opentext™

OpenText™ Documentum™ Content Management

Workflow Designer User Guide

Create and manage business processes. A process formalizes a business workflow, enabling users to repeatedly perform a process.

EDCPKL250400-AWF-EN-01

OpenText™ Documentum™ Content Management
Workflow Designer User Guide
EDCPKL250400-AWF-EN-01
Rev.: 2025-Dec-23

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

Table of Contents

Chapter 1

Introduction

OpenText Documentum Content Management (CM) Workflow Designer is used to create processes for Documentum clients. A process template defines a series of business procedures that can be used repeatedly. You can create and run multiple instances of a process template at run time. A process consists of one or more *activities* linked together by *flows*. An activity represents a task that a user or a system perform on the process components (activities and flows) to pass the document to the next activity in the process. For example, reviewing a document, checking the document into the repository, or approving the document. A flow connects two activities and defines the sequence of activities in a process. It also defines the process data that is exchanged between the activities. Process data is a collection of packages (Sysobject, Content, Folder, or Cabinet) and process variables that are processed during an activity and then passed to the next activity in the process.

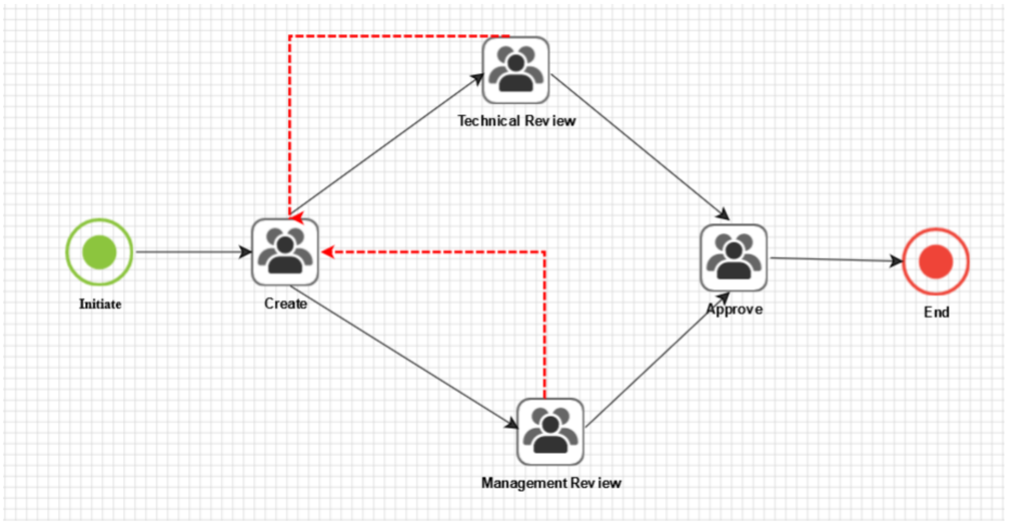
This guide describes how to use Workflow Designer to design and configure business processes..

Chapter 2

Understanding processes

Workflow Designer is used to create process templates for Documentum clients. A process defines a workflow that can be used repeatedly. You can create and run multiple instances of a process template at run time.

A process template consists of one or more *activities* linked together by *flows*. An activity represents a task that a user must perform on the business object. For example, reviewing a document, checking the document into the repository, or approving the document. A flow connects two activities and defines the sequence of activities in a process. The process data is a collection of packages and process variables that are processed during an activity and then passed to the next activity in the business process. "Process template" on page 12 provides further details about these process components.



You can define a process template that have following information flow characteristics:

- Serial: Has activities that follow a specified sequence with only one activity running at an instance.
- Parallel: Has activities that can run simultaneously at an instance.

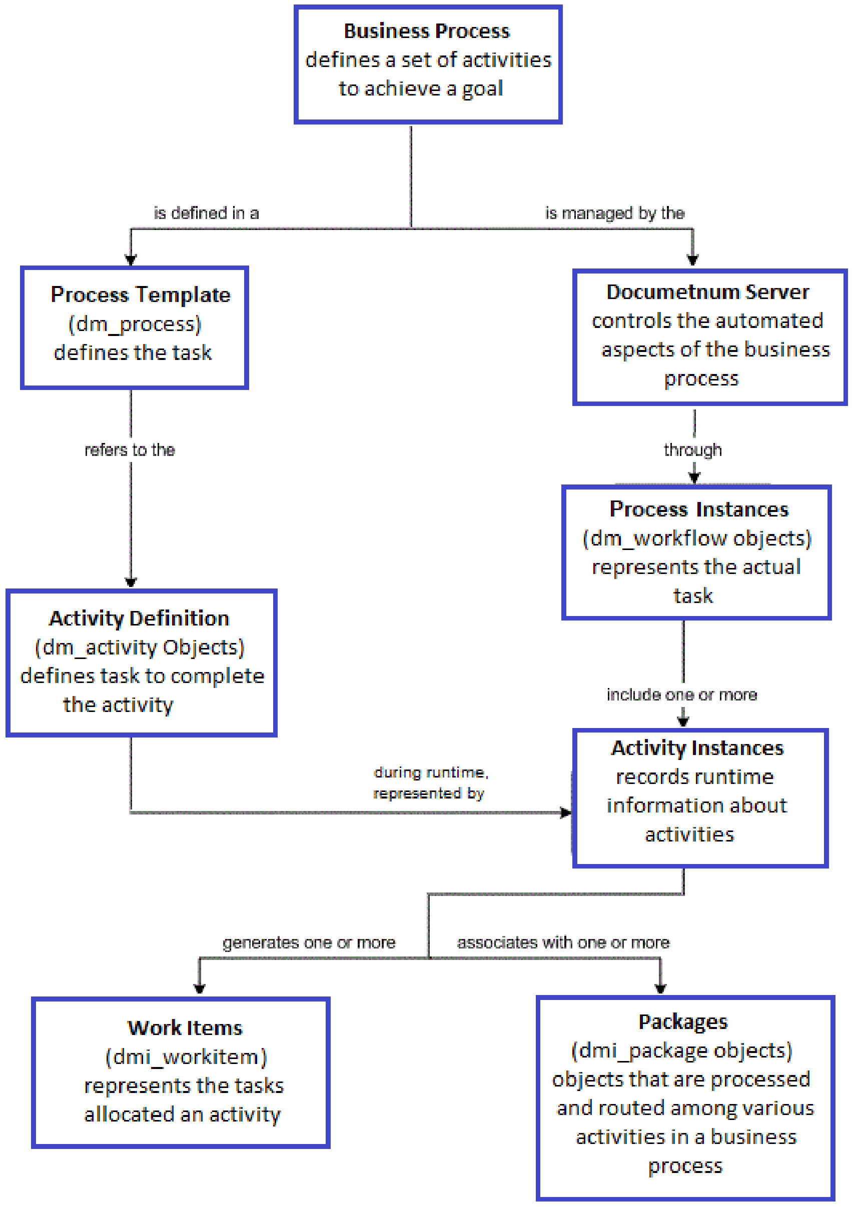
The path that a document follows in a process depends on the result of activities in a process; for example, a purchase order could be routed to different activities depending on whether the manager approves it or rejects it.

You can create generic process template that can be used in different scenarios. You can create a process template where an activity performer is mapped to an actual user name at the run time. For example, a typical workflow for creating a new document has five steps: authoring the document, reviewing it, incorporating the reviews, getting approval, and publishing the document. The authors and reviewers are different for different documents. Hence, instead of creating a separate process for each document with the author and reviewer names, you can create a process template with activity definitions that maps to the author and reviewer at run time.

A process definition is stored in OpenText™ Documentum™ Content Management Server as dm_process object. The definitions of individual activities in a process are stored as dm_activity objects.

When you start a process, the server uses the process template (the dm_process object) to create a runtime instance of the process template(a dm_workflow object). When an activity starts, the server creates one or more *work items* (tasks) and communicate to the designated activity performer.

The following flowchart illustrates how the components of a process template and runtime instance work together. For more details about the object-level implementation of a process, see *OpenText Documentum Content Management - Server Fundamentals Guide (EDCCS250200-GGD)*.


Business Process defines a set of activities to achieve a goal
is defined in a
is managed by the
Process Template (dm_process) defines the task
Documetnum Server controls the automated aspects of the business process
refers to the
through
Process Instances (dm_workflow objects) represents the actual task
Activity Definition (dm_activity Objects) defines task to complete the activity
include one or more
during runtime, represented by
Activity Instances records runtime information about activities
generates one or more
associates with one or more
Work Items (dmi_workitem) represents the tasks allocated an activity
Packages (dmi_package objects) objects that are processed and routed among various activities in a business process

2.1 Process template

A process template defines the structure of a business model. A process definition consist of a set of activity definitions and a set of flows connecting the activities.

2.1.1 Process

A process is an instance of a process template at run time. Multiple processes based on the same process template can run concurrently because a process template is separate from its runtime instance. You can use a predefined process template or create your own.

A process template consists of the following elements:

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

2.1.2 Activities

Activities represent the tasks that make a business process. A process has three types of activities:

- *Initiate activity* is the first activity in the process. A process template must have only one initiate activity.
- *Automatic, Manual,* and *AI Workflow Automation* activities are the intermediate activities between the initiate and the end activities. A process template can have multiple Automatic, Manual, or AI Workflow Automation activities.
- *End activity* is the last activity in the process. A process template can have only one End activity.

A manual activity is performed by a designated performer or group of performers. An automatic activity is run by the system.

You can define the following attributes for an activity:

- Activity name and basic activity information
- Priority
- Activity performers
- Trigger condition for an activity

- Notifications generated during an activity
- Transition information when an activity is completed
- Process data attached to an activity
- Display attributes

When an activity is initiated in the server, work items are created and a communication is initiated to the activity performers. The work items contain the process data that the user needs to work on and instructions to complete the task.

2.1.3 Flows

A flow connects two activities, defines the workflow, and enables the movement of process data (packages and process variables), their properties, and dependencies between the connected activities. A single flow can handle multiple packages and process variables.

There are three types of flows: `forward flows`, `reject flows`, and `fault flows`. A forward flow passes the packages and process variables from the current activity to the next activity in the process. For example, moving a package from the Edit activity to the Approve activity.

Reject flow determine what happens when the performer of an activity rejects the task being routed. Reject flows direct process data to a backward loop or to an alternate forward flow. For example, a package moves from the Approve activity back to the Edit activity. A red arrow with dashed line represents a Reject flow.

Fault flow defines the action to take when an associated system activity fails. It directs the flow to a manual or automatic activity to resolve the error and direct the flow to the next activity in the process. You can connect multiple automatic activities to a fault handling activity.

For example, if an AI Workflow Automation activity fails to extract all the required fields, you can configure the fault flow to direct the workflow to a manual activity where user enters all the missing required fields or to an automatic activity before proceeding to the next activity in the process. A yellow arrow with dashed line represents a Fault Handler flow.

Note: Fault flow is available only with the Advanced Workflow Designer license.

A fault handling activity resolves errors when a system activity fails. It is triggered by a fault flow and cannot have outgoing transitions, serving only as an error handler within the process. When assigned to an automatic activity, the fault handler runs each time the associated method fails. You can configure it to retry the activity a specified number of times. After exhausting the retries, the system takes a final action—continue, stop, or terminate the workflow based on the settings in the **Performer** tab.

An automatic or manual activity has at least one incoming flow and one out flow. An Initiate activity has only one or more outgoing flow(s) and an End activity has only incoming flow(s).

You can label each flow in a process template. You can set the following attributes for a flow:

- Label Styling
- Font Size
- Label settings such as customizing the label

2.1.4 Process data

Process data is the data that flows through the process at run time. Process data includes packages and process variables. Using the process properties, you can specify different types of data for a process template and 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. Process data consists of following types of data:

- **Packages:** Packages reference business object, content, and folder data. The package includes the objects and its associated content and attributes. Packages are objects in the repository that persist after a workflow completes. Activities work on these packages and then pass them to other activities. For example, a workflow can contain a loan application within a package.
- **Process variables:** Process variables store transient data that the system no longer needs when the process is completed. For example, you can create process variables for part numbers and customer addresses.
- **Execution data:** Execution data comes from the current process and work items, such as process creation date and workitem run time 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 during run time. For example, in an engineering proposals under development, an end user can attach a research paper to support the 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.

Packages and process variables are configurable. You define process data at the process level and use them in individual activities within a process.

2.1.5 Packages

Packages are placeholders for the actual business object, content, and folder data that run through a process at run time. 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 activity in the process can use that package to display customer information for the approver.

You can configure packages using the process properties 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

2.1.6 Process variables

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

Starting with 24.2 release, you can use process variables in a transition condition for manual and auto activities to determine the next activity in a process.

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

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

Note: Starting with 24.2 release, process variables are available under the **Context Data** tab for defining a transition condition.

2.2 Defining process activities

Activities in a process template define the process. You start with defining a process and its goal. The process mandates the sequence of activities required to complete a process. See “Selecting an activity” on page 16 for information about how to decide on activities.

Consider the following points when you define an activity:

- Whether you need a manual activity or an automatic activity?
- Who performs the activity? For more information, see “Selecting a performer” on page 16.
- For manual activities, whether a performer can delegate or extend the activity? For more information, see “Defining activity delegation and extension” on page 19.
- For automatic activities, what should be the priority? For more information, see “Activity priority” on page 20.
- What process data are passed across the activities in the process? For more information, see “Packages” on page 15.
- When does an activity start? For more information, see “Trigger conditions” on page 21.
- Whether an activity requires notification or not? For more information, see “Notifications” on page 22.
- What happens next in the process? For more information, see “Understanding activity transitions” on page 22.

2.2.1 Selecting an activity

Each process must have one Initiate activity and one End activity. The process can have any number of activities. The number of activities depends on the complexity of a business process. Each activity in a process template must have unique name.

2.2.2 Selecting a performer

An activity definition includes the performer information that enables process engine to determine the activity performer at run time. Workflow Designer supports a wide range of manual activity performers. For automatic activities, you must identify a user whose permissions is used to run the activity. When a manual activity starts, the server adds a work item to the Inbox of the activity performer.

“Selection categories for activity performer” on page 17 lists the performer categories. Only the first three options are available for automatic activities.

Table 2-1: Selection categories for activity performer

User category	How performers are selected
Workflow Supervisor	The system selects the user designated as the workflow supervisor when the activity starts. By default, the user who starts the workflow is the workflow supervisor.
Repository Owner	The server selects the user identified as the owner of the active OpenText Documentum CM repository.
Performer From Last Activity	The system selects the performer from the last completed activity that satisfies the trigger condition of the current activity. The **Performer from 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 identified at run time because the performer can be anyone from a group.
Previous Activity Performer To Choose At Runtime	The performer of the preceding activity selects one or more performers that must perform the next activity at run time. If this activity has multiple preceding activities, you must select an activity whose performer selects the performer for this activity.
Specific Group	Select a group from the list of groups from the connected repository. The server assigns a work item to each performer in the group at run time.
Group from Alias Set	Select a group alias from an alias set as the performer of the activity. The server assigns a work item to each performer in the member of a group alias at run time.
User from Alias Set	Select a user alias from an alias set as the performer of the activity. The server assigns a work item to a member of the user alias at run time.

Participants in a process have the option to mark themselves as unavailable in Documentum Administrator (DA) for tasks. When a process is initiated and a performer is unavailable, the process engine assigns the work item to the designated delegated performer. See “Defining activity delegation and extension” on page 19 for information about delegated users.

For information about selecting performers for an activity in Workflow Designer, see “Selecting performers” on page 68. For details about creating activities whose performers are selected at run time, see “Determining a performer” on page 18 and “Using Aliases” on page 18.

2.2.2.1 Determining a performer

When you create an activity, you must define the performer type. You can define the actual performer at the design time or you can define an alias that is mapped to an actual performer at the run time. A performer is determined at run time by:

- The process initiator, when the workflow is started
- The server, when the activity is started
- The performer of a previous activity, when the previous activity completes

If you select the **Workflow supervisor**, **Repository owner**, or **Performer from last activity** category, the actual user is defined by the category. For example, a process has only one workflow supervisor and the repository has only one repository owner.

If you select **Specific group** category, you can provide a group name when you create an activity. In case of alias, the performer is selected at the run time. Similarly, for **Group from Alias Set** or **User from Alias Set** categories, the name of a group is selected at run time.

The **Group from Alias Set** and **User from Alias Set** categories comprise of names or aliases for a list of multiple users.

2.2.2.2 Using Aliases

An alias is a pseudo name for a *category* of user or group that you use in place of an actual user or group name. At run time, the server replaces the alias with the name of the actual user or group. Using aliases in activity definitions creates a flexible process template that can be used in a variety of scenarios. For example, for a process for vacation request, each department has a different manager who must approve vacations. So, you can use the same process template for different departments. In place of specific performer names for the activities, you use an alias, such as Manager. When the process starts, the server identifies the performer for an activity.

The server resolves aliases at run time by searching one or more *alias sets* to find the alias and the corresponding value. An alias set is an object that defines a list of aliases and the corresponding actual values. You can create alias sets in Documentum Administrator, for more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250200-UGD)*.

When an alias is defined as the performer for an activity, you can specify that the server resolves the alias:

- To the group from an alias set
- To a user from an alias set

2.2.2.3 Alias Sets

In a process, multiple references are made to many performers or performer groups. Instead of referencing the actual performer or performer group, you can assign an alias name. The actual value is called the alias value. A collection of alias names is called an alias set.

2.2.2.4 Default Alias Set

The default alias set is an alias set whose values are resolved by the process initiator when starting the process. 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 Workflow Designer.

2.2.3 Defining activity delegation and extension

When you define a manual activity, you can configure whether the performer is able to *delegate* the activity to another performer or *forward* the activity by identifying an additional performer. In the case of delegation, the original performer does not complete the activity and the activity is delegated to a new performer. When an activity is forwarded, both the original performer and the designated additional performer complete the activity.

A delegation event can occur automatically or manually.

- Automatic delegation occurs when a performer is not available. The server automatically delegates the work to the configured delegated performer. If a delegated performer is not configured or is unavailable, the work item is either reassigned to the workflow supervisor or returned to the original performer depending on the activity configuration.
- Manual delegation occurs when the performer, the workflow supervisor, or a super-user elects to delegate the work item.

If forwarding is enabled, when the original performer(s) complete an activity, they can identify a second round of performer(s) for the activity. The server generates new work items for the second round of performer(s). After the second round of performer(s) completes the activity, the server evaluates the transition condition for the activity and move to the next activity. The second round of performer(s) do not have the option to forward the activity. For more information, see "Selecting performers" on page 68.

2.2.4 Activity priority

For an automatic activity, you can designate a priority value that determines the order in which the server runs the activity relative to other activities in the queue. You can also set a priority value for manual a activity, which is highlighted in the communication to the performer.

When an automatic activity is started, the activity is placed in the queue of a server facility that runs periodically. The server runs the activities in the order of priority. By default, it runs all queued automatic activities each time it is invoked. However, a system administrator can limit the number of activities run in a batch. If the server configuration setting `max_wf_jobs` is set to a low number and there are a large number of queued activities with high priority, a lower priority activity is moved down in the queue.

You can configure priority of an activity to Low (Blue), Medium (Yellow), or High (Red). For more information, see “Setting activity definitions” on page 74.

2.2.5 Packages

A package is attached to a process and is available to all the activities in the process. You attach a package to a process in the **Process Properties**. A package is an object that is processed and passed along the process.

An activity handles a package in three different ways:

- The activity can forward a package without any changes.
- The activity can forward a package with a new version of the object.
- The activity can forward a new package to the next activity.

In some business cases, the same package may pass through all the activities. For example, a process for reviewing and approving purchase orders passes the same purchase order document as a package to all the necessary activities.

The work performed by an activity on a package may result in a new version of a document. For example, an activity to review a document. The document is updated, comments are added, and checked in. In this case, the activity passes a new version of the package to the next activity. For a package, you can specify the version using an actual version number or set to CURRENT.

In some business cases, an activity may result in a completely different package. For example, suppose an activity accepts a personnel action notice. The performer (an HR employee) must file the notice, then send a different form to the accounting department.

2.2.6 Data types

Data type specifies the type of package in a process. Workflow Designer supports the following four data types:

- SysObjects (dm_sysobject)
- Content (dm_document)
- Folder (dm_folder)
- Cabinet (dm_cabinet)

You can also define custom data types derived from one of the supported data types. For more information about how to create a custom data type, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC250200-UGD)*.

2.2.7 Trigger conditions

A trigger is a signal to initiate an activity. A trigger condition defines the criteria to start an activity. At run time, an activity is initiated only after the trigger condition is met. The trigger condition can optionally include a trigger event that must occur before the activity starts.

If an activity has multiple incoming flows, you can specify the number of previous activities that must complete before this activity starts. The *trigger condition* is the minimum number of flows that must have delivered packages to the activity before the activity starts. For example, if an activity has three incoming flows, you can configure the activity to start when two of the three packages have arrived.

A *trigger event* is an event queued to the process. The event can be a system-defined event, such as `dm_checkin`, or you can specify an event name, such as promoted or released. However, because you cannot register a process to receive event notifications, the event must be explicitly queued to the process using the OpenText Documentum CM API. If you include a trigger event in the starting condition, the server must find the event you identify queued to the process before starting the activity. The same event can be used as a trigger for multiple activities, however, the event is queued once for each activity. *OpenText Documentum Content Management - Server Fundamentals Guide (EDCCS250200-GGD)* provides more details about defining and queuing events.

For information about setting an activity's trigger conditions, see “Configuring activity triggers” on page 74.

2.2.8 Notifications

When you configure an activity, you can set notification timers that send a message to the workflow supervisor if there is delay in the activities. For example, you might want the workflow supervisor to receive a notification if an activity is not started within a specific time or if an activity does not complete within a specific time period. When you create the activity, you can configure these notification timer settings.

Workflow Designer supports two types of notification for an activity:

- A pre-timer that alerts the workflow supervisor if an activity has not started within a specified time after the process starts.
- A post-timer that alerts the workflow supervisor if an activity does not complete within a specified time after it starts.

The dm_WfmsTimer system administration job in the server checks the timers and sends the notices to the workflow supervisor. The dm_WfmsTimer job is installed with the system administration job suite. It is not installed in the active state. To use the timers in a process, make sure that the system administrator has activated the functionality. The dm_WfmsTimer tool runs every hour. The workflow supervisor receives warning notifications in the form of an item in the Inbox. For more information, see "Configuring notifications" on page 75.

2.2.9 Understanding activity transitions

If an activity has multiple outgoing flows, you can define what packages are passed to the following activities depending on the outcome of the activity. For example, you can configure a performer that reviews the design of a new form with the choice of forwarding the design to the next reviewer or sending it back to the initial designer for revision. You set up this branching logic by creating flows from this activity to the two possible activities, and allowing the performer to choose which path to follow.

The transition type defines how the following activities are selected when an activity is completed. There are three types of transitions:

- **Select all connected activities**: Packages are sent to all following activities linked to the activity, including both forward flows and reject flows.
- **Let performer select the next activities**: The performer of this activity chooses the activities to which the packages are sent at the run time.
- **Select next activities based on conditions**: The server determines the activities that receive packages at run time by evaluating a set of transition conditions.

In an activity, if the performer category is **Specific group**, you must specify the number of members that must complete the task before the server considers the overall activity complete and forwards the packages to the following activities. For example, if five users receive a work item for an activity, you can specify that the activity is complete when any three performers have completed the activity.

If you let performers select the next activities, you can limit the number of following activities the performer can select. For example, if an activity has three outgoing flows, the performer can decide number of following activites that receive the packages.

If a *group* of performers select the next activity, the performer category is **Specific group**, and the transition option is **Let performer select the next activity**, you also need to advise the server about how to combine the selection options for a performer. When a group selects activities, it is possible that some performers might select forward activities while others select reject activities. The server configuration determines the activities to start. If you choose an conditional transition type, you must define at least one transition condition for that activity.

2.2.9.1 Transition conditions

You use transition conditions to define how packages are routed depending on the results of an activity. A transition condition is a logical condition with one or more associated flows. When an activity is complete at run time, the server evaluates the transition conditions to determine which following activities to start as the next step in the process. The packages are passed to the activities associated with the first transition condition that is TRUE. An activity may have multiple transition conditions, however, the server starts at the first TRUE condition.

For example, you could define an activity that routes a document depending on whether the performer checked in a new version of the document. The server uses the following logic to determine where to send the document:

```
If
(New version checked in) then   Route to activity Evaluate Updates
Else
Route to activity Continue Approval
```

Starting with 24.2 release, you can use process variables in a transition condition for manual and auto activities to determine the next activity in a process.

A transition condition is a Boolean expression. The expression is used to check attributes of the package's components, the containing process, or the last completed work item. The transition condition should include at least a value or condition.

A transition condition should have an **Else** option that the server runs if none of the transition conditions are met. The **Else** option should not have a condition associated with it. An activity can only have one **Else** case. For information about defining transition conditions for an activity, see "Configuring activity transition rules" on page 76.

Chapter 3

Understanding Workflow Designer UI

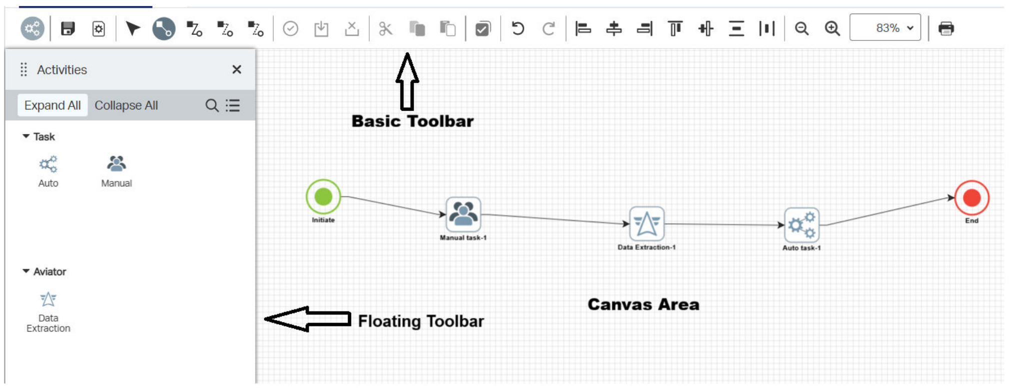
Workflow Designer is a web-based graphical tool for laying out and defining your process template. The Workflow Designer user interface is divided into two panes:

- Navigation pane or the left pane provides the option to navigate between processes. It also provides options to import, export, or migrate a process.
- Designer pane or right pane contains a canvas areas, an Activity toolbar, and a Function toolbar.

You can control the size of the two panes by positioning the mouse over the border between them and dragging the border to a new position.

You can also change to a single-pane view using the Pane button .

A configurable toolbar appears across the top of the window, providing quick access to common commands.



3.1 Role-based access

To use Workflow Designer you must have the OpenText Documentum CM license and the user should be a member of the `documentum_workflow_designer` role. For more information about OpenText Documentum CM license, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250200-IGD)* (EDCSY250200-IGD-EN-01). The following table describes the roles and the corresponding access level in Workflow Designer:

User role	Login access	Create and manage
OTDS User with valid license and `documentum_workflow_designer` role	Yes	Yes
OTDS User without valid license and `documentum_workflow_designer` role	No	No
install_owner with `documentum_workflow_designer` role	Yes	Yes
install_owner without `documentum_workflow_designer` role	Yes	No

3.2 Navigation pane

The Navigation pane enables you to navigate between processes. By default, the Workflow Designer opens in the **Processes** user interface.

3.2.1 Process navigation

The Navigation pane includes the following components:

- Processes menu
- New Process option
- Process list

3.2.1.1 Navigation menu

The navigation menu includes the following options:

- **Import**: Imports the selected process template from a package bundle into the Workflow Designer repository.
- **Export**: Exports the selected process template as a package bundle to the local system.
- **Migrate**: Migrates legacy process templates developed using Workflow Manager to the current Workflow Designer process.
- **Endpoints**: Lists all the available Endpoints and provides option to create a new Endpoint.
- **Processes**: Lists all the available process templates and provides option to create a new process template.
- **Email template**: Lists all the available email templates and provides option to create a new email template.
- **Resources**: Lists all the available resources and provides option to create a new resource.

3.2.1.2 New process

You can create a new process template using the **New Process** option in the **Processes** view pane. For more information about how to create a new process template, see "Creating a process template" on page 47 section.

3.2.1.3 Process list

This area lists all the process templates created using Workflow Designer in the connected repository. To load a process template in the designer area, click a process template from the Process list. You can also perform the following operations on a process template using the process context menu:

- Create From: Use this option to create a copy of the selected process template. For more information about how to create a copy of a process template, see "Duplicating a process template" on page 57 section.
- Delete: Use this option to delete a process template. For more information about how to delete a process template, see "Deleting a process template" on page 56 section.

The following table describes the process icons:

Icon	Description
⧉	Process is migrated but not installed.
⋲	Process is installed.
⟳	Data type is deleted or modified, or a process is out of sync. This icon disappears after you validate the process.
❗	Process has validation errors.

Note: Validation of expressions and package type attributes is not supported.

3.3 Designer pane

The Workflow Designer pane provides tools to create and configure process templates. The designer pane has a fixed designer toolbar at the top, a floating toolbar, and a canvas area for defining the layout of a process template.

3.3.1 Workflow Designer toolbar

Workflow Designer has a configurable toolbar at the top of the editor pane that provides quick access to common commands. To access a command from the toolbar, click the corresponding icon. To find out what command an icon corresponds to, position the mouse pointer over the icon to display the name of the icon. The following table lists the icons and its purpose:

Icon	Description
Open Palette	Open the Activities palette.
Save	Save the process template.
Process Properties	Open process properties dialog box.
Drag to select	Selection tool to select activities and flows in the canvas.
Straight Flow	Defines flow of information in a process template.
Multisegment Flow	Defines flow of information in a multilevel process template.
Reject Flow	Defines flow of information when an activity is rejected.
Fault Flow	Defines flow of information when an activity fails with error. **Note:** Fault flow is available only with the Advanced Workflow Designer license.
Validate	Validate the process template. **Note:** You can validate a specific version or a process version.
Install	Install the process template.
Uninstall	Uninstall the process template.
Cut	Cut the selected activity.

Icon	Description
Copy	Copy the selected activity.
Paste	Paste the selected activity to the canvas.
Select All	Select all the activites.
Undo	Undo the previous action.
Redo	Redo the previous action.
Align Left	Align the selected activities to the left.
Align Center (Vertically)	Align the selected activities to the center.
Align Right	Align the selected activities to the right.
Align Top	Align the selected activities to the top.
Align Center (Horizantally)	Align the selected activities to the middle.
Distribute Vertical Spacing	Align the selected activities as a single row.
Distribute Horizontal Spacing	Align the selected activities as a single column.
Zoom Out	Zoom out of the canvas.
Zoom In	Zoom into the canvas.
Print	Print the process template diagram in the canvas.

By default, the toolbar displays all the icons.

3.3.2 Floating toolbar

The floating toolbar contains the following tools and connectors:

Icon	Description
Automatic Activity	Adds an automatic activity to the process.
Manual Activity	Adds a manual activity to the process.
AI Workflow Automation Data Extraction	Adds an AI Workflow Automation activity to the process. It provides advanced data extraction and analysis functionality.

3.3.3 Process editor canvas

The canvas is the area where you layout the process template. To define a process template, drag the activities or other components from the toolbar into the canvas, connect them with the flows, and then define the properties of the activities and flows. See "Creating a process template" on page 47 for more information.

- To add an activity, drag it from the palette into the canvas.
- To move an activity within the process template, select it and drag to their new location. When you move an activity that has flows connecting it to other activities, the arrows representing the flows move along with the activity. Flows cannot be moved on their own.
- To remove an activity or flow from a process template, select it and click **Delete**.
- To copy an activity, select the activity and click **Copy** from the toolbar. To add the new copy to the process, click **Paste** from the toolbar.
- Select an activity to see the available actions that are available. A group of icons appears at the location of the mouse cursor, showing the available actions.

These actions require you to select the component you want to act on.

To select one or more components in the canvas:

1. Click an activity in the canvas.

   A dotted-line rectangle appears around the selected component, indicating that it is selected. Click outside the component to de-select it.

2. To select additional components, hold down the Shift key as you click each of the components.

   If you do not hold down the Shift key, selecting one activity or flow automatically de-selects any previously selected components.

3. To select all the components in the process, click Select All.

3.3.3.1 Aligning activities

The Alignment options enable you to position process activities precisely. You can align activities vertically or horizontally by their left or right edges, top or bottom edges, or by their center points.

To align activities:

1. Select the activities you want to align.
2. Click the icon corresponding to the intended alignment.

   The available alignment options are:

   - Align left
   - Align center
   - Align right
   - Align top
   - Align middle
   - Align row
   - Align column

   If you choose to align activities, the extreme most activity determines the placement of the other activities; that is, the selected activities will align the activity with most extreme position. For example, if you choose to align the bottom edges of your activities, the lowest or bottommost activity determines the placement of the other activities.
3. Click **Save**.

3.3.3.2 Zooming in or out

Each time you click Zoom In or Zoom Out, the editor pane zooms in or out by one magnification level. The current magnification level is displayed in the zoom list after you change the magnification level using the Zoom In or Zoom Out.

To zoom in or zoom out on a canvas:

1. Expand the drop-down list next to the Zoom In.
2. Select one of the zoom levels:
   - **150%**
   - **100%** (default)
   - **75%**
   - **50%**
   - **Fit**

     Magnifies or shrinks the appearance of your canvas so that it fits within the visible dimensions.
   - **Width**

     Sets the size of the canvas so that its full width fits within the visual dimensions of the canvas.
   - **Show grid line**

     Displays the grid lines in the Workflow Designer canvas area.
   - **Snap to grid**

     Aligns the selected components in the Workflow Designer canvas with the grid line

3.3.3.3 Printing a diagram

Workflow Designer 22.1 provides enhancements to the printing functionality, which includes options to select page size, page orientation, print margin, and print size. To print a diagram in the canvas:

1. From the Workflow Designer toolbar, click **Print**.
2. The **Print Setup Dialog** is opened to configure print properties. The following table describes the available print properties:

Parameter	Description
Page Format	Select the page size to print the diagram. • **A3** (11.7 x 16.5 inch) • **A4** (8.3 x 11.7 inch) • **A5** (5.8 x 8.3 inch) • **US Executive** (8.5 x 11 inch) • **US Legal** (8.5 x 14 inch) • **US Letter** (7.5 x 10.5 inch)
Page Orientation	Select the page layout: • **Landscape** • **Portrait**
Margins	Select the unit of measurement as inches or centimeters and corresponding margins in the selected unit. • **Top** • **Bottom** • **Left** • **Right** **Note:** If the specified margins exceed the page sizes, auto-validation raises the error flag.
Printoput size	Select the printout size: • **Actual size**: Print the actual size of the diagram, which may span multiple pages. • **Same as paper size**: Print on the selected page. The diagram is fitted on a single page. • **Fit to**: Print the diagram across the number of pages specified in the `Pages(s) across` x `Down` format.

3. Click **Print**.

3.4 Process properties

The process properties are categorized under three tabs: **General**, **Data**, and **Aviator**.

The following table describes the various process properties on the **General** tab:

Parameter	Description
Label	Name of the process template.
System Name	Repository system name.
Description	Process template description.
Default Alias Set	Default alias set for the process template.
Create New Alias Set	Creates a new alias set that is automatically set as the default alias set for the process template.
Workflow instruction	Additional workflow instructions.
Process can run in stateless mode	Process can run in stateless mode at runtime.
Audit Process Events	Enables logging workflow events in the audit log.
Always show validate and install prompt after saving the process	Enables the validate and install prompt to appear after a process template is saved.
Default Calendar	Default calendar for the process template, the available options are: • **Use standard calendar**: The system uses the standard calendar to calculate timer duration. The standard calendar is based on absolute time. It is a 24x7 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.

The following table describes the various process properties for packages on the **Data** tab:

Parameter	Description
Packages	Lists all the packages available for the process template.
Name	Package name.
Type	Package type.
Version	Named version number, can be text or numeric.

Parameter	Description
Visible	Whether the package is available to all the activities in the process template.
Mandatory	Whether this package is mandatory for every activity.

The following table describes the various process properties for variable on the **Data** tab:

Parameter	Description
Variables	Lists of variables available for the process template.
Name	Variable name.
Type	Type of variable, available types are: • String • Integer • Float • Boolean • Datetime You can also specify number of values a variable can store. Available options are: • Single value • Multi value
Default Value	Default value of the variable.

Note: To hide a process variable for an activity, clear **Visible at this activity** option for the specific activity.

The following table describes query space process properties available on the **Aviator** tab:

Parameter	Description
Name	Internal query space name that the system uses.
Label	Query space label that is displayed to the user.
Required packages	List of documents required for the query space.
Optional packages	List of optional documents for the query space. If there are no required packages, add a minimum of one optional package for the query space.

Parameter	Description
Prompt Instructions	Additional instructions for the AI Workflow Automation activity to improve the response. These instructions are appended to the query and indicates the role, behavior, or the format to use while retrieving the information. For example, “You are a senior mortgage underwriter (DE Certified, 15+ years experience) specializing in the four risk factors: credit, capacity, capital, and collateral. Provide concise, expert-level responses focusing on risk assessment, underwriting best practices, and regulatory compliance.”
Predefined query set	List of queries defined in a query space. To add a new query to query space, use **Add query** option. For more information about how to create a query, see “Setting process properties” on page 50.

3.5 Activity properties

The activity properties are categorized into ten tabs. The tabs available for each activity depend on its functionality:

- Task
- General
- Performer
- Execution
- Trigger
- Timers
- Transitions
- Data
- Aviator
- Display

3.5.1 Task tab

The **Task** tab is available for **Manual** and **Auto** activities. The following table describes the parameters on the **Task** tab:

Parameter	Description
Subject	Basic description of the activity.
Instructions	Special instructions for an activity.
Priority	Priority of a task. Available options are: • Low • Medium • High

3.5.2 General tab

The **General** tab is available for **AI Workflow Automation** activity. The following table describes the parameters on the **General** tab:

Note: General tab provides parameters to define the output of an activity mapping to the process and/or object data.

Parameter	Description
Output message mapping (AI Workflow Automation activity)	Defines output of an activity mapping to process data.

3.5.3 Performer tab

The following table describes the parameters for an **Auto** activity on the **Performer** tab:

Parameter	Description
Execution Setting	
Perform this task as	Performs this task on behalf of the workflow supervisor, repository owner, or the performer of the last activity.
Automatic Execution Parameters	

Parameter	Description
Execute this method automatically	Action to automatically perform from the Execute this method automatically list. The actions in the list are workflow methods. **Notes** • To make a custom method available here, the attribute *`a_special_app`* must be set. *`a_special_app`* is a dm_sysobject attribute reserved for use by Documentum products. This attribute must have the value `Workflow`. • 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.
Timeout (sec)	Time-out for an activity. The valid range of time-out is from 0 to 2147483647 seconds.
Enable troubleshooting logging	Enables the event logging for troubleshooting purpose. The execution results appear in the `/Temp/BPM/Method Exec Results` repository folder. Activity failures appear in the `/Temp/BPM/Method Exceptions` repository folder.
Exception handling	Actions for handling exceptions.
Retry on timeout or exception	• *`Maximum tries`*: Specifies the maximum number of attempts on timeout exception. • *`Interval (min)`*: Specifies the time interval between each retry attempt.
On failure	Action to perform when an auto activity fails. • *`Continue`*: Proceed to next activity. • *`Terminate Process`*: Terminates the process. • *`Halt Process`*: Halts the process.

The following table describes the parameters for the **Manual** activity on the **Performer** tab:

Parameter	Description
Performers	List of performers of the task as determined by the system. Available options are: • **Workflow Supervisor**: The system selects the user designated as the workflow supervisor when the activity starts. By default, the user that starts the Process is the workflow supervisor. • **Repository Owner**: The system selects the user identified as the owner of the active Documentum repository. **Note:** You must have SUPERUSER privileges to install the process if this performer is used in the auto activity of the process. • **Performer From Last Activity**: The system selects the performer from the previous finished activity that satisfied the trigger condition of the current activity. The Performer from 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 identified at run time because the performer can be from a group. **Note:** You must have SUPERUSER privileges to install the process if this performer is used in the auto activity of the process. • **Previous Activity Performer to Choose at Runtime**: The performer of the preceding activity selects the performer of this activity at run time. If this activity includes multiple preceding activities, use the drop-down list to select a performer from the relevant activity who will perform this activity. **Notes** – You can only select this performer type if the activity does not have any other performer type selected from the list. – If this performer type is already selected in the list, then you cannot select any other performer

Parameter	Description
	type unless this performer type is removed from the selected list. • **Specific Group**: You select a group from the list of groups from the connected repository. The server assigns a work item to each performer in the group at the run time. • **Group from Alias Set**: You select a group alias from an alias set as the performer of the activity. The server assigns a work item to each performer in the group alias at run time. • **User from Alias Set**: You select a user alias from an alias set as the performer of the activity. The server assigns a work item to a member of a user alias at run time. For more information about how to configure a performer for a manual activity, see “Configuring performers for a manual activity” on page 69.

Parameter	Description
Processed By	• **Individual users and group member at the same time**: Sends one task to 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 accepts the task. • **Individual user or group member with fewest tasks**: Sends one task to the performer who has the fewest pending tasks. • **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 is completed, 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.
Task Options	Category of available task options.
Allow user to delegate task	Specifies how to distribute the task if there is a problem with automatic delegation. Available options are: • Assign to workflow supervisor if auto-delegation fails • Assign to original performer if auto-delegation fails
Allow performer to add additional performers to this task	Enables the performer to add additional performers to complete the same task.
Performer sign-off is required	Specifies that this activity requires a sign-off before it is completed. This option includes adding an electronic signature to the task.
Allow sequential task performers to reject to previous performer	Specify whether these performers can reject the task and return it to the previous performer in the sequence, if multiple sequential task performers are selected.

3.5.4 Execution tab

The **Execution** tab is available for **AI Workflow Automation** activity. The following table describes the parameters on the **Execution** tab:

Parameter	Description
Execution Setting	Category of available performer related options.
Perform this task as	Performs this task on behalf of the workflow supervisor, repository owner, or the performer of the last activity.
Timeout (sec)	Time-out for an activity. The valid range of time-out is from 0 to 2147483647 seconds. Default value is 60 seconds.
Enable troubleshooting logging	Enables the event logging for troubleshooting purpose. The execution results appear in the `/Temp/BPM/Method Exec Results` repository folder. Activity failures appear in the `/Temp/BPM/Method Exceptions` repository folder.
Exception handling	Options available for handling exceptions in a process.
Retry on timeout or exception	• *`Maximum tries`*: Specifies the maximum number of attempts on timeout exception. • *`Interval (min)`*: Specifies the time interval between each retry attempt.
Persist process data	Save the successfully mapped package attributes and process variables when a mapping error or exception occurs. **Note:** This option is available only for AI Workflow Automation activity.
On failure	Action to perform when an auto activity fails. • *`Continue`*: Proceed to next activity. • *`Terminate Process`*: Terminates the process. • *`Halt Process`*: Halts the process.

3.5.5 Trigger tab

The following table describes the parameters on the **Trigger** tab:

Parameter	Description
Trigger	Available activity trigger options.
All input flows are selected	Indicates that the activity is triggered when all the input flows are selected.
The number of inputs flows selected	Minimum number of input flows to trigger an activity.
And when the event signal arrives	Event to trigger an activity.
The activity can run more than once in a workflow	Indicates whether an activity can run multiple times in a workflow.

3.5.6 Notification tab

The following table describes the parameters in the **Notifications** tab:

Parameter	Description
Notify the workflow's supervisor when	
The activity does not trigger within	Time limit within which an activity must start.
The activity work is not finished	Time limit within which an activity must complete.

3.5.7 Transition tab

The following table describes the parameters in the **Transition** tab:

Parameter	Description
When this activity completes	Logic to determine the next activity. You can select from the following options: • **Select all connected activities**: Indicates to select all the activities connected to the current activity. • **Let performer select the next activities**: Indicates that the performer of current activity selects the next activity. • **Select next activities based on conditions**: Indicates to select the next activity based on the specified condition.
If the **Select all connected activities** option is selected:	
All performers complete the task	Indicates that all the assigned performers complete the activity.

Parameter	Description
Performers complete the task	Indicates the minimum number of performers who should complete the activity.
If the **Let performer select the next activities** option is selected:	
Select up to	Number of next activities that can be configured.
Complete the activity when	
All performers should complete the task	Activity is marked complete only when all the performers have completed the task.
Number of performers complete the task	Activity is marked complete only when the specified number of performers have completed the task.
Start next activity when	• Any performer rejects • Any performer forwards • All performers complete the task
If both reject and forward activities are selected	• Start all selected activities • Start only rejected activities • Start only forward activities
If **Select next activities based on conditions** is selected, table with following fields is displayed:	
Logic	If-else condition construct to define the logic to select the next activity in the process.
Transition	Transition condition for the logic.
Select Statement	Defines the next activity when the transition condition returns True.

3.5.8 Data tab

The **Data** tab provides the following information about the packages and variables defined in a process:

Parameter	Description
Name	Name of the package.
Type	Type of the package.
Version	Version of the package.
Visible	Indicates whether the package is visible for this activity.
Mandatory	Indicates whether the package is mandatory for this activity.

The **Data** tab provides the following information about the variables:

Parameter	Description
Name	Variable name.
Type	Variable types. Available options are: • STRING • INTEGER • FLOAT • BOOLEAN • DATETIME Choose between single or multi value to specify number of values a variable can store. Available options are: • Single value • Multi value
Default Value	Default value of the variable.

Note: To hide a Package for an activity, clear the **Visible at this activity** option for the specific activity.

3.5.9 Aviator tab

The **Aviator** tab provides information about query spaces and query context fields available for **Manual** and **AI Workflow Automation** activity:

Parameter	Description
Name	Read-only. Name of the query space.
Label	Read-only. Query space label.
Enable Query Space	Enable or disable the selected query space for the activity.
Required packages	Read-only. List of documents required for the query space.
Optional packages	Read-only. List of optional documents for the query space. If there are no required packages, add a minimum of one optional package for the query space.

Parameter	Description
Prompt Instructions	Additional instructions for the AI Workflow Automation activity to improve the response. These instructions are appended to the query and indicates the role, behavior, or the format to use while retrieving the information. For example, “You are a senior mortgage underwriter (DE Certified, 15+ years experience) specializing in the four risk factors: credit, capacity, capital, and collateral. Provide concise, expert-level responses focusing on risk assessment, underwriting best practices, and regulatory compliance.”
Predefined query set	List of queries defined in a query space. Provides the option to change the current state of a query in a query space, for example; enabled or disabled.

3.5.10 Display tab

The **Display** tab provides the following information about label text settings:

Parameter	Description
Label Styling	
Font	Font type to use for label.
Point Size	Size of the font.

Chapter 4

Defining and managing processes templates

A process templates represent the business process through which a given object or set of objects flows. They define the overall process from beginning to end. You create process templates in Workflow Designer that can be reused to create individual process instances.

This chapter explains how to create process templates, validate them, and install them.

4.1 Opening an existing process template

You can open an existing process template to review, revise, or create a copy of the process template to use it as a starting point for a new process.

Note: To update a process template, you must first uninstall the process template.

To open a process template in Workflow Designer, select or double-click a process template from the **Process** navigation pane, to open it in the canvas.

4.2 Creating a process template

1. Design a process template that implements a business process.

   For details about designing process templates, see "Defining process activities" on page 16.

2. In the **Processes** navigation pane, click **New Process**.

3. Enter process template name in the **Label** field. Set the process properties. For more information about process properties, see "Setting process properties" on page 50.

   Note: A process template name can have maximum of 127 characters and all the special characters are allowed except for the percentage (%).

4. Click **Save**.

   The Initiate and End tasks are added in the canvas for the new process template.

5. Drag and drop activities from the **Activities** palette on to the canvas until you have one activity for each task in your process template. Available activities are:

   - **Manual Activity**
   - **Auto Activity**

- **AI Workflow Automation Activity**

6. Connect each activity to the activity that precedes it in the logical flow.

   The first activity in the process template must be connected to the Initiate task, and the last activity must be connected to the End task.

   To connect two activities, select one of the flows, move your mouse over the first activity until you see its selection box, then drag the mouse to the second activity. Release the mouse button when you see the selection box for the second activity to draw a line between the two activities.

   You connect activities using one of three create flow icons in the Workflow Designer toolbar:

   - To connect activities in a forward movement of data, use either the **Straight Flow** or the **Multisegment Flow**. The difference between the two is visual: one draws a straight line to represent the flow between activities, the other draws a line consisting of multiple segments.
   - To connect activities in a backward movement of data, use **Reject Flow**. Reject flow represent the path taken when an activity is rejected.
   - To handle faults in an activity, use **Fault Flow**. The fault flow represents the alternate path that is taken when an activity fails, redirecting the flow to a manual or automatic activity. After the task is completed, the flow resumes from the next activity after the activity.

7. Configure packages at process level.

8. Configure each activity.

9. Adjust the visual layout as necessary.

   For information about the options available for laying out the process template display, see “Process editor canvas” on page 30.

10. Save the process template.

    See “Saving a process template” on page 55.

11. Validate the process template.

    See “Validating a process template” on page 55.

12. Install the process template.

    See “Installing a process template” on page 55. After you have installed the process, it is available to users.

4.3 Configuring the default alias set

1. From the **Processes** list, select a process template.
2. Click **Process Properties**. 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 Workflow Designer.
4. Click **OK** to mark an alias set as the default alias set.

4.4 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. Go 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 Documentum 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 run time only.

4.5 Adding a performer from a specific group

1. Go to the **Performers** tab and click **Add Performer**.
2. In the **Add Performer** dialog box, select **Specific group** from the **Select a performer who will work on a task** list.
3. Select the **Group** option to view all the Group type performers or the **Roles** option to view all the Role type performers for a specific group.
4. To further filter the performer list, enter a search string in the **Select the user parameters that will contain the performer** field and click **Search**. If the resultant list of performers is more than 10, the result is split into multiple pages. You can use the navigation buttons above the result list to navigate to different result pages.
5. Select a performer from the result list.
6. Click **Add**.

4.6 Modifying a process template

You can modify a process template by changing its flow or activity definitions. When you change a template, you can either overwrite the changes or create a new process template. Any changes you make are governed by component-level permissions.

You must uninstall the process template before you modify it. When you uninstall a template, all the process instances are halted. You can create a copy of a template without uninstalling it. Updating a process template has impact on the active processes, you must update the process template when there are no active processes running.

The updated process template must be validated and installed before you can start a process based on it.

See also "Saving a process template" on page 55.

4.7 Setting process properties

You use the **Process Properties** dialog box to enter basic details about the process template you are creating. The original creator and current state of the process template is displayed in the dialog box.

To set process properties:

1. On the toolbar, click **Process Properties**.

2. On the **General** tab, enter the following details:

Field	Description
Label	Name of the process template.
System Name	Name of the system.
Description	Description of the process template.
Default Alias Set	Default alias set for a process template from the list of alias sets.
Create New AliasSet	Creates a new alias set that is automatically set as default alias set for the process template in the Workflow Designer. • Alias set label: Unique name for the alias set. • Alias set description: Description of the alias set.
Workflow Instructions	Instructions for the performer of the activities.

Field	Description
Process Settings	
Audit Process Events	Enables event auditing for system events.
Always show validate and install prompts after saving the process	System should prompt to validate and install a process template after it is saved.
Default Calendar	
Use standard calendar	System uses standard calendar.
User specific business calendar	System uses user specific business calendar.

3. Click the **Data** tab.
4. Click **Packages** to add a new package.
5. Click Ellipsis for **Packages** and click **Add Package** from the menu.
6. In the **Name** field, enter name of the package.
7. Click Ellipsis for **Type** field and select the package type in the **Package Type** dialog box. Complete one of the following steps to select an object type:
   - Click **Filter** to filter object types. Select an object type category and click the back arrow icon to return to the **Package Type** dialog box. Select an object type from the list and click **OK**.
   - Search for the package type in the **Type or select package type** field and select package type from the list. Click **OK**.
   - Use the scroll-bar to locate and select a package type and click **OK**.
8. In the **Version** text-field, specify a version for the package or type `CURRENT`.
9. (Optional) Select the **Visible across entire process** option to make the package visible to every activity in the process.
10. (Optional) Select the **This is a mandatory package** option to mark the package as mandatory by the process.
11. (Optional) Click Ellipsis for the package and click **Delete** from the popup menu to delete a package.
12. Click **Variables** to add a process variable. You must define process variables for mapping information extracted using the AI Workflow Automation activity.
13. Click Ellipsis for **Variables** and click **Add Variable** from the menu.
14. In the **Name** field, enter name of the variable .
15. Select the variable type from the **Type** list:
    - **String**

- **Integer**
- **Float**
- **Boolean**
- **Datetime**

  You can also specify number of values a variable can store, available options are:

- **Single value**
- **Multi value**

16. In the **Default Value** field, specify default value for the variable.
17. Click the **Aviator** tab.
18. In the **Query Space** area, click **Add** to add a new query space. A new query space is created and listed in the **Query Space** list. for more information about how to create query space, see "Adding a query space" on page 52
19. In the **Predefined query set** area, click **Add query** to define a query for the selected packages. For more information about how to add a query, see "Adding a query" on page 53.
20. Click **Ok** to save the query space.

4.7.1 Query spaces

This section describes

4.7.1.1 Adding a query space

1. Go to the **Aviator** tab in the **Process properties** dialog box.
2. Click the **Aviator** tab.
3. In the **Query Space** area, click **Add** to add a new query space. A new query space is created and listed in the **Query Space** list.
4. In the **Name** box, enter the name of the query space.
5. In the **Label** box, enter the label for the query space.
6. In the **Query context** area, from **Required packages** list, select the required package(s) for the query space.

   Notes

   - You must add at least one package to a query space.
   - Aviator query spaces support only packages that use the current version binding. Adding a package with a different binding type results in an error during process validation.

- Ensure that all the documents and packages are pre-embedded so that the AI Workflow Automation activity can access and query them. For more information about how to embed documents and packages, see *OpenText Documentum Content Management - Smart View User Guide (EDCCL250400-UGD)*.
- Only packages of the `dm_document` type and its sub-types are supported.
- Ensure that all the relevant documents are added for AI Workflow Automation activity to retrieve the requested information.

7. From the **Optional packages** list, select optional packages if required.
8. In the **Prompt Instructions** area, enter the additional instructions for the Aviator to retrieve the content optimally and effectively.
9. In the **Predefined query set** area, click **Add query** to define a query for the selected packages. For more information about how to add a query, see "Adding a query" on page 53.

Note: You can define multiple query spaces for a process and each query space can have multiple queries.

4.7.1.2 Adding a query

1. Go to the **Aviator** tab in the **Process properties** dialog box.
2. Select a query space.
3. In the **Predefined query set** area, click **Add query** to define a query for the selected packages.
4. In the **Name** box, enter name of the query.
5. In the **Label** box, enter label for the query.
6. In the **Query** box, enter the query definition. You can refer to packages, process variables, and queries defined in the same query space to define a query. You can also use Context selector to insert variables in a query. For more information about how to use Context Selector, see "Using Context selector to insert process variables in a query" on page 54.
7. From the **Answer mode** list, select a query mode of response. Answer mode instructs the AI Workflow Automation activity to respond in a predefined format:

   - **NotSet**: Customized response, to control the answer format using the custom prompt instructions. For example, enter: "You are a loan underwriting expert who provides concise responses under 1024 characters. Please give the response in the HTML format, wrapped in <div> tag." in the **Prompt Instruction** field.
   - **Precise**: Precise response in one or two words only.

- **Short**: Short response in the form of a meaningful sentence.
- **Detailed**: Detailed query response.

8. Click **Add** to add the query and close the **New Query** dialog box.

Notes

- Before adding a query in query space for AI Workflow Automation activity, run the query in Content Aviator on a similar set of documents to verify that the response is consistent and as expected.
- If the model version or type is changed in Aviator Model Service, re-test the workflow on a similar set of documents to verify that the query responses are consistent and as expected. Also, re-test the workflow whenever you modify process data (such as packages or variables) or update query configurations (query spaces or individual queries).

4.7.1.3 Using Context selector to insert process variables in a query

Starting with Workflow Designer release 25.4, you can insert process variables in a query using Context selector.

1. Click the Ellipses button for **Query** to open the **Context selector** dialog box.
2. In the **Process** tab, select a package attribute to insert in the query. You can add multiple packages attributes in a query.
3. Click the **Aviator** tab and select a query from query space to insert the corresponding process variable in the query.
4. Click **Apply** to insert the selected query and close the **Context selector** dialog box. For example:

```
//Syntax
${process.ai.queries.queryspacename.queryname}?
//example
${process.ai.queries.clientInfo.clientName}?
// Where clientInfo is the queryspacename and clientName is the queryname
```

4.7.1.4 Editing a query

1. Go to the **Aviator** tab in the **Process properties** dialog box.
2. Select a query space.
3. Click the Ellipsis icon for the query and select **Edit**.
4. Update the required fields in the **Edit Query** dialog box and click **Save**.

4.7.1.5 Deleting a query in a query space

1. Select a query space.
2. Click the Ellipsis icon corresponding to the query to open the context menu.
3. Click **Delete** to delete the query.

4.7.1.6 Deleting a query space

1. Select a query space.
2. Click the Ellipsis icon corresponding to the query space to open context menu.
3. Click **Delete** to delete the query space.

4.8 Saving a process template

When you have completed a process template, you must save it before you can validate and install it. Saving the process template copies your changes to the repository.

To save a process template to repository, click **Save**.

Note: If you do not save the changes to repository and close the browser, all the changes to the process template are lost.

4.9 Validating a process template

Validating a process template verifies that the process defined in the process meets system requirements.

Before validating, you need to save the process template. If validation fails, error information is listed in the Errors panel at the bottom of the canvas. To validate a process template, click **Validate**.

4.10 Installing a process template

A process template must be installed before it is available for use in an active process. To install a process template, make sure that it is validated.

If you need to make changes to an installed process, you must uninstall the process first. When you uninstall a process template, all the active processes based on the process template are halted. After making the changes, validate and install the process template again.

4.11 Uninstalling a process template

You must uninstall a process template before modifying it.

- If a process template has workflow instances, only a user with *`sysadmin`* permission and `documentum_workflow_designer` role can uninstall it.
- If a process template has no workflow instance, any user with `documentum_workflow_designer` role can uninstall it.

4.12 Deleting a process template

1. Hover the mouse pointer over the process template to delete and click Ellipsis to open the menu for the process.
2. Select **Delete** from the context menu.
3. Click **OK** to confirm.

When you delete a process template:

- A warning message is displayed for an installed process that has an active instance.
- All the process-related data and active instances are deleted permanently.
- Any open process in the canvas is closed and removed from the navigation pane. The focus moves to next open process tab.

Notes

- You cannot delete a process while another user is editing that process.
- The delete process may take a long time to complete for installed processes and the Workflow Designer may become unresponsive during the delete process.
- For a process instance, only a user with *`sysadmin`* permission and `documentum_workflow_designer` role can delete it.
- If there is no process instance, any user with the `documentum_workflow_designer` role can delete it.

4.13 Duplicating a process template

To create a copy of an existing process template, complete the following steps:

1. Hover the mouse pointer over the process template to copy and click **Ellipsis** button to open the context menu for the process template.
2. Click **Create From** in the context menu.
3. Enter the process template name in the **Label** field and click **Save**.

   Note: A duplicate copy of the process template with all the attributes is created with the specified name and added under the process list.

4.14 Exporting a process template

You can export a process template created in Workflow Designer to the local system. To export a process template, complete the following steps:

1. In the **Process Navigator** pane, click **Processes** to open the **Processes** menu.
2. Click **Export**.
3. In the **Export Process** dialog box, select a process template to export from the list of process templates. You can also select multiple processes templates to export.

   Note: Starting with Workflow Designer 22.2, a new option, **Validated**, is added that you can use to list the validated processes when exporting a process. If the **Validated** option is enabled, only the validated process templates are listed in the **Export Process** dialog box. To view all the available process templates for export, disable the **Validated** option.

4. Click **Export**.
5. Save the .pkg file containing all the selected process templates and process dependencies to the local system.

Notes

- When you export a process template, only the process templates and definitions are exported. Process dependencies such as package types, users, groups, aliassets, and methods are not exported. You must export the dependencies as a .dar file and install into the destination repository before exporting the process template. A process template installed before installing the dependencies is disabled for export.
- You can import the downloaded package (.pkg file) to another repository.
- A process template name can contain a maximum of 127 characters.

- Documentum Composer is not supported for exporting a Workflow Designer process.

4.15 Importing a process template

You can import a process template from the file system to Workflow Designer. A process template is imported as a package with .pkg file extension. A package may contain multiple process templates. To import a process template, complete the following steps:

1. In the **Process Navigator** pane, click **Processes** to open the **Processes** context menu.
2. Click **Import**.
3. In the **Import Processes** dialog box, in the **Add processes file** area, click **Click here to add files**.

   Note: You can also drag and drop a package into the **Add processes file** area to import the process templates in a package.

4. In the **File Upload** dialog box, go to the location of the package file (.pkg) and click **Open**. The package is analyzed and all the process templates in the package are listed in the dialog box.

   Notes

   - If a process template has any discrepancy in dependencies (package types, Alias sets or Groups), the select option is disabled because the Workflow Designer does not map the dependencies for the imported process templates.
   - If a process template has no dependencies or a process template with dependency exists in the destination repository, the select option is enabled.
   - While importing a process template, modifying the process template may lead to loss of data.

5. Select the required process template from the list in the **Import Processes** dialog box and click **Next**.
6. If you import a process template with a name that already exists in the destination repository, a conflict is flagged. Such process templates are highlighted in red font in the **Import Processes** dialog box. You can either override a process template or rename the process template you are importing to resolve the conflict.

   a. To override a process template with an imported process template, select the check box corresponding to the process template.

Note: The process template must not be in the installed state for the override functionality to work. You cannot override an installed process.

b. To rename the imported process template, click **Edit**, enter a new name for the imported process template and click **Tick** .

7. Click **Import** after resolving any conflicts to import the process templates.

If a process template has any discrepancy in dependencies (package types, Aliassets, or Groups), the select option is disabled because the Workflow Designer does not map the dependencies for the process templates to be imported.

- If a process template has no dependencies or a process template with dependency exists in the destination repository, the select option is enabled.
- While importing a process template, modifying the process template may lead to loss of data.
- Green tick: Indicates that the process tempalte is imported successfully.
- Red Info: Indicates that the process template is not imported.

Notes

- Workflow Designer supports process template with name upto 127 characters. If a process template name contains more than 127 characters, it is not imported.
- If you delete a process template from a Web browser and import another process template with the same name from another Web browser, a conflict is flagged.
- Composer is not supported for importing a Workflow Designer process template.

4.16 Migrating a process template from Workflow Manager to Workflow Designer

You can migrate a process template from the Workflow Manager repository to Workflow Designer repository using the **Migrate** menu option.

Notes

- If you migrate a process template that has transition conditions with attributes that are not supported in Workflow Designer (for example, process metadata), process validation does not capture such migration errors. In Workflow Designer, transition conditions are validated when you open the transition condition editor from the activity properties.
- While migrating a process template, modifying the process template may lead to loss of data.

To migrate a process template, complete the following steps:

1. In the **Process Navigator** pane, click **Processes** to open the **Processes** context menu.
2. Click **Migrate**.
3. In the **Migrate Processes** dialog box, select the process template to migrate and click **Next**. You can select multiple process templates for migration.
4. If you migrate a process template with a name that already exists in the destination repository, a conflict is flagged. Such process templates are highlighted in red font in the **Migrate Processes** dialog box. You can either override the process template or rename the process template you are migrating to resolve the conflict.

   a. To override an existing process template with the migrated process template, select the **Override** check box corresponding to the process template.

      Note: You cannot override an installed process template.

   b. To rename the migrated process template, click **Edit**, enter a new name for the imported process template in the **Target** field, and click **Tick**.
5. Click **Migrate** after resolving any conflicts to migrate the process templates.

If a process template with dependencies is already installed in the destination repository, the imported process template is not installed and only process templates without any conflict are imported. The status column indicates the migration status for a process template:

- Green check mark: Indicates that the process template is imported successfully
- Red Info: Indicates that the process template is not migrated

4.17 Managing Workflow Designer logging

You can enable Workflow Designer logging in the application server using the sample `log4j2.properties` file available in the `DocumentumWorkflowDesigner/WEB-INF/logger/` folder. The following log files are generated:

- `WFDesigner.log`: Contains all the log messages.
- `WFMigration.log`: Contains all the log messages related to migration.
- `WFImport.log`: Contains all the log messages related to import and export.

Note: You can access Workflow Designer logs for more information about auto sync related errors.

4.18 Changing default alias set for a process template

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

1. Select a process template from the **Processes list**.
2. click **Process Properties**.
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.

4.19 Concepts

4.19.1 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.

4.19.2 Processes

A process is an instance of a process model at run time. Multiple processes based on the same model can run concurrently because the process model is separate from its run time 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.

4.19.3 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.

4.19.4 Documentum Content Aviator

Documentum Content Aviator is an OpenText artificial intelligence enabled conversational search feature that helps you to search and query the content that is stored in the Documentum folders. Content Aviator deployment enables the communication between the OpenText Documentum Content Management and Content Aviator that is deployed in a Google Cloud Platform hosted by OpenText.

Starting with Workflow Designer 25.2, you can create context-based custom prompts for a workflow. The predefined custom prompts encapsulate complex queries that are available to the user at runtime through Aviator. You can configure complex queries as custom prompts to retrieve information from one or more documents (packages) in a process. Content Aviator is an interface that enables you to query packages using natural language.

A query space is a set of queries and related packages that are available in a process. You can configure a query to retrieve reliable and relevant information from one or more packages that aids in decision making with reduced human error. A query space enables you to group similar queries and packages together.

4.19.4.1 Loan application – Content Aviator use case

Consider a case of loan application process. A typical loan application process requires information about the applicant, credit history, bank statement, and current liabilities of the applicant. You can configure a loan application process in Workflow Designer with Content Aviator capabilities that can query multiple documents of an applicant and retrieve reliable and relevant information. This can make decision making easier. You can create a loan application process in a workflow with following documents:

- Client Information: Contains basic information about the customer such as age, address, marital status, job profile, experience, and reason for the loan.
- Credit history: Contains information about customers previous loan repayments, credit card payments, and settlements.
- Bank statement: Lists the monthly transactions in a customer account. It can be multiple documents depending on the number of accounts a customer holds.
- Current liabilities: Contains information about the current liabilities of a customer in terms of loans and credit card outstanding.

To process these documents, you can define multiple queries that can retrieve information for improved understanding of a customer’s application and for better decision making. The following list provides the sample queries:

- Is the application document complete: Verifies whether all the mandatory fields required for an application are filled.
- What is the current status of the application: Retrieves the status of the application based on whether all the mandatory fields are populated.

- Have all the supporting documents been submitted: Verify whether all the required documents are attached with the application.
- What is the credit history of the applicant: Retrieves the credit history of the applicant, flags if there is any instance of cheque bounce or missed payments and installments of loans.
- What is the current rating of the applicant: Retrieves information from multiple documents to check the credit history, age, bank statement, and current address of an application and provides a score based on the weightage given to each field.

4.19.5 AI Workflow Automation activity

Starting with Documentum Workflow Designer release 25.4, you can use the AI Workflow Automation activity to automate data extraction and information processing from an unstructured document. A typical workflow involve manual data analysis, interpretation of data, and classification. AI Workflow Automation activity speeds up information retrieval, reduces manual effort, and minimizes errors when retrieving standard information from a document.

The AI Workflow Automation activity uses the queries defined in a query space to retrieve information from a package(s). The retrieved information is mapped to process data for use in the downstream activities and transitions. All the queries and packages that are part of a process are available to the AI Workflow Automation activity. You can select multiple query spaces containing queries while configuring an AI Workflow Automation activity.

Notes

- AI Workflow Automation activity returns query responses as string type. To ensure compatibility, it is recommended to map these responses to process variables or package attributes of type STRING only.
- The response from an AI Workflow Automation activity may be inaccurate. Ensure that your workflow is designed to handle any inconsistencies and inaccuracies in the responses.

4.19.5.1 Insurance claim process with AI Workflow Automation activity – Use case

Insurance claim process requires information about the claimant, insurance policy, discharge summary, and bills. You can create an insurance claim process in Workflow Designer and use the AI Workflow Automation activity to process and analyze the claim documents, extract and store the information to objects, summarize the information, interpret the information to take decisions, and automate the response to the claimant.

Using the AI Workflow Automation activity in Workflow designer, you can automate end-to-end claim processing, reduce manual errors and processing time, improve claimant experience through timely updates, and support compliance and audit readiness with structured data handling (AI).

A typical insurance claim application require the following packages as input to the claim process:

- Claim details
- Discharge summary
- Hospital bills
- Insurance policy

The AI Workflow Automation activity enables you to automate key steps in the claim lifecycle – from document classification to decision-making and ensuring that the claim is processed efficiently. You can configure AI Workflow Automation activity to perform the following functions:

- Document Classification using AI Workflow Automation activity: You can configure AI Workflow Automation activity to automatically sort the submitted documents into appropriate categories. For example, you can use the following queries to categorize the documents:
  - Claim forms (Is the provided document a claim document? Answer with 'yes' or 'no'.)
  - Policy document (Is the provided document a policy document? Answer with 'yes' or 'no'.)
  - Prescription records and bills (Is the provided document a bill or prescription document? Answer with 'yes' or 'no'.)
  - Discharge summary (Is the provided document a discharge summary document? Answer with 'yes' or 'no'.)
- Document Verification using the AI Workflow Automation activity: Next, you can configure the AI Workflow Automation activity to verify authenticity of the submitted documents to ensure that they belong to same claimant. For example, you can use the following query to verify document relationship:
  - Verification Query: Are these provided documents related? answer with 'yes' or 'no'
- Information Extraction using the AI Workflow Automation activity: Next, you can configure the AI Workflow Automation activity to extract information from the claim documents and store it in process database (process objects). For example, you can use following queries to extract information from a document:
  - What is the full name of the claimant from policy document?
  - What is the date of birth of $ {process.ai.Queries.ExtractPolicyData_QuerySpace.claimant_name_Query} from policy document?
  - What is the policy number of $ {process.ai.Queries.ExtractPolicyData.claimant_name} from policy document?

  - What is the diagnosis name in the provided document?
  - what is the procedure name in the provided document?
  - what is the total bill in the provided document? Return only number without any formatting.
- Risk Assessment and Analysis using the AI Workflow Automation activity: Next, you can configure the AI Workflow Automation activity to perform risk assessment. This involves analyzing the data against predefined criteria to determine claim approval and a generating a summary that can be used for decision making. For example, you can use following queries to assess risk for a claim:
  - Do you see any risk from the provided documents to approve this claim? Answer with 'yes' or 'no'
  - Do you see any risk from the provided documents to approve this claim? Give a detailed reason.
- Decision Making using the AI Workflow Automation activity: You can configure an AI Workflow Automation activity to use the risk assessment results to determine whether a claim can be approved/rejected immediately or require further review. If further review is required, you can add a manual task in the workflow.
- Email Notification using the AI Workflow Automation activity: Based on the decision, you can configure the AI Workflow Automation activity to generate the message to send to the claimant. For example, you can use the following query to send an email regarding claim approval:
  - If there is no risk in the claim, send an email to claimant about claim approval.
  - If there is risk in the claim, send an email to claimant about claim rejection, include the reason for the rejection.
- Claim Closure: If your claim is approved, payment processing is initiated automatically. If your claim is denied or pending further information, appropriate follow-up actions are taken to ensure resolution.

Chapter 5

Working with process activities

5.1 Configuring an activity in process template

Activities are the tasks that comprise the process template. Process template configuration involves configuring the various activities that form a process. You configure activities using the Activity properties. You can access the Activity properties by double-clicking on an activity in the workflow template editor pane.

The Activity properties dialog box has the following tabs for activity configuration:

- The **Task** tab sets the priority for automatic activities and provides instructions for manual performers. See “Setting activity definitions” on page 74.
- The **Performer** tab enables you to select the activity performer and the actions a performer can perform. For more information, see “Selecting performers” on page 68.
- The **Trigger** tab provides configuration settings that determine when an activity starts. For more information, see “Configuring activity triggers” on page 74.
- The **Notification** tab sets timers to notify the workflow supervisor if a task does not start or complete within a specific time. For more information, see “Configuring notifications” on page 75.
- The **Transitions** tab settings determine the next activities in the process template. For more information, see “Configuring activity transition rules” on page 76.
- The **Data** tab is used to select a package for the activity. For more information, see “Configuring process data within an activity” on page 79
- The **Aviator** tab is used to configure Aviator functionality for each task in a process. You can enable or disable a query space and queries within a query space for each task. For more information, see “Configuring process data within an activity” on page 79
- The **Display** tab configures the visual display settings of the process template. For more information, see “Changing display settings” on page 80.

The name of the activity you are configuring appears in the text box at the top of the activity properties dialog box. Each activity must have a unique name within the process template. You can change the activity name in the activity dialog box.

5.2 Tasks: Manual activities

5.2.1 Selecting performers

The first task when defining an activity is to identify performer for an activity. Activities can be performed manually by an individual, group, or alias that you identify, or automatically by the system. For manual tasks, you can select specific performers, alias set, or allow the process participants to choose performers. For automatic activities you must specify a user whose permissions the automatic activities takes on.

To select performers for an activity:

1. In the **Activity properties** dialog box, select the **Performer** tab.
2. To select a performer, click **Add Performer**.
3. In the **Add Performer** dialog box, select a performer or a performer group or a performer group alias from the **Select a performer who will work on a task** drop-down list. For more information, see "Configuring performers for a manual activity" on page 69.
4. Click **Add**.
5. Select one of the options from **Processed By** drop-down list to select a performer from multiple performers.
    - Individual users and group members at the same time
    - Individual users and the first group member to accept the task at the same time
    - Individual user or group user member with fewest tasks
    - Individual users and the first group member to accept the task in sequence

    Note: The options **Individual users and the first group member to accept the task in sequence** is not available for performers from **Specific Group** or **Group from Alias Set**. If you have added performers from another type apart from performers from **Specific Group** or **Group from Alias Set**, this option is available.
6. Select from the following Task Options:
    - Select **Allow user to delegate task** to enable the performer to pass the task to another user or group. When you select this option, you must also specify where the task is sent if the user to whom the performer delegates it is also unavailable. The task can be forwarded to the workflow supervisor or returned to the original performer.
    - Select **Allow performer to add additional performers to this task** to enable the performer to choose another user or group to also perform this task.

- Select **Performer sign-off is required** to specify that performer sign off is mandatory to complete this activity.

Note: The option, **Allow sequential task performers to reject to previous performer**, is not supported in Workflow Designer 21.2 release.

For details about the delegation and extension options, see "Defining activity delegation and extension" on page 19.

7. Click another tab on the **Activity Properties** dialog box to save the changes and proceed or click **OK** to save the changes and close the **Activity Properties** dialog box.

5.2.1.1 Configuring performers for a manual activity

To choose Workflow Supervisor as performer for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Workflow Supervisor**.

   For more information about **Workflow Supervisor** as task performer, see "Performer tab" on page 37.
3. Click **Add**. The Workflow Supervisor is added to the **Performers** list.

To choose Repositoy owner as performer for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Repository Owner**.

   For more information about **Repository Owner** as task performer, see "Performer tab" on page 37.
3. Click **Add**. The Repository Owner is added to the **Performers** list.

To choose Performer From Last Activity as performer for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Performer From Last Activity**.

   For more information about **Performer From Last Activity** as task performer, see "Performer tab" on page 37.
3. From the **Activity** list, select an activity.
4. From the **Select User** list, select a performer.
5. Click **Add**. The Repository Owner is added to the **Performers** list.

To choose Previous Activity Performer to Choose at Runtime as performer for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Previous Activity Performer to Choose at Runtime**.

   Note: This option is enabled only if no other performer is configured for the task. To use this option, you must delete all the existing performers for the task.

   For more information about **Previous Activity Performer to Choose at Runtime** as task performer, see "Performer tab" on page 37.
3. From the **Activity** list, select an activity.
4. From the **Select User type** list, select one of the following options:
   - **Any User**
   - **Any Group**
   - **Specific Group** > **Select a Group**.
   - **Group from Alias Set** > **Select an Alias Set** > **Select an Alias of type Group**.
5. Click **Add**. The selected performer is added to the **Performers** list.

To choose a performer from a Specific Group for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Specific Group**.

   For more information about **Specific Group** as task performer, see "Performer tab" on page 37.
3. From the **Select the user parameter that will contain the performer** search list, select a group. You can also search or filter the groups using the search field.
4. Click **Add**. The selected performer is added to the **Performers** list.

To choose a performer from a Group from Process Data for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Group from Process Data**.

   For more information about **Group from Process Data** as task performer, see "Performer tab" on page 37.
3. For the **User from Process Data** field, click Ellipsis to open **Select context data** dialog box.

4. In the **Select context 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 and go to select an attribute.
   - To select a process variable, expand the **Variables** node and go to select a variable.

   Note: The first activity that follows the Initiate activity cannot have performers mapped from the package attributes.

5. Click **Ok** to close the **Select context data** dialog box.
6. Click **Add**. The selected performer is added to the **Performers** list.

To choose a performer from a User from Process Data for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **User from Process Data**.

   For more information about **User from Process Data** as task performer, see "Performer tab" on page 37.

3. For the **User from Process Data** field, click Ellipsis to open **Select context data** dialog box.
4. In the **Select context 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 and go to select an attribute.
   - To select a process variable, expand the **Variables** node and go to select a variable.

   Note: The first activity that follows the Initiate activity cannot have performers mapped from the package attributes.

5. Click **Ok** to close the **Select context data** dialog box.
6. Click **Add**. The selected performer is added to the **Performers** list.

To choose a performer from a Group from Alias Set for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **Group from Alias Set**.

   For more information about **Group from Process Data** as task performer, see "Performer tab" on page 37.

3. For the **Select Alias Set** search list, select an alias set. If you have already configured a default alias set, it is displayed as the first entry in the search list. You can also search or filter the alias sets using the search field.
4. From the **Select Alias** search list, select an alias of group type.
5. Click **Add**. The selected alias is added to the **Performers** list with the `alias_set.alias_of_type_group` naming convention.

To choose a User from Alias Set performer for the manual activity:

1. On the **Performer** tab, click **Add Performer**.
2. In the **Add Performer** dialog box, from the **Select a performer who will work on a task** list, select **User from Alias Set**.

   For more information about **User from Alias Set** as task performer, see "Performer tab" on page 37.
3. From the **Select Alias Set** search list, select an alias set. If you have already configured a default alias set, it is displayed as the first entry in the search list. You can also search or filter the alias sets using the search field.
4. From the **Select Alias** search list, select an alias of user type.
5. Click **Add**. The selected alias is added to the **Performers** list with the `alias_set.alias_of_type_user` naming convention.

5.2.1.2 Configuring a performer for an auto activity

The performer for automatic activities must resolve to single user. This requirement limits your options for automatic activities to the following user categories:

- Workflow Supervisor (the workflow initiator by default)
- Repository Owner
- Performer from last activity

Note: If **Repository Owner** or **Performer from last activity** is selected as the performer, the user requires `SUPERUSER` privileges to install the process.

To choose the user whose security access is used for an automatic activity:

1. Open **Activity Properties** dialog box and go to the **Performer** tab, choose one of the following performers from the **Select a performer who will work on a task** list:

   - **Workflow supervisor**

     The automatic activity uses the permissions of the workflow supervisor, which by default is the user who starts the process.
   - **Repository owner**

     The automatic activity uses the permissions of the repository owner.

- **Performer from Last Activity**

  The automatic activity uses the permissions of the user who performed the previous activity in the process.

2. Select an action from the **Execute this method automatically**.

3. Specify the activity time-out interval in the **Timeout (sec)** field.

4. (Optional) Select the **Enable troubleshooting logging** option to enable logging.

5. (Optional) Under the Exception handling section, select the **Retry on timeout or exception** option if you want to rerun the activity after a timeout or an exception.

   - Specify the number of attempts in the **Maximum tries** field.
   - Specify the interval after which the activity is rerun after an exception occurs in the **Interval (min)** field.
   - Select an option from the **On failure** list to specify the action to take when an activity fails. The three options available are:
     - Continue
     - Terminate Process
     - Halt Process

6. Click another tab in the **Activity Properties** dialog box to save the changes and proceed or click **OK** to save the changes and close the **Activity Properties** dialog box.

5.2.1.3 Deleting a performer

1. Open the properties dialog box for an activity and go to the **Performer** tab.

2. From the **Performers** list, select a performer and click **Delete** corresponding to the performer to delete it.

   The performer is deleted and removed from the **Performers** list.

5.2.1.4 Reordering performers in the Performers list

1. Open the properties dialog box for an activity and go to the **Performer** tab.

2. From the **Performers** list, select a performer and drag it upwards or downwards to reorder the performer as required.

3. Click **OK**.

5.2.2 Setting activity definitions

You can use the **Task** tab in the **Activity Properties** dialog box to set the priority of an activity and to provide instructions for the performers of manual activities.

To set activity options:

1. In the **Activity Properties** dialog box, select the **Task** tab.
2. Enter a description of the activity in the **Subject** text-field.
3. Enter instructions you want to include for the performer of the activity in the **Instructions** field.
4. Select a priority level from the **Priority** drop-down list.

   The priority value designates the execution priority of an activity. For more information, see "Activity priority" on page 20.

5.2.3 Configuring activity triggers

A trigger is a signal that initiates change in the status of an activity. Use the **Trigger** tab to describe the conditions that trigger the activity and send the package to the performer's Inbox.

If the activity has more than one incoming flow, you can specify the number of the previous activities that must complete before this activity starts. For example, if an activity has three input ports, you may decide that the activity can start when two of the three activities have accepted packages.

For more information about activity triggers, see "Trigger conditions" on page 21.

To manage activity trigger:

1. In the Activity properties dialog box, select the **Trigger** tab.
2. Specify the number of incoming flows that must be completed before this activity starts.

   - To start this activity only when *all* preceding activities are complete, select the **All input flows are selected** option.
   - To start this activity when some number of its preceding activities are complete, select the **The number of input flows selected** option and enter the number of preceding activities that must be complete before the activity runs in the corresponding text-field.

   When an activity has only one input flow, these options are not different.
3. To ensure a specific event occurs before the selected activity is run, select the **And when the event signal arrives** check box and enter an event name in the adjacent text box.

The event can be a system-defined event, such as dm_checkin, or you can make up an event name, such as promoted or released. *OpenText Documentum Content Management - Server Fundamentals Guide (EDCCS250200-GGD)* provide more details about defining and queuing events using the Documentum API.

4. To enable an activity to run more than once in the same workflow, select check the **This activity can run more than once in a workflow** check box.

   By default, all the activities are defined as repeatable activities. Activities with multiple performers performing sequentially cannot be repeatable. If you use an activity multiple times in a process, you must structure the process template so that only one instance of the activity is active at any time. The server cannot start an activity if a previous activity based on the same definition is still running.

5.2.4 Configuring notifications

Workflow Designer supports two kinds of notification timers for activities:

- A pre-timer that alerts the workflow supervisor if an activity has not started within a designated number of hours after the process starts.
- A post-timer that alerts the workflow supervisor if an activity has not completed within a designated number of hours after the activity starts.

The task of checking the warning timers and sending the notices to the workflow supervisor is performed by the dm_WfmsTimer system administration job in Documentum Administrator. The dm_WfmsTimer job is installed with the system administration tool suite. It is not installed in the active state. If you intend to use warning timers in processes, make sure that your system administrator activates this job. When it is active, it runs by default once an hour.

To specify when the workflow supervisor is notified:

1. Go to the **Notification** tab.
2. To notify a supervisor when a task fails to start after a specific number of days or hours, select the **The activity does not trigger within** check box, and enter the number of days, hours, or minutes from the drop-down field.
3. To notify a supervisor when a task is incomplete after a specific number of days or hours, select the **The activity's work is not finished** check box, and enter the number of days, hours, or minutes from the drop-down field.
4. Click another tab on the Activity properties to save the changes and proceed or click **OK** to save your updates and close the Activity properties dialog box.

5.2.5 Configuring activity transition rules

Transition rules determine the next activity in the process. When an activity has multiple outgoing flows, you can configure to send the package to all the following activities, or only to some of the following activities depending on the outcome of the activity. For example, you might give a performer who reviews the design of a new form the choice of forwarding the design to the next reviewer or to send it back to the designer for revision. You set up this branching logic by creating flows from this activity to the two possible following activities, then allowing the performer to choose one path to follow.

Note: Workflow Designer does not support expression-based transition conditions and `Performer from last activity` as activity performer if the auto activity is not configured with Documentum method type as `java`.

If an activity has only one outgoing flow, there is no need to set a transition condition. The **Transition** tab is grayed out with the **Select all connected activities** option selected.

For more information about transitions, see “Understanding activity transitions” on page 22.

To define the transition action:

1. Go to the **Transition** tab.
2. Specify the action when this activity is completed using the **When this activity completes** drop-down list:
   - To send packages to all following activities connected to this one (including any reject flows), choose **Select all connected activities**.
   - To enable the performer to select the next activity, choose **Let performer select the next activities**.
   - To route packages to different activities based on a set of conditions, choose **Select next activities based on conditions**.

   Note: The **Select next activities based on conditions** option is available for manual activity and auto activity. The following table lists the functions available for defining transition condition:

Function	Description
Add Condition +	Adds an ElseIf construct to the transition condition.
Edit Condition ✎	Edits the transition condition.
Reorder Condition ☰	Reorders the ElseIf constructs in the transition condition.

If you select the **Select next activities based on conditions** option, skip to step 4 in this procedure.

3. If the activity is performed by multiple performers that is, if the performer category is **All users in group** or **Some users in a group**, specify the number of performers that must complete the task:

   - If you require all the performers to complete the task, select the **All performers complete the task** option.
   - If you require certain number of performers to complete the task, select the **[ ] performers complete the task** option and enter the required number of performers in the adjacent text. If the number you enter is greater than the number of performers who receive work items for this activity at run time, the server completes the activity when all performers complete the task.

4. Specify the conditions that the server uses to determine the activities that receive the packages.

   See "Creating transition conditions" on page 77 for information about creating transition conditions.

5. Click another tab on the **Activity Properties** dialog box to save the changes and proceed or click **OK** to save your updates.

5.2.5.1 Creating transition conditions

When you choose the **Select next activities based on conditions** option, a table appears showing the defined transition conditions. Follow this procedure to add transition conditions for automatically choosing the next activity in the process. For more information about transition conditions, see "Understanding activity transitions" on page 22.

To create a transition condition:

1. Click **Tansitions** tab to create a transition.

2. From the **When this activity completes** list, choose **Select next activities based on the conditions**. A table with If-Else transition condition is displayed. By default, a transition condition contains a single If-Else statement. You can also configure this statement to add multiple conditions using multiple ElseIf constructs.

3. To configure the If-Else transition condition, complete these steps:

   a. Position the mouse over a row in the transition condition table and click

      Edit button to open the **Edit Rule** dialog box. The **Edit Rule** dialog box enables you to define a conditional construct and select a corresponding task for each condition.

   b. To define a condition, complete one of the following steps:

- Enter the condition in the **If condition** text field.
- Click Ellipsis to open the **Select context data** dialog box.

  i. Select a package, process variable, process instance, or a task from the **Context Data** tab or select a predefined function from the **Functions** tab, and then click **OK**.

     Notes

     – Starting with 24.2 release, process variables are available under the **Context Data** tab for defining a transition condition.

     – While creating or editing an If condition, you can use only the pre-defined functions that are listed under the Functions tab. If a user-defined function is used, the following error message is displayed.

     ```
     Expression is invalid. Unsupported function name ...
     ```

     While creating or editing an If condition, you can use only from the following pre-defined Process Instance and Task available under Context Data:

     – Process Instance
       - Created By
       - Created On
       - Process ID
       - Process Instance ID
       - Process Instance Name
       - Process Name

     – Task
       - Activity ID
       - Activity Name
       - Created On
       - Performer
       - Priority
       - Task ID
       - Task Name
       - Task State

  ii. In the **Edit Rule** dialog box, select a task under **Tasks** section, and then click **OK**.

4. When all of the specific transition conditions are defined, select the activities to which packages are routed if none of the conditions are met.

    a. Select the **else** row and click Edit.

    b. In the **Edit Rule** dialog box, select an attribute from the **Task** list.

    c. Click **Ok**.

5. Change the order of the conditions if necessary.

    The server evaluates transition conditions in the order they appear in the table, and routes the packages based on the first condition that evaluates to TRUE. To change the position of a condition in the table, use the Ellipsis button for a condition and drag it to required position.

5.2.6 Configuring process data within an activity

1. go to the **Data** tab.

    - To select a package, expand the **Packages** node and select the package.
    - To select a process variable, expand the **Variables** node and select a process variable.

2. Configure the packages 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 properties.
Visible at this activity	Select this option to make the package available to the performer of this activity. Clear this option if you do not want the performer of this activity to see the package at run time.
This is a mandatory package	Select to require filling the package before the activity completes.

3. Configure the process variables as described in the following table:

Field	Description
Visible at this activity	Select this option to make the 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 process variable at run time.

4. Click **OK**.

5.2.7 Configuring Content Aviator for an activity

All the query spaces created for a process are available to the activities. By default, all the query spaces are disabled. You can enable or disable a query spaces and the containing queries for each activity in a process. Complete the following steps to configure query spaces and containing queries for an activity:

1. Go to the **Aviator** tab.
2. From the **Query Space** list, select a query space.
3. Use the **Enable Query Space** switch to enable or disable the selected query space.
4. Under the **Predefined query set** section, click Ellipsis corresponding to a query to open the pop-up menu and select the **Enable** or **Disable** option based on the current state of the query.
5. Click **OK**.

5.2.8 Changing display settings

The options on the **Display** tab control how the activity appears in the visual display of the process template.

To change the display settings for an activity:

1. In the **Activity Properties** dialog box, select the **Display** tab.
2. Set the font type and font size to use for activity label in the process template.
   a. Select a font from the **Font** list.
   b. Select a point size from the **Point Size** list.

5.2.9 Auto Sync

The auto sync feature checks the repository for updates in custom object types, Alias sets, and Groups that are configured in processes template as packages and performers. If a custom object type, Alias set, or a Group is updated or deleted using other clients such as Documentum Administrator or Documentum Composer, the Auto Sync feature synchronizes the corresponding processes template for the impact and reports either an error or warning for the inconsistency in the Workflow Designer.

Auto Sync runs as a Documentum job named, `WebdesignerSyncJob`, in Documentum Administrator. Each process template in Workflow Designer has atleast one object type, Alias set, or a Group from the repository. If they are changed in the repository, two new icons are displayed for the impacted template in the navigation area of the Workflow Designer. For more information about these icons, see the “Process list” on page 27 section. You can configure the frequency of auto sync job, by default the auto sync job is run once every day.

Note: We recommend you to configure the auto sync job to run every hour or 10 minutes in development and testing phase.

Auto sync feature raises a flag for the following events in the repository:

- Delete custom type
- Delete aliases
- Delete Alias set
- Delete group
- Delete attribute of custom type

Notes

- Add Documentum repository owner user to `documentum_workflow_designer` role from Documentum Administrator for auto sync to run seamlessly.
- If a group is deleted and then another group is created with the same name as the deleted group, Auto Sync flags an error.
- If an object type, alias set, or a group is deleted from the repository and added again in the repository, you must manually re-adopt these instance:
  - Select the object type again as type for the package.
  - Select the alias set as the Default Alias Set or configure in the **Performer** settings as applicable.
  - Select the group again in the **Performer** settings.
- Any change to a type may take 2-3 minutes to reflect in the repository.

5.3 Tasks: System activities

5.3.1 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**. **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.
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.

Field	Description
Persist process data	Save the process variables that are successfully populated before an error occurs in the activity. **Note:** This option is available only for **Data Extraction** activity.
On failure	Select how to proceed when the exception cannot be resolved.

5. Click **OK**.

5.3.2 Configuring the AI Workflow Automation activity

1. Drag and drop the **AI Workflow Automation** activity onto the canvas.
2. Double-click the activity to open the properties dialog box.
3. On the **General** tab, click **No Mappings** to open **Output message mapping** dialog box. The **Output message mapping** dialog box lists the query spaces and queries under the **Source** section and process variables under **Destination** section. You can map the values retrieved by queries to the process variables using the various mapping functions available under the **Function** mapping area.
4. Click the plus sign in the **Function mapping** area and select a mapping function from the list.
5. Select a query from a query space listed under the **Source** area and connect it to the mapping function.
6. Select a process variable from the **Answers** area and connect it to the mapping function.
7. Map all the required queries to the process data and click **Save**.
8. Click **Close**.
9. Click the **Execution** tab and configure the properties as described in the following table:

Field	Description
Execution Settings	Execution settings: • **Perform this task as**: Indicates task performer role. Available options are: – **Workflow Supervisor** – **Repository Owner** – **Performer from Last Activity** • **Timeout (sec)**: Time out in seconds after which operation is aborted. • **Enable Troubleshooting Logging**: Indicates whether troubleshooting logging is enabled or not.
Exception Handling	Process to handle exceptions. • **Retry on timeout or exception**: Indicates whether to retry an operation on timeout or exception. – **Maximum tries**: Maximum number of attempts in case of a failure due to a timeout or an exception. – **Interval (min)**: Time interval after which an attempt is initiated after a failure. • **Persist process data**: Save the process variables that are successfully populated before an error occurs in the activity. • **On failure**: Action to perform in case of failure. – **Continue**: Ignore the failure and continue with the process. – **Terminate Process**: Terminate the process on failure. – **Halt Process**: Stop the process on failure.

10. Click the **Trigger** tab and configure the properties as described in the following table:

Parameter	Description
Trigger	• **All input flows are selected**: Activity is triggered when all the input flows are selected. • **The number of input flows selected**: Minimum number of input flows to trigger the activity. • **And when the event signal arrives**: Event to trigger the activity.
Frequency	**The activity can run more than once in a workflow**: Indicates whether the activity can run multiple times in a workflow.

The following table describes the parameters in the **Notifications** tab:

Parameter	Description
Notify the workflow's supervisor when	
The activity does not trigger within	Time limit within which an activity must start.
The activity work is not finished	Time limit within which an activity must complete.

11. Click the **Transition** tab and configure the properties as described in the following table:

Parameter	Description
When this activity completes	Logic to determine the next activity. You can select from the following options: • **Select all connected activities**: Select all the activities connected to the current activity. • **Select next activities based on conditions**: Select the next activity based on the specified condition.
If the **Select all connected activities** option is selected:	Complete the activity when: • **All performers complete the task**: All the assigned performers complete this activity. • **Performers complete the task**: Minimum number of performers who must complete this activity.
If the **Select next activities based on conditions** option is selected:	Configure the If-else condition construct. For more information about how to configure if-else condition construct, see "Creating transition conditions" on page 77.

12. Click the **Data** > **Packages** tab to view and configure the package properties as described in the following table:

Parameter	Description
Name	Read-only. Name of the package.
Type	Read-only. Type of the package.
Version	Version of the package.
Visible at this activity	Whether the package is visible for this activity.
This is a mandatory package	Whether the package is mandatory for this activity.

13. Click the **Data** > **Variables** tab to view and configure the variable properties:

Parameter	Description
Name	Read-only. Variable name.
Type	Read-only. Only String type variable is supported.
Mulit value	Read-only. Whether the variable can store single value or multiple values.
Default Value	Read-only. Default value of the variable.
Visible at this activity	Indicates whether the variable is visible for this activity.

Note: To hide a process variable for an activity, deselect **Visible at this activity** option for the specific activity.

14. Click the **Display** tab and configure the label styling properties for the activity:

Parameter	Description
Font	Font type to use for label.
Point Size	Size of the font.

15. Click **OK** to complete the AI Workflow Automation activity configuration.

Chapter 6

Defining and managing flows

The flow lines that connect the activities represent the flow of the document in a process. Flows enable the movement of packages, their properties, and dependencies between the connected activities. See "Process template" on page 12 for a description of flows. After you have added a flow to the process template, you configure it using the Flow Inspector. To open the Flow Inspector dialog box, double-click the flow.

The Flow Inspector dialog box provides fields to control how the flow appears in the visual display of the process template. For more information, see "Configuring flows" on page 88.

The name of the flow you are configuring appears in the text box at the top of the Flow Inspector. If more than one flow is selected, arrow buttons appear on either side of the text box, enabling you to scroll through the selected flows. The settings you make apply to the flow whose name appears in the box, unless you select the **Apply to all selected** option.

When multiple flows are selected, each tab in the Flow Inspector displays one or more check boxes labeled **Apply to all selected**. When you select this option, Workflow Designer applies the associated settings that is, those settings that appear to the right of the check box to *all* selected flows, not just the one whose name appears in the text box at the top. For example, you can select multiple flow and choose the same packages for all of them at once. Any settings for which the check box is not selected apply only to the current flow.

6.1 Creating flows

You can connect activities using one of the three Create Flow icons in the Workflow Designer toolbar:

- To connect activities in a forward movement of data, click either the **Single flow** or the **Multisegment flow** icon. The difference between the two is visual: one draws a straight line to represent the flow between activities, the other draws a line consisting of multiple segments.
- To connect activities in a backward movement of data, click **Reject** flow. Reject flows represent the path taken when the user of an activity rejects the object being processed.
- To handle faults in an activity, click **Fault Flow**. The fault flow represents the alternate path that is taken when an activity fails, redirecting the flow to a manual or automatic activity. After the task is completed, the flow resumes from the next activity after the activity.

See "Process template" on page 12 for a description of the types of flows.

6.2 Configuring flows

You can use the Flow inspector to configure how a flow appears in the process template.

To change the display settings for a flow:

1. Double-click a flow to open the **Flow Inspector**.
2. Set the font and font size specification for the package names routed over the flow.
   a. Select a font from the **Font** list.
   b. Select a font size from the **Point Size** drop-down list.

   Note: These settings are relevant only if you select to display the package names or custom information in the next step.
3. Select the **Show label** check box if you want to display the package name or custom information for a flow.
   - Select the **Show visible packages at destination activity** option to display the name of the package that are routed across the flow.
   - Select the **Custom label** to display a custom information that is specified in the corresponding field.
4. Click **Ok** to save the changes and close the **Flow inspector** dialog box.

Chapter 7

Working with Data Mapping

7.1 Tasks

7.1.1 Configuring a data mapping function

1. Go to the **Output message mapping** dialog box.
2. Click the plus (**+**) icon for **Function mapping** and select a function from the list. The selected function is added to the function mapping canvas.
3. Double-click the function to open the **Mapping details** dialog box.
   - In the **Mapping details** dialog box, click **Source/Function parameters** plus (**+**) to add a constant as an additional input value. Type the constant value in the field. You can edit or delete an input constant:
   - To remove a constant, select the constant and click **Delete**.
   - To edit a constant, click **Edit** and type a new value.
   - To change the order of the attributes, use the arrows to move them up or down in the list.
4. Configure the mapping function as described in the following table:

Field	Description
Function	Read-only field, specifies the function type.
Data type returned	Read-only field, specifies the return data type
Syntax	Read-only field, specifies the syntax of the function.
Destination attribute	Read-only field, specifies the destination attribute.
Input repeating context	Specifies how the function processes the input attributes.
Output repeating context	Specifies how the function processes the output value.

Field	Description
Source/Function parameters	Lists the input parameters for the function. To change the order of the attributes, use the arrows to move them up or down in the list. You can configure each parameter as: • **Mandatory**: Indicates that the variable is mandatory and the activity fails if the value is empty. Setting a parameter as **Mandatory** ensures that all the critical data is captured before moving to the next step. • **Default**: Select this option to indicate the activity to use default value if no value is set. You can configure the default value in the corresponding field.

5. Click **Save**.

7.1.2 Mapping data from one to many nodes

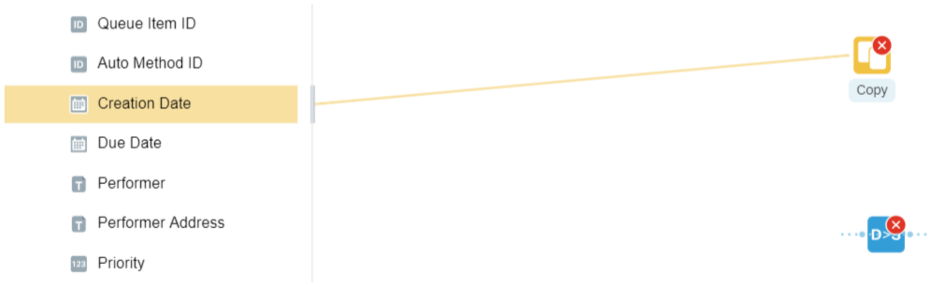
You can configure a variable as input to multiple functions whose output then sets values for multiple destination variables. You can configure this mapping with a single data mapping.

For example, a date-time variable could be the source for both of the following functions:

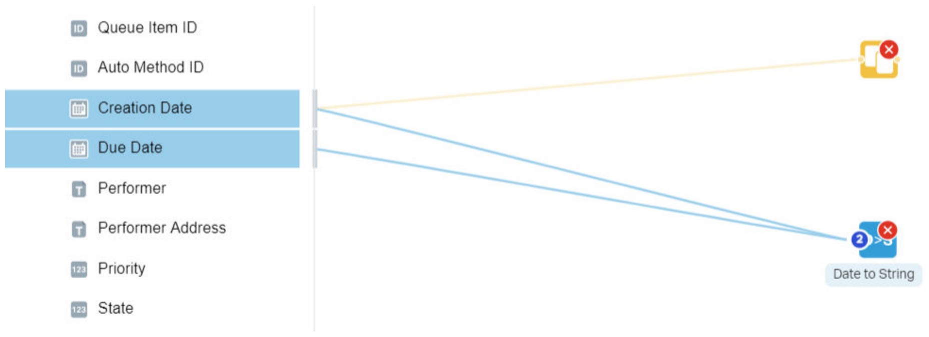
- A copy function that sets the value of a date-time destination variable
- A date-to-string function that sets the value of a string variable

To map data from one source to many destinations:

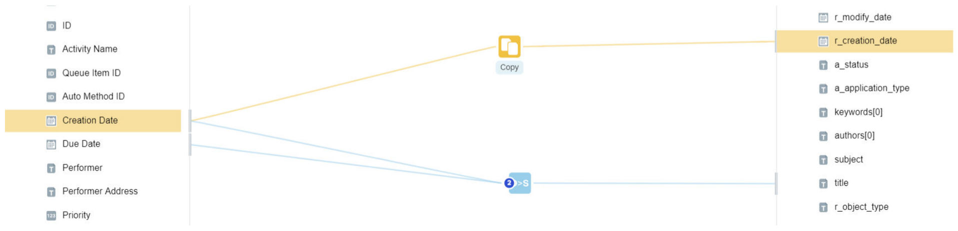
1. Add a **Copy** function and a date-to-string function to the canvas.
2. Select an input variable and drag it to the **Copy** function to create a connection.



3. Select the input variable again and drag it to the **date-to-string** function to create a connection.



4. Connect the output for the two functions to required output variables.



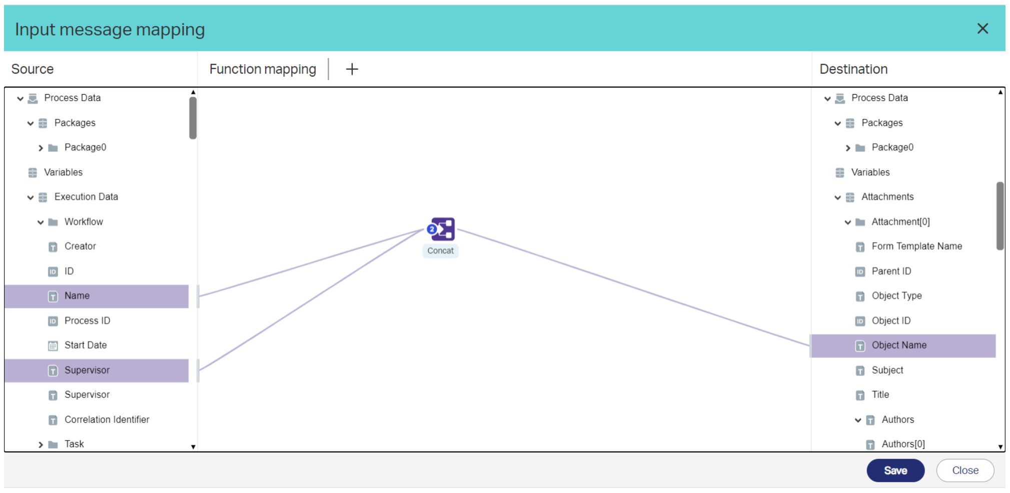
5. Click **Save**.

7.2 Concepts

7.2.1 Process data mapping

In the **Input message mapping** tool, the **Function mapping** area provides a graphical mapping between input and output attributes. At run time, the activity passes the values of the input attributes 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:



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 or if the function does not have all of its required arguments, an Info is displayed on the function indicating the error.

You can access and configure the properties of a function using the **Mapping details** dialog box. It displays the name of the function, return data type, syntax, destination attribute, and a list of the input values. You can configure the following properties of a function:

- 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 **Mapping details** dialog box to add the constant input values.
- Set input and output repeating context: When the input argument for a function is a multi-value 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 **Mapping details** dialog box.

7.2.2 Mapping functions

The source and destination areas are nodes of the process data available to the integration activity where the mapping takes place.

You can edit and configure a function using the **Mapping details** dialog box. It 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
Absolute	Number	Returns an absolute non-negative value corresponding to the number passed as an input argument.
Add	Two or more numbers.	Sum of the input arguments.

Function	Input arguments	Result
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.
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 multivalue input. For single-value inputs, the count 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.

Function	Input arguments	Result
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.
Get Ticket	String	Generates a login ticket to the given user name at run time. Uses the following syntax: user name, 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 lower case letters.
ToUpper	String	Converts the string to upper case letters.

7.2.3 Using multi-value attributes as input to functions

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 configure the input context options in the **Mapping details** dialog box for a function.

Chapter 8

Building expressions

8.1 Tasks

8.1.1 Building an expression using the selector

The selector restricts the data available to build an expression to the context you select.

To build an expression using the selector:

1. Click **Select**.
2. To use context data in the expression:
   a. Click **Context Data**.
   b. Select the data context. For example, select a business object.
   c. Select a data item for the context. For example, select a business object attribute.
   d. Click **OK**.
3. To use a function in the expression:
   a. Click **Functions**.
   b. Select a function category. For example, select **String**.
   c. Select a function.

8.1.2 Migrating expressions

While migrating a process template created through Process Builder and Documentum Workflow Manager to the latest Workflow Designer, only the expressions mapped to a package are supported.

8.2 Concepts

8.2.1 Expressions

You can build expressions that the system evaluates at run time to compare and set the process data. An expression can include hard-coded literals and data from the context in which you build the expression. The expression evaluates to a type that is consistent with how you are using the expression. For example, use a Boolean expression to return a value for a condition.

The following table describes a situation where you can build expressions:

Situation	Description
Defining a condition for selecting next activity	Returns a boolean value that will be used to determine the next activity to run.

Consider these best practices when you build an expression:

- If a long expression wraps to the next line in the expression editor, the expression editor might indicate an error by underlining the part of the expression in red even if the expression is valid. To clear the line, use the divider next to the Artifact navigator to expand the width of the expression editor so the expression does not wrap to the next line.
- Use curly brackets to include a multivalue attribute in the expression. For example:

  {1,2,3,4,5}

  You can use operators, functions, and context data inside the brackets.
- To include a particular element of a multivalue attribute in an expression, use square brackets to extract the element from the attribute. In the expression `customer.keywords[0] = = 'VIP'`, customer is the business object, keywords is an attribute of the business object, and [0] is the element. The system treats the attribute and element as a single value when evaluating the expression.
- Use single quotation marks for string literals. For example:

  ```
  'dog'
  ```
- Use parentheses to group data and set precedence. For example:

  ```
  (5 + 3) * 4
  ```
- While comparing float values in an expression, use the round(float value, precision) function to round off float values to a precise decimal point.

8.2.2 Operators

The expression editor supports the operators listed in the following table:

Type	Operator
Math	+
	-
	/
	*
Comparison	>
	>=
	<
	<=
	==
	!=
Boolean logic	AND
	OR
	NOT

You can use the + operator to concatenate string, float, and integer data types in expressions. The expression editor concatenates data types as follows:

- string + string = string
- float + float = float
- integer + integer = integer
- string + integer = string
- string + float = string
- float + integer = float

8.2.3 Functions

Workflow Designer has built-in functions that you can use in expressions. Java-based functions are evaluated in a server context.

The related topics define each of the functions available through Workflow Designer and indicate in which context they are available.

Functions can contain variable numbers of arguments.

8.2.4 Date functions

The date functions execute in the server context using the time zones of the Java Virtual Machine (JVM) in which they run. The system evaluates expressions in the context of the server or the browser. You should set your application servers to the same time zone as each other and the client machines. If your end users are geographically distributed, use the UTC-based functions when creating dates.

Function	**Description**	**Example**	**Runs in the server side context**
addDays(DateTime <*date*>, Integer <*days*>) : DateTime	Returns the DateTime produced after adding the number of days specified by <*days*> to <*date*>. The argument <*days*> can be a negative number which indicates subtraction.	addDays(now( ), -10) returns the DateTime representation of January 20, 2012 11:56:57 PM	Yes
addHours(DateTime <*date*>, Integer <*hours*>) : DateTime	Returns the DateTime produced after adding the number of hours specified by <*hours*> to <*date*>. The argument <*hours*> can be a negative number which indicates subtraction.	addHours(now(), 3) returns the DateTime representation of January 31, 2012 2:56:57 AM	Yes
addMinutes(DateTime <*date*>, Integer <*minutes*>) : DateTime	Returns the DateTime produced after adding the number of minutes specified by <*minutes*> to <*date*>. The argument <*minutes*> can be a negative number which indicates subtraction.	addMinutes(now(), 30) returns the DateTime representation of January 31, 2012 12:32:23 AM	Yes
addSeconds(DateTime <*date*>, Integer <*seconds*>) : DateTime	Returns the DateTime produced after adding the number of seconds specified by <*seconds*> to <*date*>. The argument <*seconds*> can be a negative number which indicates subtraction.	addSeconds(now(), 22) returns the DateTime representation of January 31, 2012 12:03:40 AM	Yes
createDate(Integer <*year*>, Integer <*month*>, Integer <*day*>) : DateTime	Returns the DateTime representation specified by <*year*>, <*month*>, <*day*>.	createDate(2012,1,31) returns the DateTime representation of January 31, 2012 with the time set to the current time	Yes

Function	Description	Example	Runs in the server side context
createDateTime(Integer <*year*>, Integer <*month*>, Integer <*day*>, Integer <*hour*>, Integer <*minute*>, Integer <*seconds*>) : DateTime	Returns the DateTime representation specified by <*year*>, <*month*>, <*day*>, <*hour*>, <*minute*>, and <*seconds*>. The system returns a date based on the timezone of the Java Virtual Machine (JVM) evaluating the expression. Type the hour parameter in 24-hour format. **Note:** The time is set using the timezone of the Java Virtual Machine (JVM) doing the evaluation. In the context of an application, the JVM is either on the application server where the application is running, or on the Java Method Server. It is recommended to have both of these JVMs on the same timezone and in line with the timezone of the end users. If your end users are geographically distributed, it may be better to use the createDateUTC function.	createDateTime(2012, 1,31,11,13,45) returns the DateTime representation of January 31, 2012 11:13:45 AM	Yes
createDateUTC(Integer <*year*>, Integer <*month*>, Integer <*day*>, Integer <*hour*>, Integer <*minute*>, Integer <*secs*>) : DateTime	Returns the DateTime representation specified by <*year*>, <*month*>, <*day*>, <*hour*>, <*minute*>, and <*secs*> using Universal Time, Coordinated (UTC). Type the <*hour*> parameter in 24-hour format. **Note:** This function was introduced to avoid the problems associated with creating a date without knowing the timezone in which it is being created.	createDateUTC(2012, 1,31,11,13,45) returns the DateTime representation of January 31, 2012 11:13:45 AM in UTC	Yes

Function	Description	Example	Runs in the server side context
differenceDays (DateTime <*date1*>, DateTime <*date2*>) : Integer	An integer representing the number of days between <*date1*> and <*date2*>. This operation is like subtraction between two dates. If <*date2*> is later than <*date1*>, the result will be negative. The result is converted to days and rounded to the nearest whole number.	January 31, 2012 11:13:45 AM - February 1, 2012 7:28:29 PM differenceDays(createDate(2012,1,31,11,13,45),createDate(2012,2,1,19,28,29)) returns -1 February 1, 2012 7:28:29 PM - January 31, 2012 11:13:45 AM differenceDays(createDate(2012,2,1,19,28,29),createDate(2012,1,31,11,13,45)) returns 1 January 31, 2012 12:00:00 AM - January 31 2012 11:59:59 PM differenceDays(createDate(2012,1,31,0,00,00),createDate(2012,1,31,23,59,59)) returns 0	Yes

Function	Description	Example	Runs in the server side context
differenceSeconds (DateTime <*date1*>, DateTime <*date2*>) : Integer	Returns an integer representing the number of seconds between <*date1*> and <*date2*>. The operation is like subtraction between dates. If <*date2*> is later than <*date1*>, the result will be negative. The result is converted to seconds and rounded to the nearest whole number. **Note:** The Integer data type can hold a minimum value of -2147483648 and a maximum value of 2147483647, so it is possible for an error to occur if the two dates used are more than 68 years apart.	January 31, 2012 11:13:45 AM - February 1, 2012 9:28:29 AM differenceSeconds(createDate(2012,1,31,11,13,45),createDate(2012,2,1,9,28,29)) returns -80084 February 1, 2012 9:28:29 AM - January 31, 2012 11:13:45 AM differenceSeconds(createDate(2012,2,1,9,28,29),createDate(2012,1,31,11,13,45)) returns 80084 January 31, 2012 12:00:00 AM - January 31 2012 11:59:59 PM differenceSeconds(createDate(2012,1,31,0,00,00),createDate(2012,1,31,23,59,59)) returns -86399	Yes
getDay(DateTime <*date*>) : Integer	Returns the integer value of the day of the month (1-31) used in <*date*>.	employee.hire_date = January 30, 2012 8:05:23 AM (or rather the DateTime representation of January 30, 2012) getDay(employee.hire_date) returns 30	Yes

Function	Description	Example	Runs in the server side context
getHour(DateTime <*date*>) : Integer	Returns the integer value of the hour in 12 hour notation (1–12) used in <*date*>.	employee.hire_date = January 30, 2012 8:05:23 AM (or rather the DateTime representation of January 30, 2012) getHour(employee.hire_date) returns 8	Yes
getMinute(DateTime <*date*>): Integer	Returns the integer value of the minute (0-59) used in <*date*>.	employee.hire_date = January 30, 2012 8:05:23 AM (or rather the DateTime representation of January 30, 2012) getMinute(employee.hire_date) returns 5	Yes
getMonth(DateTime <*date*>): Integer	Returns the integer value of the month (1–12) used in <*date*>	employee.hire_date = January 30, 2012 8:05:23 AM (or rather the DateTime representation of January 30, 2012) getMonth(employee.hire_date) returns 1	Yes
getUserTimeZone(): String	Returns the user time zone.	getUserTimeZone() returns GMT	No
getYear(DateTime <*date*>): Integer	Returns the integer value of the year used in <*date*>.	employee.hire_date = January 30, 2012 8:05:23 AM (or rather the DateTime representation of January 30, 2012) getYear(employee.hire_date) returns 2012	Yes
now() : DateTime	Returns the DateTime representation of the current date and current time.	now() returns the DateTime representation of January 31, 2012 12:03:18 AM	Yes

Function	Description	Example	Runs in the server side context
stringToDate(String <*dateValue*>, String <*format*>) : DateTime	Returns the DateTime representation specified by <*dateValue*> which is encoded in the date format specified by <*format*>. This function only supports date text in the locale of the Java Virtual Machine (JVM). The <*format*> can be specified using the same rules as the Java SimpleDateFormat class. See the standard JavaDocs available from Oracle for examples. The timezone is based on the timezone of the JVM doing the evaluation.	stringToDate('01/31/2012','MM/dd/yyyy') returns the DateTime representation of January 31, 2012 12:00:00 AM (with the timezone of the JVM)	Yes
stringToDate(String <*dateValue*>) : DateTime	Returns the DateTime representation specified by <*dateValue*> encoded in the <*c ISO 8601*> date format. **Note:** If unspecified, the month / day defaults to the current month / day, the time defaults to midnight, while the timezone defaults to the browser's timezone. If a time is specified, it must include both hours and minutes. The "T"delimiter, seconds, milliseconds and timezone are optional. The decimal fraction of a second, if specified, must contain at least 1 digit (there is no limit to the maximum number of digits allowed), and may be delimited by either a '.' or a ',' .	stringToDate('08/19/2012' 16:29 01:00'c') returns the DateTime representation as 2012-08-19T16:20+01:00.	Yes

Function	Description	Example	Runs in the server side context
stringToDateUTC(String <*dateValue*>, String <*format*>) : DateTime	Returns the DateTime representation specified by <*dateValue*> which is encoded in the date format specified by <*format*>, using Universal Time, Coordinated (UTC). The <*format*> can be specified using the same rules as the Java SimpleDateFormat class. See the standard JavaDocs available from Oracle for examples.	stringToDateUTC('01/31/2012','MM/dd/yyyy') returns the DateTime representation of January 31, 2012 12:00:00 AM (with the timezone of UTC).	Yes
todayUTC() : DateTime	Returns the DateTime representation of the current date with the time portion set to midnight (00:00:00) using Universal Time, Coordinated (UTC) as the timezone.	Assume it is January 31, 2012 todayUTC( ) returns the DateTime representation of January 31, 2012 12:00:00 AM (with the timezone of UTC)	Yes
isValidDateTime (Date <*widget ID*>)	Validates whether the date and time value entered in the Date-time input widget is valid.	Assume it is 03/15/2014 11:00:00 AM isValidDateTime () validates the DateTime representation of 3/15/2014 11:00:00 AM and returns the True value for the correct format.	Yes

Function	Description	Example	Runs in the server side context

toLocaleDateString (DateTime date, String locale) : String	Returns the localized string representation of the date.	var date = new Date(Date.UTC(2012, 11, 20, 3, 0, 0));
		
		// British English uses date in the day-month-year format
		
		toLocaleDateString(date,'en-GB') returns “20/12/2012”
		
		// US English uses date in the month-day-year format
		
		toLocaleDateString(date, 'en-US') returns “12/20/2012”
			No

toLocaleTimeString (DateTime date, String locale) : String	Returns the localized string representation of the time.	var date = new Date(Date.UTC(2012, 11, 20, 3, 0, 0));
		
		// British English uses 24-hour time without AM/PM
		
		toLocaleTimeString(date,'en-GB') returns “03:00:00”
		// US English uses 12-hour time with AM/PM
		
		toLocaleTimeString(date, 'en-US') returns “7:00:00 PM”
			No

Function	Description	Example	Runs in the server side context
toLocaleString (DateTime date, String locale) : String	Returns the localized string representation of the date and time.	var date = new Date(Date.UTC(2012, 11, 20, 3, 0, 0)); toLocaleString(date,'en-GB') returns “20/12/2012 03:00:00” toLocaleString(date, 'en-US') returns “12/19/2012, 7:00:00 PM”	No

8.2.5 List functions

A list (represented by the square brace notation [ ] in the data type) is an array-like representation of repeating attributes. In other words, any repeating attribute can be used where a list would be accepted. In addition, the list could be passed in with the curly brace notation { } as in the examples. The first index position is 0, not 1. The use of Any in the function signature and return values means that any data type can be passed in (string, boolean, datetime, float, integer), but the input parameters must be of the same type or an error is returned at run time.

The return value will be of the same type as the input parameters.

Function	Description	Example	Runs in the server side context

add(Any[ ] <*list1*>, Any[ ] <*list2*>, Integer index) : Any[ ]	The list produced by inserting <*list2*> into <*list1*> after the position specified by <*index*>.	add({ 'Peter', 'Paul'}, {'Mary', 'Jane'}, 1)
		
		 returns {'Peter', 'Mary', 'Jane', 'Paul'}
			Yes

add(Any[ ] <*list*>, Any <*newValue*>, <*Integer index*>) : Any [ ]	The list produced by inserting <*newValue*> into <*list1*> after the position specified by <*index*>	add({ 'Peter', 'Paul'}, 'Mary', 1)
		
		 returns {'Peter', 'Mary', 'Paul'}
			Yes

Function	Description	Example	Runs in the server side conte xt

add(Any[ ] *<list1>*, Any[ ] *<list2>*) : Any[ ]	The list produced by appending *<list2>* to the end of *<list1>*.	add({ 'Peter', 'Paul'}, {'Mary', 'Jane'})
		
		returns {'Peter', 'Paul', 'Mary', 'Jane'}
			Yes

add(Any[ ] *<list>*, Any *<t>*) : Any[ ]	The list produced by appending *<t>* to the end of *<list1>*.	add({ 'Peter', 'Paul'}, 'Mary')
		
		returns {'Peter', 'Paul', 'Mary'}
			Yes

compare(Any[ ] *<list1>*, Any[ ] *<list2>*, *<Boolean strict>*) : Boolean	Compares two lists and returns TRUE if *<list1>* and *<list2>* have the same values in the list. If *<strict>* is set to TRUE, then the order of the elements is considered; if *<strict>* is set to FALSE then the order of the elements in not considered.	compare({ 'Peter', 'Mary', 'Paul'}, {'Paul', 'Mary', 'Peter'}, FALSE) returns TRUE and compare({ 'Peter', 'Mary', 'Paul'}, {'Paul', 'Mary', 'Peter'}, TRUE) returns FALSE	Yes
contains(Any[ ] *<list>*, Any *<object>*) : Integer	Compares *<object>* to each item in *<list>* and returns the index position of the first match. If no match is found, -1 is returned.	contains({ 'Peter', 'Mary', 'Paul'}, 'Paul') returns 2 and contains({ 'Peter', 'Mary', 'Paul'}, 'Jane') This example returns -1	Yes

remove(Any [ ] *<list>*<>, Integer *<startIndex>*, Integer *<endIndex>*) : Any [ ]	Removes items from *<list>* beginning with the position *<startIndex>* and ending with the position *<endIndex>* , inclusively. If *<startIndex>* is the same as *<endIndex>* then only one item will be removed (the one specified by *<startIndex/endIndex>*).	remove({ 'Peter', 'Paul', 'Mary', 'Jane'}, 0, 1})
		
		This example returns {'Mary', 'Jane'}
			Yes

Function	Description	Example	Runs in the server side conte xt
size(Any[ ] <*list*>) : Integer	The number of items in <*list*> as an integer.	size({ 'Peter', 'Paul', 'Mary'}) This example returns 3	Yes
allEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if all the values in the list are equal to <*value*> and otherwise returns FALSE. The allEquals function returns a Boolean result.	allEquals (Arrays.asList(new String[]{“pen”, ”pen”}), “pen”) This example returns TRUE.	Yes
allGreaterThan(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if all the values in the list are greater than <*value*> and otherwise returns FALSE. The allEquals function returns a Boolean result.	allGreaterThan(Array s.asList(new Float[] {3.2F, 3.2F}), 1.1F) This example returns TRUE.	Yes
allGreaterThanEqual s(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if all the values in the list are greater than or equal to <*value*> and otherwise returns FALSE. The allGreaterThanEquals function returns a Boolean result.	allGreaterThanEqual s(Arrays.asList(new Float[]{3.2F, 3.2F}), 3.3F) This example returns FALSE.	Yes
allLesserThan(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if all the values in the list are lesser than <*value*> and otherwise returns FALSE. The allLesserThan function returns a Boolean result.	allLesserThan(Arrays .asList(new Float[] {3.2F, 3.2F}), 3.3F) This example returns TRUE.	Yes
allLesserThanEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if all the values in the list are lesser than or equal to <*value*> and otherwise returns FALSE. The allLesserThanEquals function returns a Boolean result.	allLesserThanEquals (Arrays.asList(new Float[]{3.2F, 3.2F}), 3.3F) This example returns TRUE.	Yes

Function	Description	Example	Runs in the server side context
allNotEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if none of the values in the list is equal to the <*value*> and otherwise returns FALSE. The allNotEquals function returns a Boolean result.	allNotEquals (Arrays.asList(new Integer[]{1, 1}), 2) This example returns TRUE.	Yes
anyGreaterThan(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if any value in the list is greater than <*value*> and otherwise returns FALSE. The anyGreaterThan function returns a Boolean result.	anyGreaterThan (Arrays.asList(new Float[]{3.2F, 3.2F}), 1.1F) This example returns FALSE.	Yes
anyGreaterThanEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if any value in the list is greater than or equal to <*value*> and otherwise returns FALSE. The anyGreaterThanEquals function returns a Boolean result.	anyGreaterThanEquals (Arrays.asList(new Float[]{3.2F, 3.2F}), 3.2F) This example returns TRUE.	Yes
anyLesserThan(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if any value in the list is lesser than <*value*> and otherwise returns FALSE. The anyLesserThan function returns a Boolean result.	anyLesserThan(Arrays.asList(new Float[] {3.2F, 3.2F}), 3.1F) This example returns FALSE.	Yes
anyLesserThanEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if any value in the list is lesser than or equal to <*value*> and otherwise returns FALSE. The anyLesserThanEquals function returns a Boolean result.	anyLesserThanEquals(Arrays.asList(new Float[]{3.2F, 3.2F}), 3.3F) This example returns TRUE.	Yes

Function	Description	Example	Runs in the server side context
lastEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if the last value in the list is equal to <*value*> and otherwise returns FALSE. The lastEquals function returns a Boolean result.	lastEquals (Arrays.asList(new Integer[]{1, 1}), 1) This example returns TRUE.	Yes
lastGreaterThan(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if the last value in the list is greater than <*value*> and otherwise returns FALSE. The lastGreaterThan function returns a Boolean result.	lastGreaterThan (Arrays.asList(new Float[]{3.2F, 3.5F}), 1.1F) This example returns TRUE.	Yes
lastGreaterThanEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if the last value in the list is greater than or equal to <*value*> and otherwise returns FALSE. The lastGreaterThanEquals function returns a Boolean result.	lastGreaterThanEquals (Arrays.asList(new Float[]{3.2F, 3.1F}), 3.2F) This example returns FALSE.	Yes
lastLesserThan(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if the last value in the list is lesser than <*value*> and otherwise returns FALSE. The lastLesserThan function returns a Boolean result.	lastLesserThan (Arrays.asList(new Float[]{3.2F, 3.2F}), 3.3F) This example returns TRUE.	Yes
lastLesserThanEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if the last value in the list is lesser than or equal to <*value*> and otherwise returns FALSE. The lastLesserThanEquals function returns a Boolean result.	lastLesserThanEquals (Arrays.asList(new Float[]{3.2F, 3.2F}), 3.3F) This example returns TRUE.	Yes

Function	Description	Example	Runs in the server side context
lastNotEquals(Any[] <*list*>, Any <*value*>) : Boolean	Compares <*value*> to each item in the <*list*> and returns TRUE if the last value in the list is lesser than <*value*>and otherwise returns FALSE. The lastNotEquals function returns a Boolean result.	lastNotEquals(Arrays .asList(new Integer[] {1, 1}), 2) This example returns TRUE.	Yes

8.2.6 Conditional functions

A logical function is a control flow statement that executes certain expressions when a particular condition is true.

Function	Description	Example	Runs in the server side context
ifThenElse(Boolean <*condition*>, String <*result1*>, String <*result2*>) : String	Returns <*result1*> if <*condition*> is TRUE, and returns <*result2*> if <*condition*> is FALSE. This ifThenElse function is specifically for returning a String result.	ifThenElse (customer.net_worth > parameters.<namespace>.vip_net_worth_minimum, 'Customer is a VIP', 'Customer is not a VIP') This example returns 'Customer is a VIP' if customer.net_worth > parameters.<namespace>.vip_net_worth_minimum is TRUE. or it returns 'Customer is not a VIP' if customer.net_worth > parameters.<namespace>.vip_net_worth_minimum is FALSE.	Yes

Function	Description	Example	Runs in the server side context
ifThenElse(Boolean <*condition*>, Boolean <*result1*>, Boolean <*result2*>) : Boolean	Returns <*result1*> if <*condition*> is TRUE, and returns <*result2*> if <*condition*> is FALSE. This ifThenElse function is specifically for returning a Boolean result.	ifThenElse (customer.net_worth > parameters.<namespace>.vip_net_worth_minimum, TRUE, FALSE) This example returns TRUE if customer.net_worth > parameters.<namespace>.vip_net_worth_minimum is TRUE. or it returns FALSE if customer.net_worth > parameters.<namespace>.vip_net_worth_minimum is FALSE.	Yes

Function	Description	Example	Runs in the server side context
ifThenElse(Boolean <*condition*>, DateTime <*result1*>, DateTime <*result2*>) : DateTime	Returns <*result1*> if <*condition*> is TRUE, and returns <*result2*> if <*condition*> is FALSE. This ifThenElse function is specifically for returning a DateTime result.	In this example we want to return the date that the item must have been purchased at the earliest for the request to be valid. If the customer is requesting a cash refund then the return policy stipulates the purchase date should not be more than seven days ago. So we return the current date and subtract seven days. If the customer just wants an exchange or a store credit, the date threshold is today's date minus 30 days. ifThenElse (customer_refund_request.refund_or_exchange == 'Cash Refund',addDays(now(), -7), addDays(now(), -30)) This example returns January 23, 2012 8:05:23 AM (or rather the DateTime representation of January 23, 2012) if (customer_refund_request.refund_or_exchange == 'Cash Refund', and the current date-time returned by now( ) is January 30, 2012 8:05:23 AM. or it returns December 31, 2011 8:05:23 AM (or rather the DateTime representation of	Yes

Function	Description	Example	Runs in the server side context
		December 31, 2011)ifcustomer_refund_request.refund_or_exchange == 'Store Credit'and the current date-time returned by now() is January 30, 2012 8:05:23 AM.	
ifThenElse(Boolean <*condition*>, Float <*result1*>, Float <*result2*>) : Float	Returns <*result1*> if <*condition*> is TRUE, and returns <*result2*> if <*condition*> is FALSE. This ifThenElse function is specifically for returning a Float result.	In this example we demonstrate the use of nested ifThenElse statements. If the customer has a great credit score, above 750, the bank will qualify him for a loan at a 3.5% interest rate. If the credit score is between 700 and 749, the bank will qualify him for a 5.5% interest rate, otherwise the interest rate will be 8.99%. ifThenElse (customer.credit_score >= 750, 0.035, ifThenElse(customer.credit_score >= 700, 0.055, 0.0899)) This example returns 0.035 if customer.credit_score is greater than or equal to 750. or it returns 0.055 if customer.credit_score is greater than or equal to 700 and less than 750. or it returns 0.0899 if customer.credit_score is less than 700.	Yes

8.2.7 Math functions

The mathematical functions, also categorized as “math”functions in the browser context, contain a set of functions that perform mathematical operations and some functions that deal with type conversion involving integers and float numbers.

Function	**Description**	**Example**	**Runs in the server side context**
Browser context:abs(Float <*num*>) : Float Server context:abs(Float <*value*>) : Float	Returns the absolute value, or more precisely, the value ignoring the sign of a float number, <*num*>*.	abs(-4.123) returns 4.123	Yes
Browser context:abs(Integer <*num*>) : Integer Server context:abs(Integer <*value*>) : Integer	Returns the absolute value, or more precisely, the value ignoring the sign of an integer number, <*num*>*.	abs(-4) returns 4	Yes
Browser context:arrayAverage(Integer[ ] <*numAry*>) : Float Server context:avgInt(Integer[ ] <*values*>) : Float	Returns the average of the numbers in the specified list of integers, <*numAry*>*. A list, represented by the square bracket notation [ ] at the end of the data type, is an array-like representation of repeating attributes. The input can come from a repeating attribute or a hardcoded list. Lists are written with a curly bracket notation {value1,value2,...} as in the example.	arrayAverage({-100,12,14,4,-6,82}) returns 1.0	Yes

Function	Description	Example	Runs in the server side context
Browser context:arrayMax(Integer [ ] <*numAry*>) : Integer Server context:max(Integer [ ] <*values*>) : Integer	Returns the largest number in the specified list, <*numAry*>*. A list, represented by the square bracket notation [ ] at the end of the data type, is an array-like representation of repeating attributes. The input can come from a repeating attribute or a hardcoded list. Lists are written with a curly bracket notation {value1,value2,...} as in the example.	arrayMax({-100,12,14,4,-6,82}) returns 82	Yes
Browser context:arrayMin(Integer [ ] <*numAry*> : Integer Server context:min(Integer [ ] <*values*> : Integer	Returns the the smallest of the numbers in the specified list, <*numAry*>*. A list, represented by the square bracket notation [ ] at the end of the data type, is an array-like representation of repeating attributes. The input can come from a repeating attribute or a hardcoded list. Lists are written with a curly brace notation {value1,value2,...} as in the example.	arrayMin({-100,12,14,4,-6,82}) returns -100	Yes

Function	Description	Example	Runs in the server side context
Browser context:average(Integer <*num1*>, Integer <*num2*>) : Float Server context:avgInt(Integer <*int1*>, Integer <*int2*>) : Float	Returns the average of the two integers, <*num1*>* and <*num2*>*, specified as arguments.	average(4,9) returns 6.5	Yes
avgFloat(Float <*int1*>, Float <*int2*>) : Float	Returns the average of the two float numbers, <*int1*>and <*int2*>, specified as arguments.	average(4.0,9.0) returns 6.5	Yes
avgFloat(Float[ ]<*values*>) : Float	Returns the average of the numbers in the supplied list of float numbers, <*values*>). A list, represented by the square bracket notation [ ] at the end of the data type, is an array-like representation of repeating attributes. The input can come from a repeating attribute or a hardcoded list. Lists are written with a curly bracket notation {value1,value2,...} as in the example. The first index position is 0, not 1.	avg({-100.2,12.5,14.1,4.6,-6.135,82.99}) returns 1.309166666666667	Yes
Browser context:ceiling(Float <*num*>) : Integer Server context:ceiling(Float <*value*>) : Integer	Returns the integer result of rounding up <*num*>* to the nearest integer value (that is higher than <*num*>*).	ceiling(92.53) returns 93	Yes

Function	Description	Example	Runs in the server side context
floatToInt(Float <*number*>) : Integer	Returns the integer representation of <*number*>. The number is not rounded off to the nearest value, it is only truncation after the decimal.	floatToInt(92.999) returns 92	Yes
Browser context:floor(Float <*num*>) : Integer Server context:floor(Float <*value*>) : Integer	Returns the integer result of rounding off <*num*>* to the nearest integer value (that is lower than <*num*>*).	floor(92.53) returns 92	Yes
intToFloat(Integer <*number*>) : Float	Returns the float representation of the float value for <*number*>.	intToFloat(92) returns 92.0	Yes
Browser context:max(Integer <*num1*>, Integer <*num2*>) : Integer Server context:max(Integer <*int1*>, Integer <*int2*>) : Integer	Returns the larger of the two integers, <*num1*>* and <*num2*>*, specified as arguments.	max(4,8) returns 8	Yes
Browser context:min(Integer <*num1*>, Integer <*num2*>) : Integer Server context:min(Integer <*int1*>, Integer <*int2*>) : Integer	Returns the smaller of the two integers, <*num1*>* and <*num2*>*, specified as arguments.	min(4,8) returns 4	Yes
Browser context:mod(Integer <*num*><>, Integer <*divisor*>) : Integer Server context:mod(Integer <*numberDividend*><>, Integer <*numberDivisor*>) : Integer	Returns the modulo, or remainder, of <*num*>* divided by <*divisor*>*.	mod(5,2) returns 1	Yes

Function	Description	Example	Runs in the server side context
power (float <*num*>, Integer <*num*>) : float	Returns the result of raising <*num*>to the power <*power*>. Accepts an integer value as the first argument.	power(3,3) returns 27	No
Browser context:power(Float <*num*>, Integer <*power*>) : Float Server context:power(Float <*numberBase*>, Integer <*numberExponent*>) : Float	Returns the result of raising <*num*>* to the power <*power*>*. Accepts a float value as the first argument.	power(3.0,3) returns 27.0	Yes
random( ) : Float	Returns a random float number between 0 and 1.	random( ) returns 0.1263	Yes
Browser context:round(Float <*num*>, Integer <*precision*>) : Float Server context:round(Float <*number*>, Integer <*precision*>) : Float	Returns <*num*>* rounded to the nearest float number with the specified <*precision*>* in significant digits. A tie-breaking situation is handled by rounding up.	round(4.235, 2) returns 4.24	Yes
stringToFloat(String <*number*>) : Float	Returns the float representation of the string <*number*>.	stringToFloat('92.123') returns 92.123	Yes
stringToInt(String <*number*>) : Integer	Returns the integer representation of the string <*number*>.	stringToInt('92') returns 92	Yes

* Arguments differ for the function signature in the server context.

8.2.8 String functions

Strings are a representation of text. String literals are wrapped within single quotes (') in an expression. Unless otherwise noted, string functions where text is matched, compared, or replaced are case-sensitive.

Function	Description	Example	Runs in the server side context
contains(String <*str1,*>, String <*str2*>) : Boolean	Returns TRUE if <*str2*> is contained within <*str1*>. Else, returns FALSE.	contains ('The loan expirationdate has passed', 'loan') This example returns TRUE.	Yes
endsWith(String <*str1*>, String <*str2*>) : Boolean	Returns TRUE if <*str1*> ends with <*str2*>. Else, returns FALSE.	endsWith ('The loan expiration date has passed', 'passed') This example returns TRUE.	Yes
floatToString(Float <*number*>) : String	Returns the Float value passed in, <*number*>, as a String value. One of several "Data Type Conversion"function s, since the Expression editor does not perform implicit data type conversions (also known as "Data Type Casting").	floatToString(4.134) This example returns 4.134.	Yes
intToString(Integer <*number*>) : String	Returns the integer value passed in, <*number*>, as a string value. One of several "Data Type Conversion"function s, since the Expression editor does not perform implicit data type conversions (also known as "Data Type Casting").	intToString(42) This example returns 42.	Yes

Function	**Description**	**Example**	**Runs in the server side context**
length(String <*str*>) : Integer	Returns the number of characters that <*str*> has as an integer.	length ('The loan expiration date has passed') This example returns 35.	Yes
match(String <*text*>, String <*pattern*>) : Integer	Returns the character position of where the first match begins, where <*text*> is the text to search within, and <*pattern*> is the string that should be found within the search text. The first position is 0, not 1. If <*pattern*> is not found as a match within <*text*> then -1 is returned.	match ('The loan expiration date has passed', 'has passed') This example returns 25.	Yes
replace(String <*text*>, String <*matchPattern*>, String <*replacePattern*>) : String	Returns the string produced by replacing all occurrences of <*matchPattern*> within the string <*text*> with <*replacePattern*>.	replace ('The loan expiration date has passed. A notification has been sent. ', 'has','has not') This example returns 'The loan expiration date has not passed. A notification has not been sent.'	Yes
startsWith(String <*str1*>, String <*str2*>) : Boolean	Returns TRUE if <*str1*> begins with <*str2*>. Returns FALSE if not.	startsWith ('The loan expiration date has passed', 'The loan') This example returns TRUE.	Yes

Function	Description	Example	Runs in the server side context
substring(String <*str*>, Integer <*start*>) : String	Returns the string produced by extracting characters from <*str*> beginning from the position indicated by <*start*>. The first position is 0, not 1. In the browser context only one function is provided for both variations of substring.	substring('Loan Dates', 5) This example returns 'Dates'.	Yes
substring(String <*str*>, Integer <*start*>, Integer <*end*>) : String	Returns the string produced by extracting characters from <*str*> beginning from the position indicated by <*start*> up to and not including position <*end*>. The first position is 0, not 1. In the browser context, only one function is provided for both variations of substring.	substring('Loan Dates',0,9) This example returns 'Loan Date'.	Yes
toLower(String <*str*>) : String	Returns the string produced by converting all of the characters in <*str*> to lowercase.	toLower('LOAN AMOUNT') This example returns 'loan amount'	Yes
toUpper(String <*str*>) : String	Returns the string produced by converting all of the characters in <*str*> to uppercase.	toUpper('Due Date') This example returns 'DUE DATE'.	Yes

Function	Description	Example	Runs in the server side context
dateToString(DateTime <*dateValue*>, String <*format*>) : String	Returns the string representation of <*dateValue*> formatted according to the rules specified by <*format*>. This function only supports output text in the default locale of the Java Virtual Machine (JVM). The <*format*> can be specified using the same rules as the Java SimpleDateFormat class. See the standard JavaDocs available from Oracle for examples. The timezone is based on the timezone of the JVM doing the evaluation.	dateToString(now(),' MMM dd, yyyyhh:mm a') This example returns Jan 31, 2012 11:37 AM.	Yes
dateToStringUTC(DateTime <*dateValue*>, String <*format*>) : String	Returns the string representation of <*dateValue*> formatted according to the rules specified by <*format*> using Universal Time, Coordinated (UTC). This function only supports date text in the English locale. For example, November is understood, but novembre is not. The <*format*> can be specified using the same rules as the Java SimpleDateFormat class. See the standard JavaDocs available from Oracle for examples.	dateToString(now(),' MMM dd, yyyy hh:mm a') This example returnsJan 31, 2012 11:37 AM.	Yes

Chapter 9

Migrating from the Workflow Manager

9.1 Concepts

9.1.1 Workflow migration

Use the Migration navigator to migrate legacy workflows developed using Workflow Manager to the Workflow Designer. The main objective of the workflow migration is to migrate important and complex parts of workflow design automatically—expressions, data mapping, transition paths, process variables, packages, endpoints, and performers. The migration process is seamless to ensure that the process data mapping, expression, and flow remains intact. Workflow migration is supported from Documentum Repository.

Notes

- Only processes created using Workflow Manager are supported for migration.
- Activity level package settings are corrupted when a process created in Workflow Manager is migrated to Workflow Designer. For example, package1 attached to activity1 and package2 attached to activity2 in a process created in Workflow Manager, when migrated to Workflow Designer, package1 may get attached to activity2 and package2 may get attached to activity1.

In Workflow Designer, if an activity has a single output flow, the **Let Performer select next activity** option is invalid. When you migrate a workflow to Workflow Designer, if an activity has only one output flow configured with the **Let Performer select next activity** option for transition, Workflow Designer converts this option to **Select all Connected Activities**.

The following table lists the process properties that are supported and auto migrated as part of migration process:

Table 9-1: Process properties migration

Workflow Manager properties	Auto migration	Additional information
Template name	Yes	Template name is auto migrated to Label.
Original Creator	Not supported	
process owner	Not supported	

Workflow Manager properties	Auto migration	Additional information
Default alias set	Yes	
Audit Trail	Yes	
Description	Yes	
Workflow instructions	Yes	

The following table lists the components that are supported as part of process data migration:

Table 9-2: Process data migration

Workflow Manager properties	Auto migration	Additional information
Package Name	Yes	• All packages are listed in the **Process Properties** tab after migration. • Packages configured for begin activity are shown as mandatory and visible. The remaining packages are marked as non mandatory.
Package Type	Yes	
Process Variable	Yes	
Version	Yes	

The following table lists the components that are supported as part of workflow migration:

Table 9-3: Workflow flow migration

Workflow Manager properties	Auto migration
Process Flows	Yes
Show package as label	Yes

The following table lists the components that are supported as part of activity migration:

Table 9-4: Activity migration

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
Task	Activity reference ID	Not supported	
	Priority	Yes	
	Subject	Yes	
	Instructions	Yes	
Trigger	All input flows are selected	Yes	
	This number of input flows selected	Yes	
	When this event arrives	Yes	
	This activity can run more than once in a workflow	Yes	
Notification	The activity does not trigger within	Yes	Days and hours are auto migrated into minutes post migration.
	The activity's work is not finished	Yes	Days and hours are auto migrated into minutes post migration.
Transition	Select all connected activities	Yes	
	Let the activity's performer choose	Yes	
	Select next activity based on these conditions	Yes	
Data	Package Name	Yes	Packages configured for an activity are shown as mandatory and visible, and rest of the packages are shown as non-mandatory.
	Package Type	Yes	
	Version	Yes	

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Process Variable	Yes	
	Version	Yes	
Display	Activity Image	Not supported	
	Image Size	Not supported	
	Label Font	Yes	
	Show these labels		
Performer (Auto Activity)	Type of activity – Auto / Manual	Supported	
	Peformer > Workflow supervisor	Yes	
	Peformer > Repository owner	Yes	
	Peformer > Previous activity's performer	Yes	
	Peformer > Specific user	Not Supported	You must configure user from alias set or any other performer types post migration.
	Execute this method automatically	Yes	
	Save execution results	Yes	It is migrated to **Enable troubleshooting logging**.
	Method timeout in	Yes	
	Stop execution	Yes	
	Continue execution	Continue	
Performer (Manual Activity)	Workflow supervisor	Yes	
	Repository owner	Yes	
	Previous activity's performer	Yes	
	Specific User > **Assign performer(s) now**	Not supported	

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Specific User > **Have performer(s) of activity determine the performer of this activity**	Yes	You must configure **user from aliasset** or any other performer types post migration.
	Specific User > **Have performer(s) of activity determine the performer of this activity**	Yes	**Previous Activity Performer to choose at runtime** > **Any user**
	Specific User > **Define Performer alias** > **Default Alias set**	Yes	Post migration performer configuration. **User from Alias Set** > *`<name of the default aliasset>`* *`<name of user alias>`*
	Specific User > **Define Performer alias** > **Specific alias set**	Yes	**User from Alias Set** > *`<name of the aliasset>`* *`<name of user alias>`*
	Specific User > **Define Performer alias** > **Alias set of document in package**	Not supported	
	Specific User > **Define Performer alias** > **Alias set of previous performer**	Not supported	
	All Users in group > **Assign performer(s) now**	Yes	**Specific Group** > **Processed by 'Individual users and group members at the same time'**

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	All Users in group > Have performer(s) of activity determine the performer of this activity	Yes	**Previous Activity Performer to choose at runtime > Any Group** **Previous Activity Performer to choose at runtime > Processed by 'Individual users and group members at the same time'**
	All Users in group > Define Performer alias > Default Alias set	Yes	Post migration performer configuration. **Group from alias set** > *`<name of the default aliasset> <name of group alias>`* **Group from alias set > Processed by 'Individual users and group members at the same time'**
	All Users in group > Define Performer alias > Specific alias set	Yes	**Group from alias set** > *`<name of the aliasset> <name of group alias>`* **Group from alias set > Processed by 'Individual users and group members at the same time'**
	All Users in group > Define Performer alias > Alias set of document in package	Not supported	

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	All Users in group > Define Performer alias > Alias set of previous performer	Not supported	
	Single User in group > Assign performer(s) now > First to acquire the work item	Yes	**Specific Group > Processed by 'Individual users and first group member to accept the task, at the same time'**
	Single User in group > Assign performer(s) now > Least amount of unfinished work items	Yes	**Specific Group > Processed by 'Individual users and group member with fewest task'**
	Single User in group > Have performer(s) of activity determine the performer of this activity > First to acquire the work item	Yes	**Previous Activity Performer to choose at runtime > Any Group** **Previous Activity Performer to choose at runtime > Processed by ' Individual users and first group member to accept the task, at the same time'**
	Single User in group > Have performer(s) of activity determine the performer of this activity > Least amount of unfinished work items	Yes	**Previous Activity Performer to choose at runtime > Any Group** **Previous Activity Performer to choose at runtime > Processed by 'Individual users and group member with fewest task'**

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Single User in group > Define Performer alias > Default Alias set > First to acquire the work item	Yes	**Group from alias set >** *<name of the default aliasset> <name of group alias>* **Group from Aliasset > Processed by 'Individual users and first group member to accept the task, at the same time'**
	Single User in group > Define Performer alias > Default Alias set > Least amount of unfinished work items	Yes	**Group from alias set >** *<name of the default aliasset> <name of group alias>* **Group from Aliasset > Processed by 'Individual users and group member with fewest task**
	Single User in group > Define Performer alias > Specific alias set > First to acquire the work item	Yes	**Group from alias set >** *<name of the aliasset> <name of group alias>* **Group from Aliasset > Processed by 'Individual users and first group member to accept the task, at the same time**

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Single User in group> Define Performer alias > Specific alias set > Least amount of unfinished work items	Yes	**Group from alias set** > *`<name of the aliasset>`* *`<name of group alias>`* **Group from Aliasset > Processed by 'Individual users and group member with fewest task**
	Single User in group > Define Performer alias > Alias set of document in package	Not supported	
	Single User in group > Define Performer alias > Alias set of document in package	Not supported	
	Single User in group > Define Performer alias > Alias set of previous performer	Not supported	
	Some user from group > Assign multiple performers now > Specific users and/or groups	Not supported	
	Some users from group > assign multiple performers now > Performer alias(es) which will be resolved by the workflow initiators	Yes	**Group from Aliasset** > *`<name of aliasset>`* *`<name of group alias>`* **Group from Aliasset > Processed by 'Individual users and group member at the same type**

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Some users from group > assign multiple performers now > Performer alias(es) which will be resolved at run-time from the alias set		**Group from Aliasset** > `<name of aliasset>` `<name of group alias>` **Group from Aliasset > Processed by 'Individual users and group member at the same type**
	Some users from group • Have performer(s) of activity determine the performer of this activity • Specific groups		**Previous Activity Performer to choose at runtime > Specific Group > Processed by 'Individual users and first group member to accept the task, at the same time**
	Some users from group > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved by the workflow initiator		**Previous Activity Performer to choose at runtime > Group from Aliasset >** `<default aliasset name>` `<name of group alias>` **Previous Activity Performer to choose at runtime > Individual users and first group member to accept the task, at the same time**

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Some users from group > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved at run-time from the alias set		**Previous Activity Performer to choose at runtime > Group from Aliasset >** *`<aliasset name>`* *`<name of group alias>`* **Previous Activity Performer to choose at runtime > Individual users and first group member to accept the task, at the same time**
	Multiple sequential performers > assign multiple performers now > Specific users and/or groups	Not supported	
	Multiple sequential performers > assign multiple performers now > Performer alias(es) which will be resolved by the workflow initiator		**Group from Aliasset** > *`<name of default aliasset>`* *`<name of group alias>`* **Group from Aliasset > Individual users and the first group member to accept the task, in sequence**
	Multiple sequential performers > assign multiple performers now > Performer alias(es) which will be resolved at run-time from the alias set		**Group from Aliasset > <name of aliasset> > <name of group alias>** **Group from Aliasset > Individual users and the first group member to accept the task, in sequence**

Tab Name	Workflow Manager properties	Auto migration	Additional information or equivalent configuration in Workflow Designer
	Multiple sequential performers > Have performer(s) of activity determine the performer of this activity > Specific groups		**Previous Activity Performer to choose at runtime > Specific Group** **Previous Activity Performer to choose at runtime > Individual users and the first group member to accept the task, in sequence**
	Multiple sequential performers > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved by the workflow initiator		**Previous Activity Performer to choose at runtime > Group from Aliasset : <default aliasset name>** or **Group from Aliasset : <name of group alias>** **Previous Activity Performer to choose at runtime >Individual users and the first group member to accept the task, in sequence**
	Multiple sequential performers > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved at run-time from the alias set		**Previous Activity Performer to choose at runtime > Group from Aliasset : <aliasset name>** or **Group from Aliasset : <name of group alias>** **Previous Activity Performer to choose at runtime >Individual users and the first group member to accept the task, in sequence**

The following table lists the auto performer mappings.

Table 9-5: Auto Performer Mappings

Workflow Manager performer properties	Equivalent configuration in Workflow Designer
Workflow supervisor	Workflow supervisor
Repository owner	Repository owner
Previous activity's performer	Performer from last activity

The following table lists the manual performer mappings.

Table 9-6: Manual Performer Mappings

Workflow Manager performer properties	Equivalent configuration in process Designer
Workflow supervisor	Workflow supervisor
Repository owner	Repository owner
Previous activity's performer	Performer from last activity
Specific User > **Have performer(s) of activity determine the performer of this activity**	You must configure **user from aliasset** or any other performer types post migration.
Specific User > **Have performer(s) of activity determine the performer of this activity**	**Previous Activity Performer to choose at runtime** > **Any user**
Specific User > **Define Performer alias** > **Default Alias set**	**User from Alias Set** > *`<name of the default aliasset>`* *`<name of user alias>`*
Specific User > **Define Performer alias** → **Specific alias set**	**User from Alias Set** > *`<name of the aliasset>`* *`<name of user alias>`*
All Users in group > **Assign performer(s) now**	**Specific Group** > **Processed by 'Individual users and group members at the same time'**
All Users in group > **Have performer(s) of activity determine the performer of this activity**	**Previous Activity Performer to choose at runtime** > **Any Group** **Previous Activity Performer to choose at runtime** > **Processed by 'Individual users and group members at the same time'**
All Users in group > **Define Performer alias** > **Default Alias set**	**Group from alias set** > *`<name of the default aliasset>`* *`<name of group alias>`* **Group from alias set** > **Processed by 'Individual users and group members at the same time'**

Workflow Manager performer properties	Equivalent configuration in process Designer
All Users in group > Define Performer alias > Specific alias set	Group from alias set > `<name of the aliasset>` `<name of group alias>` Group from alias set > Processed by 'Individual users and group members at the same time'
Single User in group > Assign performer(s) now > First to acquire the work item	Specific Group > Processed by 'Individual users and first group member to accept the task, at the same time'
Single User in group > Assign performer(s) now > Least amount of unfinished work items	Specific Group > Processed by 'Individual users and group member with fewest task'
Single User in group > Have performer(s) of activity determine the performer of this activity > First to acquire the work item	Previous Activity Performer to choose at runtime > Any Group Previous Activity Performer to choose at runtime > Processed by ' Individual users and first group member to accept the task, at the same time'
Single User in group > Have performer(s) of activity determine the performer of this activity > Least amount of unfinished work items	Previous Activity Performer to choose at runtime > Any Group Previous Activity Performer to choose at runtime > Processed by 'Individual users and group member with fewest task'
Single User in group > Define Performer alias > Default Alias set > First to acquire the work item	Group from alias set > `<name of the default aliasset>` `<name of group alias>` Group from Aliasset > Processed by 'Individual users and first group member to accept the task, at the same time'
Single User in group > Define Performer alias > Default Alias set > Least amount of unfinished work items	Group from alias set > `<name of the default aliasset>` `<name of group alias>` Group from Aliasset > Processed by 'Individual users and group member with fewest task
Single User in group > Define Performer alias > Specific alias set > First to acquire the work item	Group from alias set > `<name of the aliasset>` `<name of group alias>` Group from Aliasset > Processed by 'Individual users and first group member to accept the task, at the same time

Workflow Manager performer properties	Equivalent configuration in process Designer
Single User in group> Define Performer alias > Specific alias set > Least amount of unfinished work items	**Group from alias set >** *`<name of the aliasset>`* *`<name of group alias>`* **Group from Aliasset > Processed by 'Individual users and group member with fewest task**
Some users from group > assign multiple performers now > Performer alias(es) which will be resolved by the workflow initiators	**Group from Aliasset >** *`<name of aliasset>`* *`<name of group alias>`* **Group from Aliasset > Processed by 'Individual users and group member at the same type**
Some users from group > assign multiple performers now > Performer alias(es) which will be resolved at run-time from the alias set	**Group from Aliasset >** *`<name of aliasset>`* *`<name of group alias>`* **Group from Aliasset > Processed by 'Individual users and group member at the same type**
Some users from group • Have performer(s) of activity determine the performer of this activity • Specific groups	**Previous Activity Performer to choose at runtime > Specific Group > Processed by 'Individual users and first group member to accept the task, at the same time**
Some users from group > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved by the workflow initiator	**Previous Activity Performer to choose at runtime > Group from Aliasset >** *`<default aliasset name>`* *`<name of group alias>`* **Previous Activity Performer to choose at runtime > Individual users and first group member to accept the task, at the same time**
Some users from group > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved at run-time from the alias set	**Previous Activity Performer to choose at runtime > Group from Aliasset >** *`<aliasset name>`* *`<name of group alias>`* **Previous Activity Performer to choose at runtime > Individual users and first group member to accept the task, at the same time**
Multiple sequential performers > assign multiple performers now > Performer alias(es) which will be resolved by the workflow initiator	**Group from Aliasset >** *`<name of default aliasset>`* *`<name of group alias>`* **Group from Aliasset > Individual users and the first group member to accept the task, in sequence**

Workflow Manager performer properties	Equivalent configuration in process Designer
Multiple sequential performers > assign multiple performers now > Performer alias(es) which will be resolved at run-time from the alias set	**Group from Aliasset > <name of aliasset> > <name of group alias>** **Group from Aliasset > Individual users and the first group member to accept the task, in sequence**
Multiple sequential performers > Have performer(s) of activity determine the performer of this activity > Specific groups	**Previous Activity Performer to choose at runtime > Specific Group** **Previous Activity Performer to choose at runtime > Individual users and the first group member to accept the task, in sequence**
Multiple sequential performers > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved by the workflow initiator	**Previous Activity Performer to choose at runtime > Group from Aliasset : <default aliasset name>** or **Group from Aliasset : <name of group alias>** **Previous Activity Performer to choose at runtime >Individual users and the first group member to accept the task, in sequence**
Multiple sequential performers > Have performer(s) of activity determine the performer of this activity > Performer alias(es) which will be resolved at run-time from the alias set	**Previous Activity Performer to choose at runtime > Group from Aliasset : <aliasset name>** or **Group from Aliasset : <name of group alias>** **Previous Activity Performer to choose at runtime >Individual users and the first group member to accept the task, in sequence**

9.1.2 Best practices for workflow migration

Follow these practices while migrating a workflow from Workflow Manager to Workflow Designer:

- A process must not be modified by multiple users simultaneously.
- Processes must not be installed simultaneously from different Web browsers.
- All repositories must be in sync for the types, alias sets, and groups used in processs before a workflow is imported or exported between two repositories.
- Migrate and import operations must not be invoked simultaneously from different Web browsers.