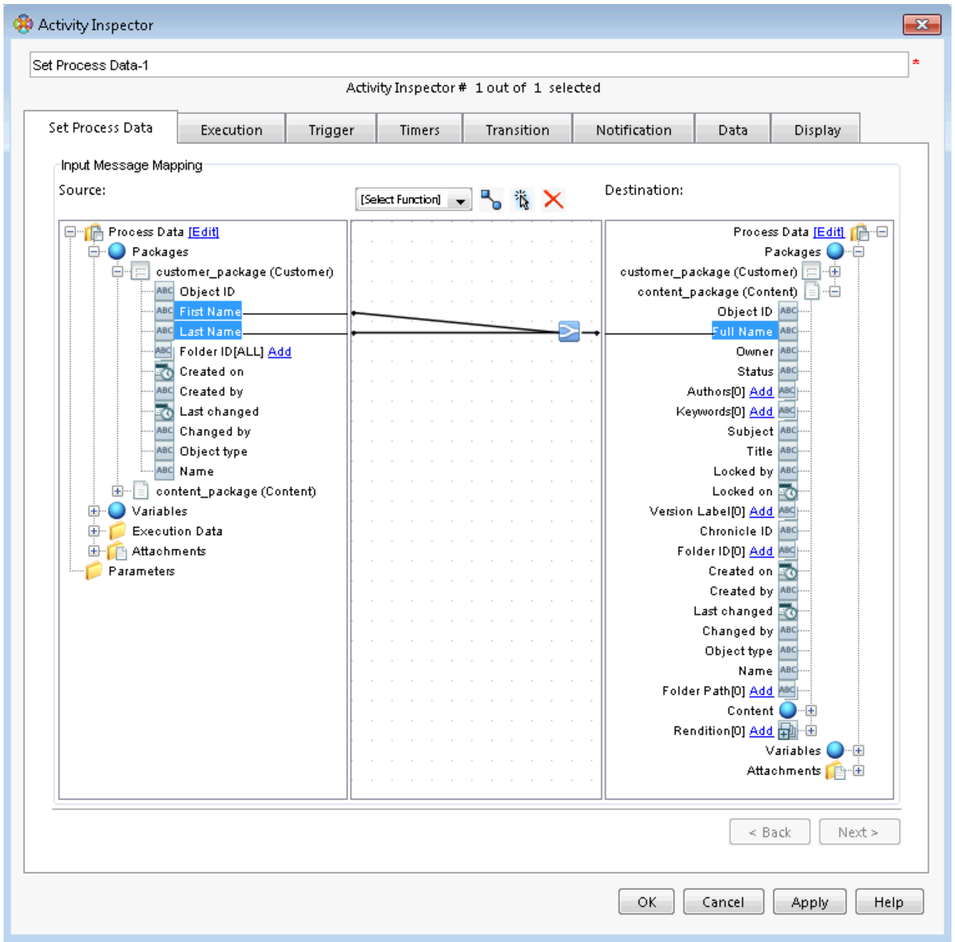
1. Right-click a **Data** node in the process data mapping tree and select **Add Element**.
2. In the **Schema** dialog box, click **Browse**.
3. In the **Select Resource** dialog box, select the schema, and click **Finish**.
4. In the **Element** field, select a root element from the XML schema and click **OK**.

   The system updates the context tree with the XML structure described by the XML schema. You can use the XML structure for data mapping.

7.1.21 Using content in a process

This task provides an example of how you can use content in a process. It shows you how to change the content name automatically to the first and last name of a Customer in the **Set Process Data** activity.

1. Create a Customer business object and add attributes for First Name and Last Name.
2. Create a content object and add an attribute for Full Name.
3. Add the Customer business object as a package to your process.
4. Add the content object as a package to your process.
5. Drag and drop the **Set Process Data** activity onto the canvas.
6. Double-click the activity to open the **Activity Inspector** to the **Process Data Mapping** tab.
7. In the **Input Message Mapping** screen **Source** area, expand your Customer business object package.
8. In the **Input Message Mapping** screen **Destination** area, expand your content object package.
9. Map the **First Name** and **Last Name** attributes of the Customer package to the **Full Name** attribute of the content object package in the **Destination** area using the **Concat** function. “Data mapping functions” on page 195 provides additional information.



10. Click **OK**.

7.2 Concepts

7.2.1 Flows

Flows connect activities in the process model and orchestrate the business logic. Flows also enable the movement of packages, process variables, properties, and dependencies that exist between the connected activities.

Arrows in a process model indicate flows. You can configure and label flows to improve the clarity of the process model.

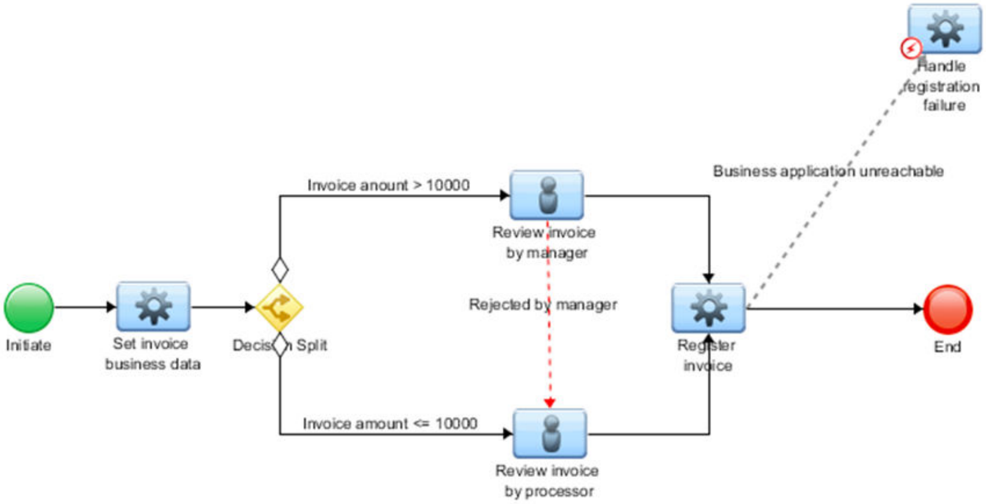
There are three types of flows:

- **Forward flows:** Advance process data from one activity to the next. For example, a package moves from an Edit activity to an Approve activity. A solid black arrow represents a Forward flow.

- **Reject flows:** Determine what happens when the performer of an activity rejects the task being routed. Reject flows direct process data in a backward loop or in an alternate forward flow. For example, a package moves from the Approve activity back to the Edit activity. A red arrow with a dashed line represents a Reject flow.
- **Fault handler flows:** Determine what action to take when an associated system activity fails. A gray arrow with a dashed line and an icon represents a Fault handler flow.

All step activities must have at least one flow coming in and one flow going out. An Initiate activity has only one outward flow and no incoming flow. An End activity can have only one incoming flow and no outward flow.

The following figure illustrates the three types of flows:

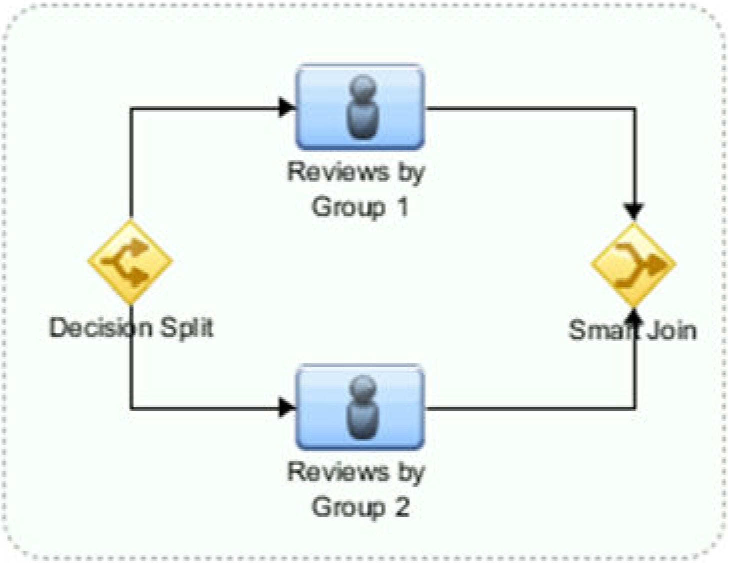

Handle registration failure
Business application unreachable
Invoice anount > 10000
Review invoice by manager
Rejected by manager
Initiate
Set invoice business data
Decision Split
Register invoice
End
Invoice amount <= 10000
Review invoice by processor

In this figure, the Reject flow is labeled *Rejected by manager*. The Fault handler flow is labeled *Business application unreachable*. All other flows are Forward flows. Diamonds at the base of the arrows indicate transition information in the previous activity. For example, the Forward flows labeled *Invoice amount* > *10000* and *Invoice amount* <= *10000* have a diamond at the base of the arrow next to the Decision split activity. The Decision split activity contains transition information for the flows to route the invoice to the correct reviewer.

7.2.2 Parallel Flows

A Parallel Flow is a contained group of activities that splits into a dynamic number of possible flows. Each triggered connection is then merged back into the main process through a Smart Join activity. The Smart Join activity automatically determines the number of active connections and waits for them to complete before merging back into the main process flow.

The following figure shows a Parallel Flow:

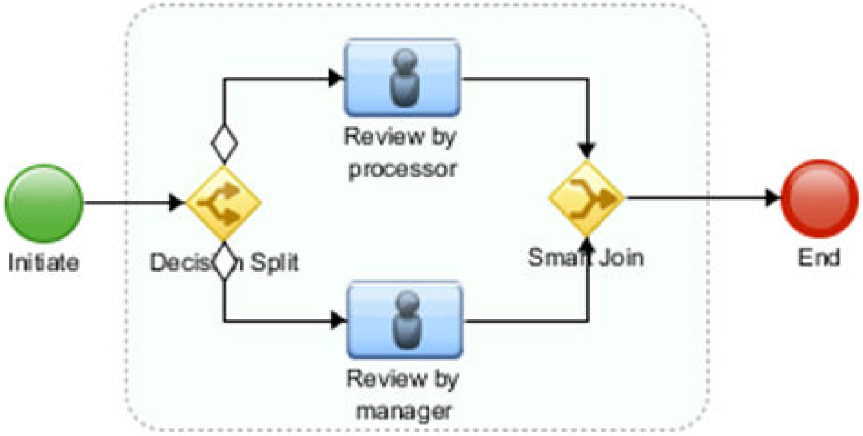


Use a Parallel Flow when you are unable to determine the exact number of input flows for a Join activity until runtime. A traditional Join activity requires you to specify the number of input flows when designing the activity. The Smart Join in a Parallel Flow contains additional intelligence to handle varying input flows.

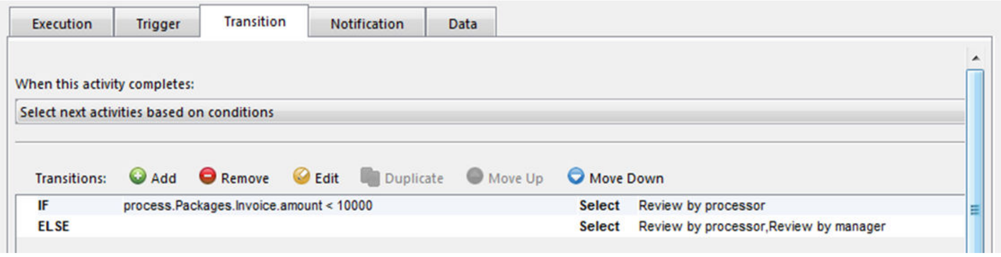
Within a Parallel Flow, you can replace or add activities between the Decision Split and the Smart Join activity. The first activity is a Decision Split activity that you can configure or replace with either a manual or a system activity. The Smart Join activity cannot be edited or removed from the parallel flow container. It is pre-configured to start when all triggered activities within the Parallel Flow are complete.

For example, in an invoice review process, a Processor reviews invoices less than $10,000. A Processor and a Manager review invoices $10,000 or greater. Since the invoice amount is not available until runtime and there can be either one or two input flows, this process uses a Parallel Flow.

The following figure illustrates the invoice review process:



In this example, the first activity (Decision Split) contains the following transition configuration:



When this activity completes:

Select next activities based on conditions

Transitions:

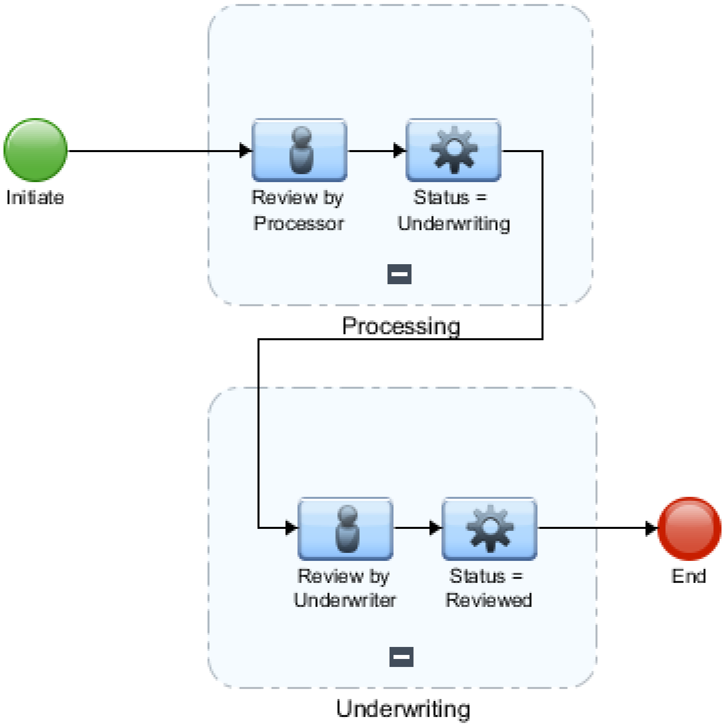
IF	process.Packages.Invoice.amount < 10000	Select	Review by processor
ELSE		Select	Review by processor,Review by manager

The Smart Join activity uses the transition information of the Decision Split activity to follow the process business logic and continue the process flow. If the process receives a $9,000 invoice, the Smart Join activity triggers when the Review by processor activity completes. If the process receives a $12,000 invoice, the Smart Join activity triggers when both the Review by processor and Review by manager activities complete.

7.2.3 Organizing process activities into groups

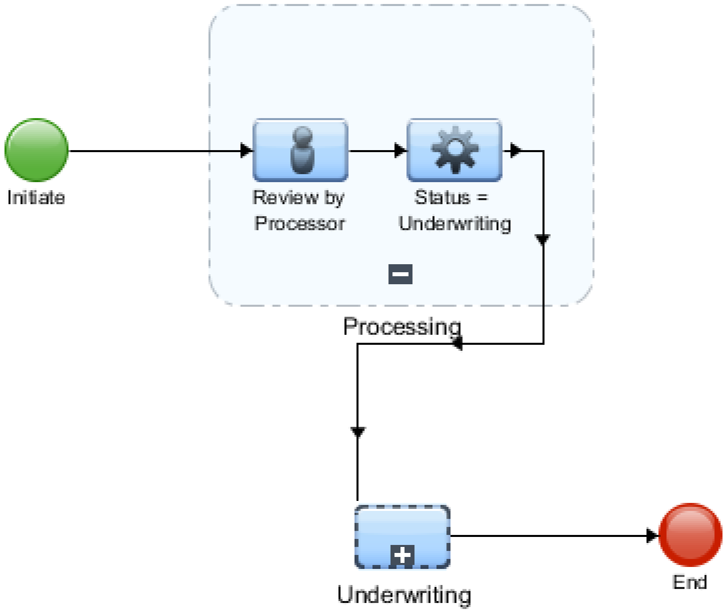
Process activity groups enable you to organize your process model visually. To simplify a process model, you can group related activities using the Group activity. You can expand a group to view the individual activities or collapse it for a condensed view.

The following figure illustrates a grouping of activities in the swim lane style:



In this example, there is an Underwriting group and a Processing group. The swim lane grouping shows the activities related to each of these functions.

The following figure shows a collapsed Underwriting group and an expanded Processing group:



Flows do not connect directly to the group container. They connect to the individual activities within the container.

7.2.4 Understanding which process icons to use

The following table describes the process model editor icons:

Icon	Name	Description
	Process Properties	Sets the properties of the process model. "Configuring general information for your process model" on page 65 provides additional information.
	Select Objects	Enables you to select individual objects in the process model.
	Straight flow	Adds a forward flow (black arrow with a single line) between two activities. "Flows" on page 182 provides additional information.

Icon	Name	Description
	Multi-Segment flow	Adds a forward flow (black arrow with multi-segment lines) between two activities. “Flows” on page 182 provides additional information.
	Create Reject Flow	Adds a reject flow (red arrow with a dashed line) between two activities. “Flows” on page 182 provides additional information.
⚠	Use this flow to assign Fault Handlers	Adds a fault handler flow (gray arrow with a dashed line and an icon) between two activities. “Flows” on page 182 provides additional information.
🗒	Create Note	Adds a visual indicator to annotate the process model. “Annotating the process model” on page 169 provides additional information.

7.2.5 Using trigger conditions to start an activity

A trigger is a action that starts an activity. Trigger conditions define the starting criteria for an activity. At runtime, the server does not start an activity until the trigger conditions of the activity are met. The trigger conditions can include a trigger signal that must occur before the activity starts.

If the activity has more than one incoming flow, you can specify how many of the previous activities must complete before this activity starts. For example, if an activity has three incoming flows, you can set the activity to start when two of the three incoming flows are complete.

Before you run the application and start the process, verify that you have defined the Trigger tab of each activity with the correct number of incoming activities.

7.2.6 Using activity timers in a process

When you configure a process activity, you can set timers to start specific actions when work is delayed or not complete. For example, you can set reminders for task performers to complete a task or start escalation processing on problems with Service Level Agreements (SLAs).

There are two types of timers for activities:

- **Process Instance Started:** If an activity has not started within a designated amount of time after the process starts, the system performs a specified action.
- **Activity Started:** If an activity has not completed within a designated amount of time after the activity starts, the system performs a specified action.

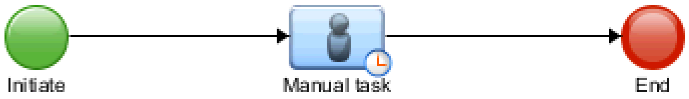
Depending upon the timer type, you can specify the following actions to begin when the timer times out:

- **Call Process:** Launches a process.
- **Call Stateless Process:** Launches a stateless process and uses the return values to update process data.
- **Set Value:** Displays the data mapping tool and enables you to update process data values.
- **Delegate Task:** Delegates the end-user task to another performer.
- **Complete Task:** Completes the current end-user task.

A timer can have multiple timeout actions. For example, after a timer completes a task, it can also set the value of an attribute on a process variable or package. The transition conditions for the activity can test for this value and change the process flow.

You can configure the timer actions to repeat at a specified interval until the timer expires. For example, you can send email notifications every two hours until the activity completes.

When an activity has a timer set for it, a small clock appears in the corner of the activity icon as shown in the following figure:

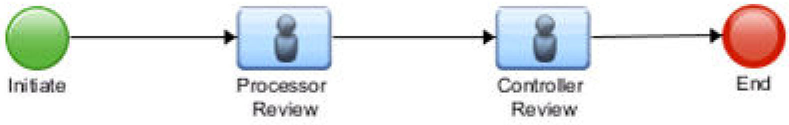


You can also configure due dates on timers that start when the activity starts.

7.2.7 Activity timer duration expression examples

A common reason to use a timer is when you have a Service Level Agreement (SLA) for a task to complete at a certain time. For example, in a claim processing application, the claim must process within 10 days. Because several tasks must complete to process the claim, the application uses two time measures for escalation. The first measure is to set a timer based on when the process starts. A manager receives a notification when the activity does not start on time. The second measure is when the activity starts but is not yet complete. During this time, a timer can provide an escalation to a manager and reassign the task.

The following figure illustrates a simple claim processing process:



The following examples use this claim processing process.

Fixed time example

In the claim processing process, a processor has three days to review and complete a claim. This process uses 3 timers to ensure that the processor reviews the task in three days. The application parameter for controlling this SLA is *Processor review SLA*.

The following table provides fixed time duration expression examples:

Timer	Duration expression (in days)
Sets a reminder for the day before the process review due date, which sends a notification to the task performer.	parameters.app.processor_review_sl - 1
Sets an alert for the day the process review is due, which goes to the task performer and a manager.	parameters.app.processor_review_sl
Reassigns the task to a manager on the day after the process review due date.	parameters.app.processor_review_sl + 1

Relative time example

In the claim processing process, the processor review activity must be complete five days after the process starts. This process uses three timers to ensure that the activity completes on time. The application parameter for this SLA is *Processing completion time SLA*.

The following table provides relative time duration expression examples:

Timer	Duration expression (in days)
Sets a reminder for the day before the processing completion due date, which sends a notification to the task performer.	differenceDays(addDays(process.process_instance.start_date,parameters.app.processing_completi - 1),now())
Sets an alert for the day the processing completion is due, which goes to the task performer and a manager.	differenceDays(addDays(process.process_instance.start_date,parameters.app.processing_completi),now())
Starts an escalation process on the day after the processing completion time due date.	differenceDays(addDays(process.process_instance.start_date,parameters.app.processing_completi + 1),now())

User-defined time example

Instead of using an application parameter, you can use an attribute on a package or process variable to determine a due date. In this case, the duration expressions in the timers use the due_date process variable.

The following table provides user-defined time duration expression examples:

Timer	Duration expression (in days)
Sets a reminder for the day before the processing completion due date, which sends a notification to the task performer.	differenceDays(addDays(process.Variables.due_date, - 1),now())
Sets an alert for the day the processing completion is due, which goes to the task performer and a manager.	differenceDays(process.Variables.due_date,now())
Reassigns the task to a manager on the day after the processing completion time due date.	differenceDays(addDays(process.Variables.due_date, 1),now())

7.2.8 Using transitions to specify the next process activities

Transitions determine the next activity in a process. With multiple outgoing flows, you can configure the transition so that only selected activities occur based on the outcome of the activity. For example, you can enable a form design reviewer to forward the design to the next reviewer or send it back to the designer for revision. You set up this branching logic by creating flows from the first activity to the two possible subsequent activities. You then allow the performer to decide which path to follow.

The transition selected in an activity defines the next activities in the process flow when the activity completes at runtime. There are three types of runtime transitions:

- **Select all connected activities:** The system routes the process flow to all the subsequent activities.
- **Let performer select the next activities:** The performer of the activity determines the next activities in the process flow.

- **Select next activities based on conditions:** The system routes the process flow to the subsequent activities by evaluating the transition conditions in the activity.

When an activity has multiple outgoing flows with branching logic, it represents a decision point in the overall business process. To show the decision point clearly in the process model, you can insert an explicit Decision Split activity into the flow. Instead of configuring transitions in the current activity, connect the activity to a single Decision Split activity and configure the transitions in the Decision Split activity.

Determining transition conditions

Transition conditions enable you to define activities that route tasks differently depending on the results of the activity. A transition condition is a logical condition with one or more associated flows. When an activity completes at runtime, the system evaluates its transition conditions to determine which subsequent activities to start as the next step in the process. It moves the process forward to the activities associated with the first transition condition that is TRUE. An activity can have multiple transition conditions, although the server only selects the first TRUE one at runtime.

When you use transition conditions, include an Else action. If none of the transition conditions are TRUE, the system uses the Else transitions.

7.2.9 Activity notifications

The system sends a OpenText Documentum CM notification and an email notification to an end user when one of the following events occurs in an activity:

Event	User who receives the notification
When an automatic task fails	Workflow supervisor
When a user task is delegated	Delegated user
When a user task is created	Task performer

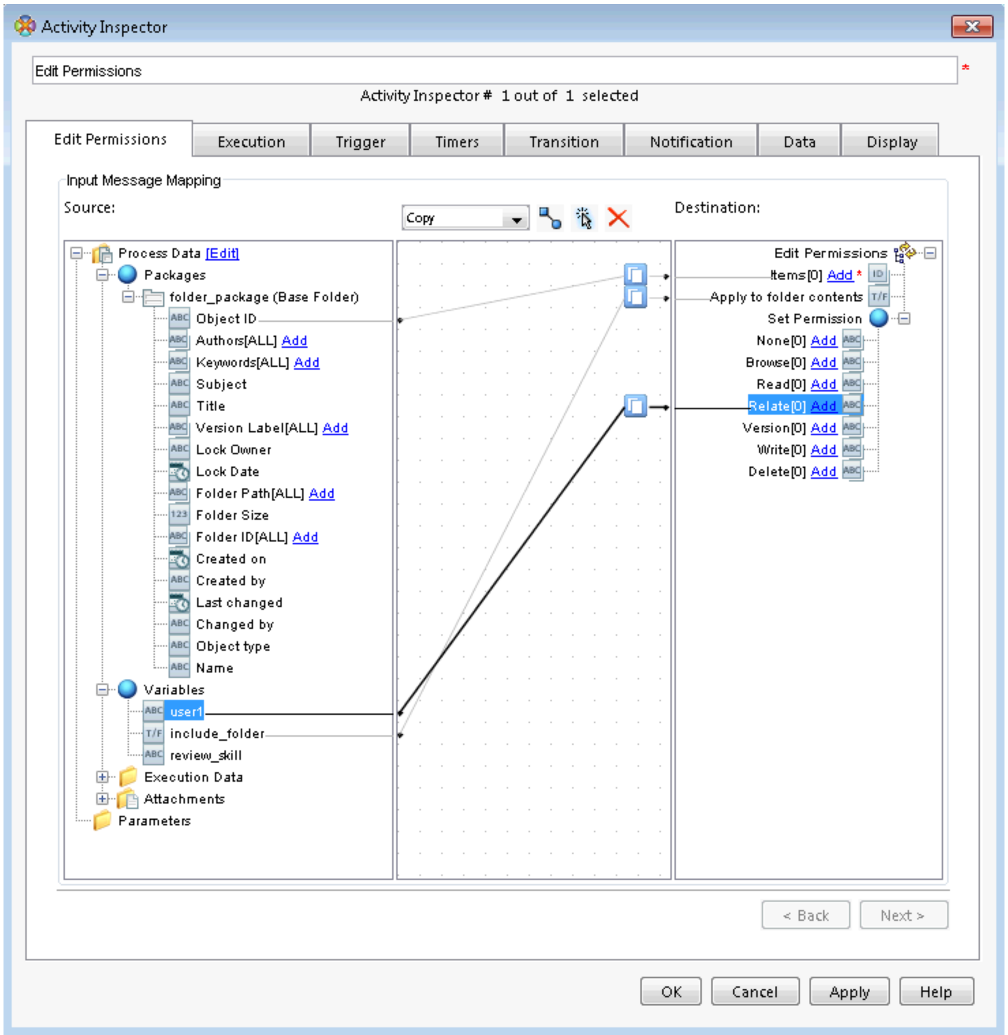
Instead of using the system email notifications, you can create custom email notifications using an email template. The email server configured in the repository configuration sends the email notifications.

Notifications configured in the process properties of the process model override system notifications.

7.2.10 Mapping data within activities

The process data mapping tool provides a graphical interface that simplifies the process of passing process data. The Source area is on one side of the screen and the Destination area is on the other side of the screen. You can map process data or attributes in the Source area to server or process attributes in the Destination area. These attributes can include process method arguments, web service parameters, database query return values, and attributes specific to services such as JMS, HTTP, and FTP.

The following figure shows the process data mapping tool:



Edit Permissions

Activity Inspector # 1 out of 1 selected

Source: Copy

Many automated activities require mapping information from one source of data to another. For example, a loan-origination process includes an activity that calls a web service to look up the credit score of a customer. The activity passes the social security number from the loan application package to the web service. The activity then copies the returned credit score into another package attribute so that it is available to subsequent activities. You can map data from package attributes to the web service input parameters and from the web service output message to package attributes.

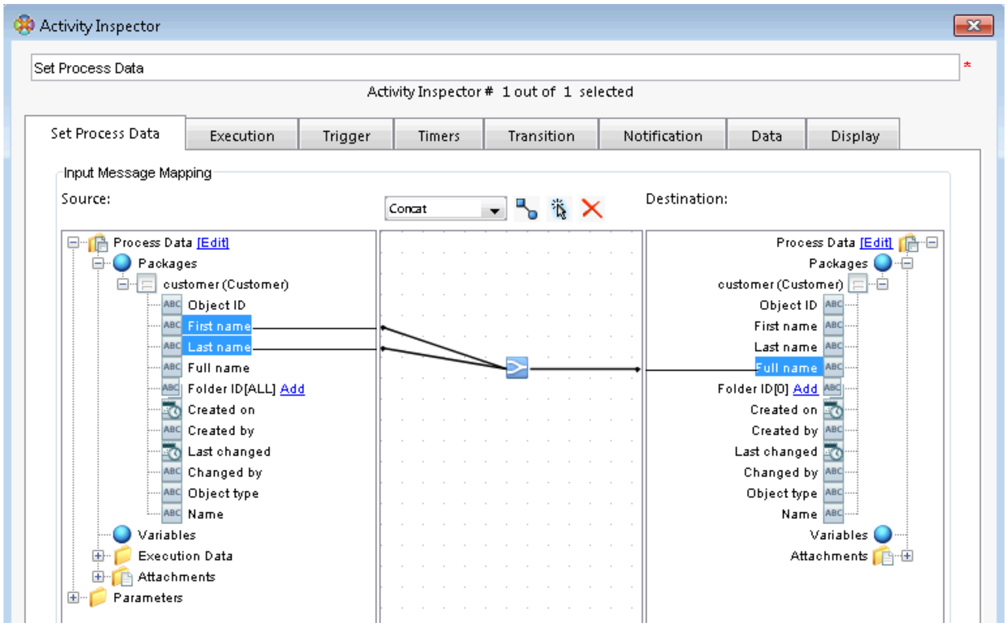
The contents of the Source and Destination areas depend on the type of activity you configure. For activities that require input values, the Source area typically shows the attributes for all business process packages, process variables, and the runtime execution variables. Runtime execution variables, such as the supervisor name, are available at runtime.

The center area displays the functions used to transfer data from the Source area to data in the Destination area. The mapping tool enables you to copy values directly from one source of data to another. You can also perform data type conversions, concatenate strings, perform mathematical operations on numbers, and include constant values.

7.2.11 Process data mapping

In the process data mapping tool, the center area contains icons representing process data mapping functions. It also contains lines connecting the function icons to their input arguments and output destinations. At runtime, the activity passes the values of the input arguments to the function and saves the result as the value of the destination attribute.

The following figure illustrates data mapping in the process data mapping tool:



Activity Inspector

Set Process Data

Activity Inspector # 1 out of 1 selected

Concat

You create one mapping function at a time. The process data mapping tool requires you to complete one mapping (by selecting its input parameters and output destination) before starting on the next mapping. If the data type of an attribute does not match the data type that the function expects, the connection appears as a dashed line. When the data types match, the connection appears as a solid line.

If a data mapping function does not have all of its required arguments, a red X appears on the function icon , indicating that the function is incomplete.

You can double-click a function icon to open the Function Editor. It displays the name of the function, its syntax, and a list of the input values. The Function Editor enables you to:

- **Modify the order of the attributes:** For a function that accepts multiple input arguments, the list of input arguments follows the order of your source attribute selections. You can change the order of these attributes.
- **Use constant input values instead of attributes:** Some functions have no attributes as inputs because all of their input values are constants.

  Instead of selecting attributes in the Source area, use the Function Editor to add the constant input values.
- **Set repeating context:** When the input argument for a function is a multivalue attribute, the function can process the values individually or all at one time. When the function writes its result into the destination attribute, the function can overwrite existing values or add new attribute values. You can specify how to write these values in the Function Editor.

7.2.12 Data mapping functions

A data mapping function operates on data from a source and saves the result to a destination. Source and destination are nodes of the process data available to the integration activity where the mapping takes place.

A function icon represents the function in the process data mapping tool. When double-clicked, the function icon opens to the Function Editor. The Function Editor displays the name of the function, its syntax, and a list of the expected input values. The names of the attributes linked to the function appear in XPath format. If the value is based on an enumeration set, you can view a list of the enumeration values of the destination node of the function.

Data mapping functions are described in the following table:

Function	Input arguments	Result
Add	Two or more numbers.	Sum of the input arguments.
Add Business Day	An integer date value, a string for the calendar, and an integer for the noOfDays.	Adds a business day to the noOfDays value. The value for a business day is based on the selected business calendar.

Function	Input arguments	Result
Add Days	An integer date value and an integer for the noOfDays.	Returns a date after adding the specified number of days to the date.
Byte To String	Two strings, the first representing the binary data, and the second specifying its encoding value, such as UTF-8, UNICODE, and so on. Default encoding value is UTF-8.	Data as a string.
Concat	Two or more strings.	Concatenated string consisting of the input arguments in order.
Copy	One argument of any type.	Unchanged input argument.
Count	(Object param[])	Returns the number of values in the multi-value input. For single value inputs, the return is 1.
Date to String	A date and a string representing a valid date pattern. The date pattern must conform to the standard Java SimpleDateFormat. For details, refer to the Java API and developer reference documentation.	Date value as a string with the specified pattern.
Divide	Two or more numbers.	Result of dividing the first input argument by the second argument. When there are more than two arguments, each subsequent number is used to divide the previous result.
Evaluate Expression	One or more arguments of any type.	Return type depends on the input argument type. For example: `power (2.0,2.0)` returns 4.0 because the input parameter type is float.
Get Day	Integer.	Returns an integer that represents the day segment of the date.
GetEmailAddress	String.	Queries dm_user to return an email address for a user.
Get Month	Integer.	Returns an integer that represents the month segment of the date.

Function	Input arguments	Result
Get Ticket	String.	Generates a login ticket to the given username at runtime. Uses the following syntax: username, scope, time out, single use, and server
Get Value	String parameter that specifies the object and the index position number.	Returns a value from a specified position in the index.
Get Year	Integer.	Returns an integer that represents the year segment of the date.
Join	Two or more string arrays.	Creates a join of the selected inputs.
Multiply	Two or more numbers.	Product of the input arguments.
Split	String that can include an optional index position value.	Returns a repeating string or a position in the repeating value if the optional index position is used.
String To Byte	Two strings, the first representing the data, and the second specifying its encoding value, such as UTF-8, UNICODE, and so on. Default encoding value is UTF-8.	Binary data.
String to Date	Two strings, the first giving a date and the second specifying its pattern. The date pattern must conform to the standard Java SimpleDateFormat. This function supports MMMM format in the English locale only. For details, refer to the Java API and developer reference documentation located on the Sun developer website.	Value of Date data type.

Function	Input arguments	Result
Substring	A string, a number representing how many of the initial characters to remove from the string, and optionally a number representing the position of the last character to include in the substring.	String consisting of characters from the first input argument, starting from the specified start position and ending at the specified end position. For example, if the input arguments are `unhappy` and `2`, the result is the string `happy`. If the input arguments are `unhappy`, `2`, and `5`, the result is `hap`.
Subtract	Two or more numbers.	Result of subtracting the second number from the first number. When there are more than two numbers, each subsequent number is subtracted from the previous result.
ToLower	String.	Converts the string to lowercase letters.
ToUpper	String.	Converts the string to uppercase letters.

7.2.13 Evaluate Expression

Evaluate expression function enables you to run complex expressions in a data mapper process. It simplifies function chaining and conditional evaluation in data mapper process using expressions. You can define complex expressions in a single data mapper function. Evaluate expression function can be a simple expression or a complex set of functions that are available in the Expression editor.

Evaluate expression function accepts string as the first parameter and a list of objects as subsequent parameters. The return type of the function depends on the expression and the input parameters. If the input parameter is float, the function returns a float value. For example: **power** (2.0,2.0) will return 4.0 because the input parameter type is float.

You can also use the functions available in the Advanced Workflow Expression Editor to create an expression. Functions that accept arrays or lists as input parameters are not supported in the Evaluate Expression function. You can use multiple functions to create an expression using the objects as parameters.

Example:

```
ifThenElse([x] == 1 and [y] == 0, 'high', 'low')
power(max([ab1], [cd_]), [xy]) - 1
floatToInt(avgInt(stringToInt('1'), stringToInt('2')))
```

Notes

- If an expression uses same parameter multiple times, you need to provide parameter value only for the first instance.
- You can validate the syntax of an expression during the design time. (Negative) Logical validation of an expression is not currently available.
- The parameter name can contain alphabets, numbers, and underscore but cannot start with a number.

7.2.14 Multi-value attributes

Multi-value attributes are attributes that contain more than one value. For example, the Authors attribute of the content object is a multi-value attribute since content can have more than one author.

The process data mapping tool supports multi-value as well as single-value attributes. Use multi-value attributes when you want to input multiple values into your process data mapping functions. You can identify a multi-value attribute by the Add link available next to its name. The Add link enables you to specify separate values for the attribute. These values are index values (for example, Authors [1] and Authors [2]).

The following table describes the attribute nodes with index values:

Attribute node	Description
<<*attribute name*>>[All]	Points to the complete set of values for that attribute. For example, Authors[ALL] specifies all authors in the set.
<<*attribute name*>>[<*n*>]	Indicates a specific index position for the attributes. For example, Authors[0], Authors[1], and Authors[2].
<<*attribute name*>>[FIRST]	Indicates the first index position for the attribute. For example, Authors[FIRST] is the first author listed.
<<*attribute name*>>[LAST]	Indicates the attribute at the end of any other existing attributes. For example, Authors[LAST] is the last author listed.

Multi-value attributes with the index [ALL] are located in the Source area of the process data mapping tool. They are not found in the Destination area. When mapping a multi-value attribute with the index [ALL], you also have to specify the context for the multi-value attribute. The context provides information on how the process function handles the inputs.

7.2.15 Using multi-value attributes as input to functions

In process data mapping, a function can consume the values of a multi-value attribute using the following input context options:

- **All**: The function performs its action on all of the values at the same time and creates a single-valued attribute output. For example, if you map a multi-value attribute whose values are integers to an Add function, the output is the sum of all the input values.
- **For-each**: The function performs its action once on each individual value to produce multiple outputs. For example, the multi-value attribute LoanValue is an integer type with values 100,000, 200,000, and 300,000. If you use the Multiply function to multiply the values by 0.10, the output is 10,000, 20,000, and 30,000.

You can double-click a function icon to open the Function Editor. The Function Editor enables you to configure the input context options.

7.2.16 Specifying conditions for mapping data

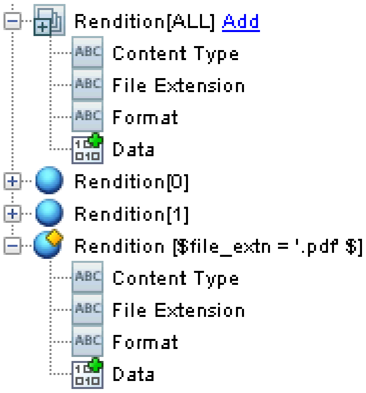
A multi-value item node (⊞) with the index ALL is a pointer to a complete set of attributes. A child node with an integer index points to a single item in that set. In addition to the nodes with index values, you can add nodes based on conditions. A conditional node (●) has its value set by the system when a specified condition becomes true.

For example, the Workitem[ALL] node is a list of objects that contain data about activities. A process activity called MyLastActivity requires data about an activity called MyFirstActivity. In the process data of MyLastActivity, you can create a conditional Workitem node whose value is set when the condition `act_name = MyFirstActivity` is true. Data about MyFirstActivity can then be mapped from the conditional node.

You can add conditional nodes to any multi-value item nodes, including:

- Workitem[ALL] nodes located within an Execution Data node
- Rendition[ALL] nodes located within a content package and within an Attachments node

The following figure shows a conditional node added to a multi-value item node:



7.2.17 Login tickets

Login tickets allow end users to connect securely to the repository without having to provide credentials. You can configure the ticket to access one or more repositories, to allow more than one use, and to expire after a specified period.

For example, a process activity sends an email to an end user requesting an information update. The email contains a link to the content in the repository. The login ticket integrates into the link. When the end user clicks the link to access the content in the repository, the end user does not have to log in.

Use the Get Ticket function within the process data mapping tool to generate a login ticket for a specific end user. The performer of the activity needs superuser privileges. Only a user with superuser privileges can grant superuser privileges.

Chapter 8

Migration activities

8.1 Tasks

8.1.1 Adding a deployment environment

Your Advanced Workflow system or application administrator has the environment connection details for adding an Advanced Workflow deployment environment. The *OpenText Documentum Advanced Workflow Deployment Guide* contains details on deployment environments.

1. On the toolbar, click **Preferences**.
2. Select **Deployment > Deployment Environments**.
3. Click **Add**.
4. In the **Add Deployment Environment** dialog box, specify the connection details for the application server as described in the following table:

Field	Description
Protocol	Select one of the protocols—HTTP or HTTPS. If you choose to set SSL mode for the deployment environment, select the HTTPS protocol.
Environment name	Type the environment name.
Hostname	Type the application server IP address.
Port number	Type the application server port number.
Username	Type the deployment username on the application server.
Password	Type the password for the deployment username on the application server.
Deployment Agent (xDA) Environment	Lists the active xDA environments that can be used to deploy the application created using Advanced Workflow.

   Click **Test Connection** to verify the connection.

5. Click **Finish**.
6. To edit environment details:

   a. In the **Preferences** dialog box, select **Deployment > Deployment Environments**.

b. Select the environment and click **Edit**.

c. Edit the environment details and click **Finish**.

d. Click **OK**.

8.1.2 Adding a design-time environment

You can use a design-time environment to run your application in a preview mode, debug your process, or adopt existing types from a repository.

The design-time environment is not part of application deployment.

1. On the toolbar, click **Preferences**.

2. Select **Deployment > Design-time Environments** and click **Add**.

3. In the **Add Design-time Environment** dialog box, specify the repository details as described in the following table:

Field	Description
Docbroker Hostname	Type the repository IP address.
Docbroker Port	Type the repository port number.
Repository Name	Click **Refresh** and select the repository.
Username	Type the repository username to authenticate the connection.
Password	Type the password to authenticate the connection.
Domain	Type the domain name to authenticate the connection.

(Optional) Click **Test Connection**.

4. Click **Next**.

5. Select the **Specify Global Registry Repository** check box and specify details as described in the following table:

Field	Description
Docbroker Hostname	Type the global registry repository IP address.
Docbroker Port	Type the global registry repository port number.
Repository Name	Click **Refresh** and select the global registry repository.
Username	Type the username on the global registry repository to authenticate the connection.

Field	Description
Password	Type the password to authenticate the connection.
Domain	Type the domain name to authenticate the connection.

(Optional) Click **Test Connection**.

6. Specify the **Environment name** and **Description**.
7. Set this environment as a default environment, if required, and click **Finish**.
8. To edit the environment details:

   a. In the **Preferences** dialog box, select **Deployment > Design-time Environments**.

   b. Select the environment and click **Edit**.

   c. Edit the environment details and click **Finish**.

   d. Click **OK**.

8.1.3 Creating a run configuration

1. On the toolbar, click **Preferences**.
2. Select **Deployment > Run Configurations** in the **Preferences** dialog box.
3. Click **Add**.
4. In the **Add Run Configuration** dialog box, specify basic settings as described in the following table:

Field	Description
Name	Type a name for the configuration.
Environment	Select the deployment environment for the configuration.
Deployment method	Select **Full** to deploy the entire application. Select **Incremental** to deploy only application components that changed since the last deployment or new application components (for example, an updated business object model or a new application page).

Field	Description
Data Policy	The data policy defines what happens to application data in the environment when you run a new version of the application. Examples of application data include runtime instances of business objects, content objects, processes, and reports. Select **Clean** to delete application data. Select **Minimal** to update application data associated with application components that changed since the last deployment. Select **Maintain** to preserve application data. Running the updated application does not affect existing application data in the environment. **Best practice**: Use **Maintain** when there is an existing production deployment of the application you are working on and you want to change the application. Setting the deployment to use the **Maintain** mode ensures you capture any data upgrade issues before you deploy to the production environment. Use a representative set of test data in your development environment to accurately identify any issues that might occur in production when you try to redeploy the upgraded application. An environment set to development mode supports all the data policies. An environment set to production mode supports only the **Maintain** policy. If you have test data that is important, create test loader scripts using API, DQL, or Java DFC code to load the data. If your data policy is set to **Minimal** or **Clean**, you could lose the test data, in which means that you need an automated way to re-populate the data.
Description	Type a description of the configuration.
Build application	Perform one of the following: • *Always*: All deployment artifacts are generated every time. • *Only when modified*: The deployment artifacts are not generated if there are no changes in the application.

Note: If you run a previously deployed application in an environment set to development mode and the application shares a library with another application in the environment, the system deletes the data for both applications when the deployment method is **Full** and the data policy is **Clean**.

5. Click **Next**.
6. To assign different endpoint and parameter values to the configuration:
   a. In the **Setting** column of the dialog box, expand the node for the endpoint or parameter.
   b. In the **Data** column, double-click the cell for the value that you want to change.
   c. Type a new value.
7. Click **Finish**.

8.1.4 Editing a run configuration

1. On the toolbar, click **Preferences**.
2. Select **Deployment > Run Configurations** in the **Preferences** dialog box.
3. To edit basic settings only:
   a. Click the cell for the setting and type or select a new value. Go to step 5.
   b. To edit endpoint and parameter values:
      i. Select the run configuration from the list and click **Edit**.
      ii. Click **Next**.
      iii. In the **Setting** column of the dialog box, expand the node for the endpoint or parameter.
      iv. In the **Data** column, double-click the cell for the value that you want to change.
      v. Type a new value.
      vi. Click **Finish**.
4. Click **OK**.

8.1.5 Packaging an application for a later deployment

1. On the toolbar, click **Package**.
2. In the **New application package** dialog box, specify the folder for the application package in one of two ways:
   - Type the path to the folder in the **Browse Folder** text box.
   - Click **Select Folder** to browse available folders. Select the folder for the application package. Click **Select Folder**. The **New application package** dialog box displays the path to the selected folder in the **Browse Folder** text box.
3. Click **Finish**.

   The system validates the application. If the application is invalid, an error message is displayed. If the application is valid, the system compiles the application, creates an application package, and displays the path to the application package in the **New application package** dialog box.
4. Click **OK** to close the **New application package** dialog box.

You can provide the application package location to your Advanced Workflow application or system administrator. The application or system administrator uses the xDA to deploy the application package to a staging or production environment.

8.1.6 Packaging using Apache Maven

Ensure that you follow these requirements:

- Install Apache Maven 3.6.0. For CLI packaging to work in offline mode, Apache-Maven 3.6.0 is the supported version.
- Set JAVA_HOME to the directory where JDK 11.0.9 is installed. For example, choose JDK 11 32-bit for a Advanced Workflow 32-bit application and JDK 11 64-bit for a Advanced Workflow 64-bit application.
- Set up Maven so that it is accessible from the console.

1. Browse to the Advanced Workflow application directory in your file system.
2. Set up a Maven repository and add the repository to Advanced Workflow application.
3. Edit the POM file for each project and add the `<repository>` ,`<pluginRepository>`, elements. Add the `<vmArgs>` element to the import and validate sections of the POM file:

```
<repositories>
  <repository>
    <id>Builder</id>
    <url>http://localhost:2910/maven</url>
   </repository>
    <repository>
    <id>Builder-offline</id>
    <url>file:///C:/.../advwfProcessDesigner/maven/designer</url> (Path to Maven
directory)
   </repository>
   <!--  internal repositories go here -->
  </repositories>
-----------------------------------------------------------
  <pluginRepositories>
   <pluginRepository>
  <id>Builder</id>
    <url>http://localhost:2910/maven</url>
   </pluginRepository>
    <pluginRepository>
   <id>Builder-offline</id>
   <url>file:///C:/.../advwfProcessDesigner/maven/designer</url> (Path to Maven
directory)
    </pluginRepository>
 <!--  internal plugin repositories go here -->
  </pluginRepositories>
------------------------------------------------------------
```

4. Edit the `settings.xml` file in Maven and add the `<mirror>` element:

```
<settings xmlns=" http://maven.apache.org/SETTINGS/1.0.0 "     xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance "     xsi:schemaLocation=" http://
maven.apache.org/SETTINGS/1.0.0    http://maven.apache.org/xsd/settings-1.0.0.xsd
">
  <profiles>
    <profile>
      <id>advwfProcess-designer</id>
      <properties>
        <designerPath>C:\advwfProcessDesigner</designerPath>
      </properties>
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>advwfProcess-designer</activeProfile>
  </activeProfiles>
  <mirrors>
    <mirror>
      <id>Builder-offline</id>
      <url>file:///c:/advwfProcessDesigner/maven/designer</url>
      <mirrorOf>Builder</mirrorOf>
     </mirror>
  </mirrors>
</settings>
```

Ensure that the `settings.xml` is available under `<home-directory>/.m2` folder.

In case you have performed step 3, you can skip this step.

5. To increase the heap size, add the <vmArgs> element to the POM file.

```
   <!--  To increase the heap size, add this element <vmArgs>-->

      <plugin>
         <groupId>com.emc.advwfProcess.builder</groupId>
         <artifactId>advwfProcess-import-project</artifactId>
         <version>1.0.4</version>
      <configuration>
          <mavenRepoPath>maven/designer</mavenRepoPath>
            <vmArgs>-Xms512m -Xmx768m -XX:PermSize=256m
                                         -XX:MaxPermSize=512m</vmArgs>
        </configuration>
      </plugin>
      <plugin>
         <groupId>com.emc.advwfProcess.builder</groupId>
         <artifactId>advwfProcess-validate</artifactId>
         <version>1.0.10</version>
         <executions>
```

```
            <execution>
               <id>advwfProcess-validate</id>
               <goals>
                  <goal>run</goal>
               </goals>
            </execution>
         </executions>
         <configuration>
          <dir>${basedir}/Artifacts</dir>
          <mavenRepoPath>maven/designer</mavenRepoPath>
            <vmArgs>-Xms512m -Xmx768m -XX:PermSize=256m
                                         -XX:MaxPermSize=512m</vmArgs>
        </configuration>
      </plugin>
```

6. Save your changes.

7. Create a new POM.xml file at the same `Applications` folder level:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:
                                                            xsi="http://www.w3.org/
2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                                                            http://
maven.apache.org/xsd/maven-4.0.0.xsd">
            <modelVersion>4.0.0</modelVersion>
            <groupId>cactus</groupId>
            <artifactId>parent</artifactId>
            <version>1.0.0</version>
            <packaging>pom</packaging>

            <modules>
                        <module>../Project_New_Customer_On_Boarding</module>
                        <module>../Customer_Services</module>
                        <module>../Casual_and_Property_Insurance</module>
            </modules>

            <build>
                   <plugins>
                        <plugin>
                             <groupId>com.emc.advwfProcess.builder</groupId>
                             <artifactId>advwfProcess-import-project</artifactId>
                             <version>1.0.4</version>
                             <configuration>
                               <vmArgs>-Xms512m -Xmx768m -XX:PermSize=256m
                                              -XX:MaxPermSize=512m</vmArgs>
                             </configuration>
                         </plugin>
<!--Below is an example of a plugin declaration that is used by all the modules
that this pom encompasses. Additional plugins can be added as required. -->
                       <plugin>
                              <artifactId>maven-deploy-plugin</artifactId>
                              <version>2.5</version>
                              <executions>
                                 <execution>
                               <id>default-deploy</id>
                               <phase/>
                              </execution>
                               </executions>
                       </plugin>
                    </plugins>
            </build>

            <pluginRepositories>
                        <pluginRepository>
                               <id>Builder</id>
                               <releases>
                                        <enabled>true</enabled>
                                        <updatePolicy>never</updatePolicy>
```

```
                                </releases>
                               <snapshots>
                                         <enabled>true</enabled>
                                         <updatePolicy>never</updatePolicy>
                               </snapshots>
                                  <url>file:///C:/tmp1/BuildZips/x5290/
advwfProcessDesigner/
                                                                     maven/designer
                                  <!--Designer maven offline location--></url>
                         </pluginRepository>
            </pluginRepositories>
</project>
```

This step is required only when there are multiple modules consisting of application and various projects.

8. Run the `mvn advwfProcess-import-project:run` command to import projects to your local machine.

9. Navigate to the newly created POM file at command prompt and run the following command:

```
mvn install "-DdesignerPath=<path_advwfProcess_Process
```

Maven creates .WAR and .config.xml files for the application created using Advanced Workflow.

10. You can also run the following maven commands:

Syntax

```
mvn [goals | phases] [options]
```

Arguments

Argument	Type	Description
advwfProcess-import-project:run	Goal	Imports the project to your workspace in local machine. You may get an error if the project is already available in your workspace.
validate	Phase	Validates the project for errors and ensure relevant information is available.
generate-sources	Phase	Generates a source code of a project.
package	Phase	Compiles the WAR file of an application created using Advanced Workflow.

<table>
<thead>
<tr><th>Argument</th><th>Type</th><th>Description</th></tr>
</thead>
<tbody>
<tr><td>deploy</td><td>Phase</td><td>Deploys the Advanced Workflow application to the environment. It has two arguments:
<ul>
<li><b>envConfigFile</b>: Specify the relative or absolute path to environment configuration properties files. Add these lines to this file:
<pre><code># dataPolicy: CLEAN, MINIMAL,
                                MAINTAIN
# deploymentMethod:
INCREMENTAL,
FULL
deploymentService=com.emc.advwfProcess
    builder.deployment
    XMSDeploymentService
dataPolicy=MINIMAL
deploymentMethod=INCREMENTAL
xda.host=
xda.port=
xda.username=
xda.password=
xda.environmentName
xda.protocol
repositoryServicesOnly=true

//repositoryServicesOnly is an
        optional parameter.
        Set to true if you need to
        perform a model-only deployment.
        Set false for deployment to
        all endpoints including the
        application host
        Default value is false.</code></pre></li>
<li><b>appConfigFile</b>: Specify the Project relative path or absolute path to application configuration XML file.</li>
<li><b>-Dorg.eclipse.equinox.http.jetty.http.port</b>: Default port number is 2910. Configure the port number to use for the internal Jetty server connection. This is a mandatory parameter for mvn cli deploy command.</li>
</ul>
<b>Note:</b> It is recommended not to have Advanced Workflow running while you execute the maven commands.</td></tr>
<tr><td>install</td><td>Phase</td><td>Installs the package in the local repository.</td></tr>
<tr><td>clean</td><td>Phase</td><td>Remove all projects generated by the previous build.</td></tr>
</tbody>
</table>

Argument	Type	Description
-D<key>=<value>	Option	Defines system properties: • (Mandatory) designerPath- Specify the path of Advanced Workflow where it is installed. • (Optional) workspace- Specify the path of workspace. • –DoutputDirectory- Specify the location of the output directory to which you can export the multiple project JAR files of the application.
-X	Option	Shows debug information.

11. To obtain the version of the project, use the following command:

```
mvn org.apache.maven.plugins:maven-help-plugin:2.1.1:evaluate
                                 -Dexpression=project.version
```

    You can parse the output of the command to fetch the value of the project version.

    The Apache Maven site provides information about Maven goal plug-ins and phases.

8.1.7 Adopt object models from repository

Ensure the repository endpoint is configured to perform this activity.

1. In the **Migration Navigator > Object Models**, right-click any of the object model and click **Adopt from Repository**.

2. In the **Adopt from Repository** dialog box, select a design-time environment. The system populates the name of the design-time environment.

3. Select the name of a project to which you want to import the types from the repository. Click **Next**.

4. Specify the required business objects, content, cabinet, and folder types you want to adopt.

   When you adopt the subtype of dm_sysobject, the system allows you adopt it either as a Business Object or Content model. After the data type is adopted, you cannot adopt it to another type, so use your discretion before you select the data type for adoption.

   When an object that is a subtype of dm_sysobject is adopted as a Content, the system allows you to version and attach the content to that object.

5. Click **Finish** and then click **OK**.

   On successful operation, the system displays the adopted data types. After the data type is adopted to a specific application, the system does not allow the same type to be adopted again.

8.1.8 Migrating an application from OpenText Documentum CM xCP Process Only 22.2 to Advanced Workflow 22.4

Advanced Workflow does not provide support to create project and library. Hence, when you migrate an application to Advanced Workflow 22.4, you must move all the artifacts from projects and libraries to an application. Complete these steps to migrate an application from Documentum xCP Process Only 22.2 to Advanced Workflow 22.4:

1. Open Documentum xCP Process Only 22.2 and move all the required artifacts from projects to application using the **Move** option.
2. Rename the application to `Advanced_Workflow` and change the namespace to `daw`.

   Note: In Advanced Workflow 22.4, application name and namespace is read-only and set to the following default values:

   - Application name: Advanced_Workflow
   - Namespace: daw
3. Navigate to the **Application** tab, select an application model, and click **Export**.
4. In the **Export Application** dialog box, select a destination folder.
5. Click **Finish**. After the application is exported successfully to the selected folder, the **Packaging Application** dialog box appears, click **Ok** to finish the application export process.
6. Open Advanced Workflow 22.4 and click **Import Application**.
7. In the **Import Application** dialog box, click **Browse** and navigate to the application folder to import.
8. Select the **Import and copy** option.
9. Click **Finish** to import the application.

   Notes

   - After you import and deploy an application, Advanced Workflow creates a new application and objects in the repository corresponding to the imported artifacts. You can identify the new objects using the *`dm_process`* namespace attribute.
   - You must deprecate the old application and objects after migrating and deploying the application to Advanced Workflow 22.4. You must also reconfigure the other OpenText Documentum CM applications to use the migrated objects. For example, reconfigure your D2 application to point to new objects deployed through Advanced Workflow.

8.1.9 Migrating types from Documentum Composer

You can migrate Types created using the Composer application to the Advanced Workflow.

This procedure applies to Business Object, Shareable Business Object, Lightweight Business Object, Folder, Content, Shareable Content, Lightweight Content, Folder, and Cabinet models.

1. In the **Migration Navigator** dialog box, right-click **Composer Types** and select **Migrate from Composer Project**.
2. In the **Migrate from Composer Project** dialog box, specify the properties as described in the following table:

Field	Description
Project Location	Click **Browse** to navigate to the project folder on your file system and click **OK**. This project is created using the Composer application.
Project	Select the application or project to which you intend to migrate the Composer Types.

3. Click **Next**.
4. Review the Types and select the check box next to each Type that you want to migrate.
5. Click **Finish**.

Notes

- The Composer Type Migration Rule allows you to choose the data type for the migration of the selected composer type.
- Composer does not support importing Advanced Workflow artifacts because the artifacts listed in Sysobjects do not have proper object names. In addition, when the artifacts are installed on another docbase, the references are lost.
- After the required prerequisite for migration is all set and validated, if you attempt to make any changes to selected artifact (for example, rename or move artifact) the outcome does not reflect the desired outcome. You may encounter errors in the **Problems** tab and you must manually fix such errors.

8.1.10 Using migration navigator

This procedure is applicable to these components—process, types, permission sets, and alias sets.

1. On the toolbar, click **Migration Navigator**.
2. Select any of the component and right-click on it.
3. Select the **Migrate from Composer Project** option and specify the path of the project. For example, path of the project created using Composer.

   Note: You can migrate the process using the **Migrate from Repository** option as well. Before you perform, you must set the design-time environment.

4. In the **Migrate from Composer Project**, select the application or project to which you intend to migrate the selected component.
5. Click **Next**.
6. Select the listed component artifact for migration and click **Finish**.
7. Right-click or double-click the component and select **Fetch migration steps**.
8. Depending on the component you select, perform the following:

   a. *Alias Sets:* On the **Alias Values** tab, select the alias value as an application parameter or artifact as applicable.

      Notes

      - For the **Cabinet path** alias type defined in Composer, you would get an equivalent **Cabinet or Folder path** alias type in Advanced Workflow.
      - As a best practice, if you intend to retain the same values of alias defined in Composer, migrate the artifact used in alias first followed with the Alias Set migration.

   b. *Composer Types*: To migrate types created using the Composer application to the Advanced Workflow, refer the “Migrating types from Documentum Composer” on page 215.

   c. *Permission Sets*: On the **Permission Set Migration Rules** tab, map the accessors to the application parameters to resolve the dependency.

      In the Process Builder applications, the values to accessors are provided by the install parameters. The equivalent of install parameter is application parameter in the current version of Advanced Workflow. You must map the install parameters with the application parameters, so that these values are populated in accessors during migration.

   d. *Process*: Depending on the structure of the process and its components, various tabs may appear—Process properties, Process packages, and Process parameters, and Activities.

      Only relevant tabs appear depending on the steps performed during the pre-migration phase. If any of the process component does not appear, it indicates that the migration assistant has automatically migrated the component successfully.

   e. *Object Models*: To adopt the Business Object, Cabinet, Content, and Folder object model from the repository, see Adopt Data Types.

9. Resolve any dependency using the pre-migration steps.

   Note: After you have resolved all these dependencies, you can proceed to the migration.

10. Right-click on the component and click **Start Migration**. The dialog box indicates the successful completion of migration. If any errors occur, the system prompts you with a log of errors. The migration log provides a record of activities performed to migrate artifacts in Advanced Workflow. The logs can be found at the same location as the Advanced Workflow log file. You must resolve the errors to complete the migration.

    During migration, the artifacts undergo the change in state as follows:

    ```
    //Change in state
    Pending -> Ready -> Done

    Pending -> Ready (Occurs when you validate the inputs using Validate Inputs.)
    Ready -> Done (Happens when you perform the Start Migration activity.)
    ```

    Currently, the migration do not persist its state. If you close the migration artifact or close the advwfProcess Process, you may loose the progress and you must restart the migration process.

    The system validates the entire migration and indicates the successful completion of the migration.

8.1.11 Exporting preferences

1. On the toolbar, click drop-down arrow and select **Export Preferences** to export preference settings to your local system.

2. Depending on your requirement, you can select one of the options:

   a. *Export all*: Select all preference settings of the pre-defined environments—Deployment environment, Design-time environment, and Log preference.

   b. Select one or more preference settings from the listed environments.

3. In **Description** text field, you can view the description of the selected environment.

4. After you have selected the environments you intend to export, browse and specify the name of the file to which you want to export the details of preference settings.

Note: The preference settings are saved as comma-separated values in Eclipse Preference Format (epf).

5. Select **Overwrite existing files without warning** if you want to replace the changes you made to the same output file.

8.1.12 Importing preferences

1. On the toolbar, click drop-down arrow and select **Import Preferences** to import preference settings from your local system.

   Note: You cannot import preference settings of run configurations.

2. Specify the name of the file from which you want to import the details of preference settings from your local system.

3. Depending on your requirement, you can select one of the options:

   - *Import all*: Select all of the pre-defined environments—Deployment environment, Design-time environment, and Log preference.
   - Select one or many specified environments from the listed environments.

   Notes

   - When you select the **Import all** option, the existing default environment is retained for deployment and design-time environments. If no environment is marked as default, the system picks up the first entry.
   - When you select the **Import all** option, the system retains the latest preference settings of the Log viewer.
   - If you import the same environment again, the system appends the name of the environment with a suffix, for example, Development Environment, Development Environment (1), Development Environment (2), and so on.
   - When you import preferences settings for the first time, you add up new entries to the existing preference settings in your local system. However, the system throws the following exception:

     ```
     org.osgi.service.prefs.BackingStoreException: Exception
        occurred while saving project preferences
     ```

     This occurs only when you import preferences for the very first time. Restarting the Advanced Workflow enables you to fix the issue.

4. In **Description** text field, you can view the description of the selected environment.

8.2 Concepts

8.2.1 Configurations for running applications

A run configuration stores deployment settings for running an application from Advanced Workflow so you do not have to specify the settings every time you run the application. Run configurations enable a team of application developers to run the same application in different environments. For example, you create a configuration to run an application in development environment A. Another application developer uses a source control system to get the latest version of the application and creates a configuration to run the application in development environment B.

You can test different endpoint and parameter values with your application by assigning different endpoint and parameter values to the run configuration. Changing the values in the run configuration does not change the values in the endpoint and parameter models.

When you run the application in an xDA environment set to the development mode, the system overwrites previously deployed parameter and endpoint values with the parameter and endpoint values in the configuration file. If the environment mode is set to production, the system does not overwrite previously deployed parameter and endpoint values.

8.2.2 Packaging an application

An application must be packaged before it can be deployed into a development, staging, or production environment. Packaging is the automated process of validating an application, compiling it, and putting the application in the correct format for a deployment environment.

There are three methods for packaging an application:

- Running an application from the toolbar packages and deploys an application, typically in a development environment.
- Packaging an application from the toolbar. An application or system administrator then uses the xDA to deploy the application package to a staging or production environment.
- Packaging an application using command-line interface.

8.2.2.1 Package Using Command-line Interface

Using the following command-line interface method, you can package applications and projects created using Advanced Workflow.

Package using Apache Maven

All Advanced Workflow projects are standard Maven projects. Each project contains a project object model (POM) file in the XML format located at the root folder of the application.

This is supported on Windows and Linux platforms.

To package on Linux:

- You must first install the 64–bit version of JDK 7.0.x or later in the system as Java is not packaged in the Linux build.
- You must install the Apache Maven 3.0.5.
- The Advanced Workflow application version should be same for Linux and Windows when exporting an application.

For the list of supported versions of Windows and Linux platforms, see *OpenText Documentum Advanced Workflow Release Notes*.

8.2.3 Application deployment log file

When you click **Run Application** on the toolbar, the system generates the `runapp.log` file. All application deployment messages, including validation, status, warning, and error messages are logged to this file. Each time you run the application, the system overwrites the log file.

View the application deployment log file in any of the following ways:

- During application deployment, view the running log messages in the **Console** view.
- If deployment fails, click **View Log** in the **Preparing to Run Advanced Workflow application** dialog box.
- After deployment succeeds or fails, navigate to `advwfProcessDesigner\Applications\<application workspace>\<application project>\` and open the `runapp.log` file in a text editor.

8.2.4 Build Automatically

The *Build Automatically* feature enables the system to build the application automatically whenever you open your application or save any artifacts. The process of building the application is incremental in nature; that means only application components changed or impacted by the change since the last build are taken in consideration. Note that when you create a project for the first time using Advanced Workflow, automated building of your application is disabled by default, thereby reducing the time it takes to open your Advanced Workflow application.

Disable Build Automatically

The process of building all the artifacts can sometimes be time-consuming due to the size and complexity of those artifacts. To perform all the activities related to creating an application without disrupting the flow of your work, you can disable the *Build Automatically* feature and enable it when you are ready. For instance, if you are required to create 200 business objects for an application, you can disable the *Build Automatically* feature and the create all the business objects. After you have successfully created these 200 business objects, you can enable the automated build. The incremental build for these newly created 200 business objects will be triggered once.

You cannot disable the ongoing build until the build is completed.

Note that the Preview mode does not work when you disable the automated building of your application. To preview the changes, you must enable the automated build setting.

The *Build Automatically* setting is preserved for the next instance of the Advanced Workflow. For example, if you disable Build Automatically and then close Advanced Workflow, the feature will be disabled when you open Advanced Workflow the next time.

8.2.5 Validate Application

When you enable the *Build Automatically* feature, the build is triggered and artifacts are validated automatically so there is no explicit need for you to validate the artifacts. When the feature is disabled, the *Validate* application functionality lets you validate the artifacts without building the application. Note that when you disable the *Build Automatically* feature, the system does not validate the application for errors. The subsequent incremental build performs validation for the application components that changed since the last build. Usually, if the application has any errors, the errors are shown in the *Problems* view of the page.

8.2.6 Type Adoption

Type adoption feature enables you to adopt data types from the repository to the Advanced Workflow application. Using type adoption, Advanced Workflow application can read or write data instances created through OpenText Documentum CM clients, such as previous versions of Documentum Administrator, Webtop, and TaskSpace, and adopted in the Advanced Workflow. Similarly, clients can read or write new instances of adopted types created in an Advanced Workflow application. Thus, the type adoption feature provides interoperability between clients and Advanced Workflow applications accessing data instances.

In this section, Advanced Workflow application refers to the application created using the current version of Advanced Workflow.

You can adopt only subtypes of dm_sysobject, dm_document, dm_cabinet, and dm_folder data types. When you adopt a data type that is inherited from another data type, all the super types are also adopted. For example, if the premium_customer type is inherited from privileged_customer type, which in turn is inherited from the customer type, when you adopt the premium_customer type, its super data types, privileged_customer, and customer types are also adopted. The adopted data types then appear in the object models navigator.

If you provide a label for a data type during type creation using clients, the system uses the label values. For example, consider a data type loan_customer is created with a label of Loan Customer. When you adopt this data type in Advanced Workflow, the object model appears with *Loan Customer* as its label and loan_customer as the system name. However, if you do not provide a label for a data type during type creation, the system uses the default label values set by the Documentum CM Server. For subtypes of dm_sysobject, the label is *SysObject*. For subtypes of dm_content, the label is *Document*. For subtypes of dm_folder, the label is *Folder*. And for subtypes of dm_cabinet, the label is *Cabinet*.

Subtype	Label
dm_sysobject	SysObject
dm_content	Document
dm_folder	Folder
dm_cabinet	Cabinet

When data types are adopted in an Advanced Workflow application, they behave similar to any other data types created in Advanced Workflow. However, Advanced Workflow does not provide support to:

- Modify the attributes of adopted data types
- Add relationships with other types
- Attach type fragment to those data types

As a best practice, use the MAINTAIN data policy while deploying adopted data type on the production environment, to avoid data loss.

8.2.7 Migration navigator

On the toolbar, use **Migration Navigator** to migrate legacy Process Builder processes and other artifacts. You can migrate artifacts and its components from both Composer and Repository projects.

The following table gives you a snapshot of what activities are supported for these supported artifacts:

Supported Artifacts	Migrate from Composer Project	Migrate from Repository	Adopt from Repository
Alias Sets	Yes	Yes	No
Composer Types	Yes	No	No
Permission Sets	Yes	Yes	No
Process	Yes	Yes	No
Structure Data Type	No	No	No
Types	No	No	Yes

Perform the steps mentioned in the following sections to know how to migrate or adopt these components.

8.2.8 Process migration

Use Migration navigator to migrate the legacy Process Builder business processes developed using Process Builder to the current Advanced Workflow business processes. The main objective of the process migration is to migrate the most important and complex parts of process design automatically—expressions, data mapping, transition paths, process variables, packages, endpoints, and performers. The entire migration happens in a seamless manner so that process data mapping, expression, and process flow remains intact. Process migration is supported from both OpenText Documentum CM Repository and Composer projects.

Note: Only processes created using Documentum Process builder 6.6 and later are supported from repository and composer projects.

The entire process migration occurs in four stages:

1. *Process Discovery*: Based on your requirement, you ascertain the legacy Process Builder processes you intend to migrate. Verify the involved dependencies such as types used in packages, type of process parameters, structure data types, performers, business calendars, work queues, Java modules, endpoints, and sub processes.
2. *Pre-migration steps*: The system actually checks the process and identifies all the dependencies. Follow the steps as mentioned in the *Prerequisites* section.

3. *Migration steps*: Migrate the process with associated dependencies. Also, the components that do not appear in the pre-migration steps are migrated automatically.

   Note: The process migration does not migrate the process completely. Any feature of the Process Builder process that is not supported in Advanced Workflow, is replaced either with an equivalent Advanced Workflow feature or a default value is assigned. Follow the post-migration steps to populate these values.

4. *Post-migration steps*: Few components configuration may not be migrated. You have to manually migrate these components to complete the process migration entirely.

Prerequisite Requirements (Pre-migration steps)

- *Object Models*: Adopt object models for packages using the type adoption and type migration feature.
- *Application parameters*: Create equivalent application parameters for process parameter , work queue, business calendar, life cycle, user, and group.
- *Endpoints*: Endpoint refers to external system such as Web Service, HTTP, Database, JMS, Email, FTP, xPression, and OpenText Documentum CM repository. In Process Builder, the endpoints are saved as part of the activity definition. In Advanced Workflow, the endpoints are referred by Endpoint artifacts. Before migration, create the endpoints in Advanced Workflow and refer them at the pre-migration step.
- *Permission Set*: Migrate or adopt permission set for **Create ACL** activity.
- *Alias Set*: Migrate or create alias set to be used in Manual activity performer configuration.
- *Java module*: Create Java module for **BOF Module** activity template and task priority for **Work queue** activity.
- *Process*: Any sub processes used in the main process through **Invoke Process** or **Synchronous Invoke Process** activity template must be migrated separately similar to any other process and before you migrate the main process. If you are migrating a Synchronous process (referred as Stateless process in Advanced Workflow), the migrated process is not marked as **Stateless** by default. It should be marked as **Stateless** post the migration. The data mappings involved in **Invoke Process** or **Synchronous Invoke Process** activity template cannot be honored if the child process is refactored due to SDT, incompatible variable name, or incompatible package name.

The following table provides a quick snapshot of the components that are supported as part of Process migration:

Table 8-1: Process Properties Migration

Tabs	Component	Auto migration	Pre-migration step	Post-migration step	Additional Information
General	Template name	Yes	(Optional) Specify a new process label to the template.	Not applicable	Not applicable
Original Creator	No	Not applicable	Not applicable	This feature is not supported.	
Current State	No	Not applicable	Not applicable	This feature is not supported.	
Version Label	No	Not applicable	Not applicable	This feature is not supported.	
Process Template owner	No	Not applicable	Not applicable	This feature is not supported.	
Description	Yes				
Default alias set	Yes	Not applicable	Not applicable	No applicable.	
Workflow Instructions	Yes	Not applicable	Not applicable	Not applicable	
Audit Trail settings	No	Not applicable	Not applicable	This feature is not supported.	
Store document name to the package at runtime	Yes	Not applicable	Not applicable	Not applicable	
Select Template for email notifications	No	Not applicable	Define a new email template.	Not applicable	

Tabs	Component	Auto migration	Pre-migration step	Post-migration step	Additional Information
Data	Packages	No	If the name of the package contains any special or multi-byte characters that are not allowed in Advanced Workflow, you must provide a new name for the same.	Not applicable	• Migration of associated xForm is not supported. • “This package can be used to generated report” field is not supported.
Process variables—Simple Data Type	Yes	If the name of the variable contains any special or multi-byte characters that are not allowed in Advanced Workflow, you must provide a new name for the same.	Not applicable	Process variables are migrated automatically but their ACLs are not migrated.	
Process variables—Structure Data type	No	See SDT migration section.	Not applicable	Not applicable	

Tabs	Component	Auto migration	Pre-migration step	Post-migration step	Additional Information
Process parameters	No	• You must create an equivalent application parameter of the same type. • If process parameters are used in performer configuration, for example, Performer based on process data and performer selection based on process data, then create application parameters of user, group and work queue type.	Not applicable	Not applicable	
Advanced	Variable ACL	No	Not applicable	Assign a new permission set.	Not applicable
Process ACL	No	Not applicable	Assign a new permission set.	Not applicable	
Calendar	No	Create or assign Business Calendar application parameter.	Not applicable	Not applicable	

Tabs	Component	Auto migration	Pre-migration step	Post-migration step	Additional Information
Process Parameter Form	No	Not applicable	Not applicable	This feature is not supported.	
Hi-Fidelity Form (ODT Form)	No	Not applicable	Not applicable	This feature is not supported.	
Correlation Set	Yes	Not applicable	Edit the correlation set to add new attributes that maps to simple variable types if the Process Builder process has SDT mapped in the correlation set. The data mapping pertaining to the correlation set in **Inbound step (Wait for)** activity must be changed to reflect the new attributes added to the correlation set.	The attributes in the correlation set mapped to simple variable types are auto migrated. Those that involves SDT attributes are ignored during migration.	

The following table provides a quick snapshot of the components that are supported as part of activities migration. This is applicable to both automatic and manual activities.

Table 8-2: Activities Migration

Component	Auto migration	Pre-migration step	Post-migration step	Additional information
Properties	No	Not applicable	Not applicable	The system does not allow you to migrate the Properties tab of auto activities as it is not supported in Advanced Workflow. Only task priority is auto migrated in the Execution tab of the activity.
Performer (See "Performer Type Migration" on page 233 section.)	No	Not applicable	Not applicable	Not applicable
Trigger	Yes	Not applicable	Not applicable	Not applicable
Transition	Yes	Not applicable	Expressions containing attributes from **The running workflow** and **Last completed work item for the activity** are not supported in Advanced Workflow. You have to rewrite the expression again post the migration activity.	Not applicable

Component	Auto migration	Pre-migration step	Post-migration step	Additional information
Timers	Yes	Create a Business Calendar parameter and assign it in the pre-migration step provided the timer configuration contains Business Calendar. Create a User Application parameter and assign it in the pre-migrtaion step if the timer action is to **Delegate Task** to a user.	Timer actions **Notification** and **Run Java Method** are not supported in Advanced Workflow. You can use Stateless or Stateful process with **Execute Java Service** activity to run custom Java methods as part of timer actions.	**Use email template** field is not supported in Advanced Workflow.
Notification	No	Not applicable	Email templates used in notification are not auto migrated. You must create a new email template post the migration activity.	Not applicable
Data	Yes	Not applicable	Not applicable	• Migration of associated xForm is not supported. • **This package can be used to generated report** field is not supported.

Component	Auto migration	Pre-migration step	Post-migration step	Additional information
Display	Yes	Not applicable	Custom images are not auto migrated. You must assign custom images again post the migration activity.	Not applicable

Generic Notes:

- *Default Initiate Activity*: The activity is not migrated automatically. For a process that does not have a link from default initiate activity, the post-migration step is to add a link to the first activity and configure the trigger condition.
- *Web Service inbound activity*: The activity is not migrated automatically. Web Service inbound activity configured using WSDL from the local file system is migrated with a blank WSDL URL. You must configure the WSDL URL.
- *Endpoint*: The email template configured in SMTP activity template is auto migrated. All input and output message mapping is auto migrated.
- *XML schema associated to activity content*: There can be XML schema with attachment or package content or with the content received from external sources (such as HTTP or JMS services) and use the mapper to map elements from the schema. The schema are automatically migrated and added as a resource in Advanced Workflow application during migration. The data mapping to and from schema elements are preserved.
- *XML based transition condition*: If transition condition is based on a package that has an XML document, the expressions are migrated seamlessly using XML functions.
- The *JumpToArtifact* link does not work for Process —> Package Type. You should go to navigator for editing and viewing artifacts.

The following table lists the Process Builder auto activities that are not supported in Advanced Workflow. When a Process Builder process contains any of these activities, such activities are replaced with Execute Java Service activity as placeholder or an equivalent Advanced Workflow auto activity templates. For any placeholder Execute Java Service activity, user must create a Java module separately using Java module feature and associate it to the process migration at post-migration step. The placeholder activity does not honor any configuration or data mapping inherited from the Process Builder activity.

Table 8-3: Non-supported and Supported Auto Activities

Process Builder Automatic activity template	Current Automatic activity template
FS2 Search	Execute Java Service (as a placeholder activity)
Create iCalendar Event	Execute Java Service (as a placeholder activity)
XSL Transformation	Execute Java Service (as a placeholder activity)
Fax Outbound	Execute Java Service (as a placeholder activity)
Queue Task Rework Decision	Execute Java Service (as a placeholder activity)
Set Queue Task Skill	Execute Java Service (as a placeholder activity)
Link To Folder	Move (as a placeholder activity)
Create ACL	Create or Apply ACL
New Case from Template	Create (as a placeholder activity)
BOF Module	Execute Java Service
Invoke Process	Call process
Synchronous Invoke Process	Call stateless process
Create Folder	Create (as a placeholder activity)
Post Event to Parent Process	Signal Parent Process (as a placeholder activity)
dm_noop_auto_method/ dm_bpm_noop_method/ dm_noop_auto_method_java	Join
Custom methods that can be chosen in performer configuration tab	Java method activity
Timer Delegation to Group	Execute Java Service (as a placeholder activity)

The following table provides a quick snapshot of the components that are supported as part of performer type migration:

Table 8-4: Performer Type Migration

Performer Type	Auto migration	Pre-migration step	Post-migration step	Additional Information
Workflow supervisor	Yes	Not applicable	Not applicable	Not applicable
Repository Owner	Yes	Not applicable	Not applicable	Not applicable
Performer from Previous Activity	Yes	Not applicable	Not applicable	Not applicable
• Specific user • All Users in a group • Single user from a group • Multiple users from a group • Multiple sequential performers	No	Create an appropriate User or Group parameter or alias set manually and assign it in pre-migration step for these configurations: 1. Assign performer. 2. Define performer. alias 3. Select performer based on condition. You should select same number of performers in pre-migration step as it is set in the Process Builder process.	Following performer configuration is not supported and is migrated to performer type *Workflow supervisor*: • Have performer of activity <activity_name> to determine the performer of the activity. • Select performer based on conditions—Type a DQL query. Performer aliases other than **Specific alias set** is migrated to performer type Workflow supervisor.	Following performer configurations are auto migrated: • Select performer based on process data. • **Select performer based on condition** with user selection based on process data. In Advanced Workflow, the work queue task priority can be dynamic only if task priority module is configured. During migration, you must set the priority to medium (default priority level) for work queue task if there are no task priority module configured in the Process Builder process.

Performer Type	Auto migration	Pre-migration step	Post-migration step	Additional Information
Work Queue	No	Create a work queue parameter and assign it in pre-migration step.	Not applicable	Not applicable
Select performer based on process data	Yes	Not applicable	Not applicable	Not applicable

8.2.9 Core extension library

The library contains pre-adopted system types. It helps end users to use system types in Advanced Workflow application for process migration. For example, when you migrate a Process Builder process in Advanced Workflow and consider one of the package types is XML instance, dm_xfm_instance. To migrate the process with XML instance type, you must import the library, which in turn shows XML instance type in Advanced Workflow application and uses it in process migration.

Note: Only **dm_xfm_instance** is supported in core extensions library for this release.

Following are the dm_xfm_instance XML instance limitations when used in Advanced Workflow application:

- The XML instance does not adopt its associated eForms.
- Storage mappings are not honored for existing or new XML instances when updated from Advanced Workflow application.
- The new XML instance created from Advanced Workflow application is not interoperable with the Process Builder clients, such as Webtop, Taskspace.
- When XML instance is used in process migration, the associated schema is not auto migrated and process data mapping rules with schema attributes is lost.

  Note: Once the post process migration is complete, you can fetch the schema from repository, then add it to the application as a resource and use it in process, similar to any other schema, to define the rules again for all required activities.

8.2.10 Structured Data Type migration

The process variables of type Structured Data Types (SDTs) are not supported in Advanced Workflow. Instead, use Business Object (BO) to support the functionality of SDT. The SDTs have to be manually converted to Business Objects at the pre-migration stage. To convert SDT to BO, you must map the SDT attributes with Business Object attributes using the pre-migration step of process migration. The mapping is used to migrate SDT attributes to BO attributes in data mapper or expression used in transition conditions and conditional performer configuration. The logical grouping of data in SDTs can either be mapped to a single BO or converted to multiple BOs and linked together using the **Relations** functionality.

Note: As a best practice, it is recommended to denormalize or flatten the SDT to Business Object with maximum of three types to avoid any performance issues.

The process variables of type SDT is added as package to the process during migration. You should provide a name to the package added to process at pre-migration stage. As part of SDT migration, any package that is added to a process is marked as mandatory by default.

For example, the *Employee* SDT has department and address as groups and branch as subgroup of department. You can choose either one BO (Employee) or two BOs (Employee and Department) to denormalize the SDT.

Structured data types	Business object

`- employee_name(attribute)`
`    - employee_age (attribute)`
`    1 group-department`
`        - department_name (attribute)`
`        1.1group- branch`
`            - branch_name (attribute)`
`    2 group-address`
`        - address (attribute)`
	`- employee_name(attribute)`
	`- employee_age (attribute)`
	`- department_name (attribute)`
	`- branch_name (attribute)`
	`- address (attribute)`
	Or,
	`BO-employee`
	`- employee_name(attribute)`
	`- employee_age (attribute)`
	`- address (attribute)`
	`BO-department`
	`- department_name (attribute)`
	`- branch_name (attribute)`

Chapter 9

Troubleshooting application deployment issues

9.1 Outbound activities fail when configured to create secure connections for message delivery

Problem

If an outbound activity (Email, FTP, JMS, HTTP, or Web Service) is configured to create secure (SSL or TLS) connections to deliver messages, it fails with the following exception at runtime when creating a connection to the server:

java.security.InvalidKeyException: Illegal key size

Cause

Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files are required for the secure connections.

Resolution

When outbound activities (Email, FTP, JMS, HTTP, and Web Service) are configured to deliver messages over secure (SSL or TLS) connections, apply JCE Unlimited Strength Jurisdiction Policy Files if the server uses the following cryptographic algorithms for encryption with sizes:

- DES more than 64-bit
- DESede all sizes
- RC2 more than 128-bit
- RC4 more than 128-bit
- RC5 more than 128-bit
- RSA all sizes
- All other algorithms more than 128-bit

To apply JCE Unlimited Strength Jurisdiction Policy Files:

1. Download Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files from the Oracle website.
2. Locate the JAVA used by the JBoss Method Server. You can locate it as INSTANCE_JAVA_HOME in startMethodServer.cmd in `<Documentum>\jboss7.1.1\server`.

3. Back up the `local_policy.jar` and `US_export_policy.jar` from `INSTANCE_JAVA_HOME/jre/lib/security` folder.
4. Stop all the Java processes using INSTANCE_JAVA_HOME Java (typically Documentum Server Java processes).
5. Copy the downloaded policy files to the `INSTANCE_JAVA_HOME/jre/lib/security` folder.
6. Start the Java processes that you shut down in step 4.

9.2 Deadlock in Email inbound activity

Problem

Emails are not processed by email inbound activity and deadlock situation is noticed in the database.

Resolution

Perform the following actions:

1. Enable <Snapshot Isolation> and set READ_COMMITTED_SNAPSHOT to ON at the database level.
2. Enable <Snapshot Isolation> at database level and set READ_COMMITTED_SNAPSHOT to ON in the code. That means the code must enable snapshot isolation before calling any SQL query.

9.3 Clean temporary files available in Temp directory

Problem

At rare occasions, the temporary files get created on your machine that fills up the hard disk and clogs the application related activities. For example, the possible scenarios:

- When multiple application server instances are running and all pointing to the default system temporary folder
- When multiple applications are redeployed continuously, temp files get accumulated

Cause

The Advanced Workflow application creates a set of temporary files (with a prefix advwfProcessdar) during its execution. These DAR files contains Artifacts, Metadata and Resources.

Resolution

Typically, the temporary files are generated at these scenarios:

- xDA—At the time of Advanced Workflow application deployment
- Documentum CM Server—Java method server running on JBoss
- DFC scripts—when DFC scripts are executed and it further modifies a Advanced Workflow artifact

Set a separate temporary folder using**–Djava.io.tmpdir** JVM variable and then clean the temporary files for Documentum CM Server and DFC applications. Even the temporary files are deleted when the application server shuts down gracefully.