Systemwalker
Runbook Automation

Studio User's Guide

Windows/Linux

B1X1-0116-03ENZ0(00)
September 2012

Preface

Purpose of this Document

This document describes the procedures for developing and managing Automated Operation Processes and operation components using Systemwalker Runbook Automation Studio V15.1.1.

Intended Readers

This document is intended for people who develop and manage Automated Operation Processes and operation components using Systemwalker Runbook Automation.

It is assumed that the reader is familiar with general operations on Windows operating systems.

When using advanced process modeling features, the manual assumes that the reader has basic programming skills and is familiar with JavaScript, XML technologies, and Web Services concepts.

Abbreviations and Generic Terms Used

- The term "Windows(R) 7" refers to the following products:
    - Windows(R) 7 Home Premium(x86)
    - Windows(R) 7 Professional(x86)
    - Windows(R) 7 Ultimate(x86)
    - Windows(R) 7 Home Premium(x64)
    - Windows(R) 7 Professional(x64)
    - Windows(R) 7 Ultimate(x64)
- The term "Windows Server 2008" refers to the following products:
    - Microsoft(R) Windows Server(R) 2008 R2 Standard(x64)
    - Microsoft(R) Windows Server(R) 2008 R2 Enterprise(x64)
    - Microsoft(R) Windows Server(R) 2008 R2 Datacenter(x64)
    - Microsoft(R) Windows Server(R) 2008 Standard(x86)
    - Microsoft(R) Windows Server(R) 2008 Enterprise(x86)
    - Microsoft(R) Windows Server(R) 2008 Standard(x64)
    - Microsoft(R) Windows Server(R) 2008 Enterprise(x64)
    - Microsoft(R) Windows Server(R) 2008 Datacenter(x64)
    - Microsoft(R) Windows Server(R) 2008 Standard without Hyper-V(x86)
    - Microsoft(R) Windows Server(R) 2008 Enterprise without Hyper-V(x86)
    - Microsoft(R) Windows Server(R) 2008 Datacenter without Hyper-V(x64)
- The term "Windows Vista(R)" refers to the following products:
    - Windows Vista(R) Home Basic(x86)
    - Windows Vista(R) Home Premium(x86)
    - Windows Vista(R) Business(x86)
    - Windows Vista(R) Ultimate(x86)
    - Windows Vista(R) Enterprise(x86)
    - Windows Vista(R) Business(x64)

    - Windows Vista(R) Ultimate(x64)
- The term "Windows Server 2003" refers to the following products:
    - Microsoft(R) Windows Server(R) 2003, Standard Edition
    - Microsoft(R) Windows Server(R) 2003, Enterprise Edition
    - Microsoft(R) Windows Server(R) 2003, Standard x64 Edition
    - Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition
    - Microsoft(R) Windows Server(R) 2003 R2, Standard Edition
    - Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition
    - Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition
    - Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition
- The term "Windows(R) XP" refers to the following products:
    - Microsoft(R) Windows(R) XP Professional x64 Edition
    - Microsoft(R) Windows(R) XP Professional
    - Microsoft(R) Windows(R) XP Home Edition
- The Oracle Solaris Operating System may be referred to as Solaris, Solaris Operating System or Solaris OS.
- Microsoft (R) Cluster Server and Microsoft (R) Cluster Service are referred to as MSCS.
- The versions of Systemwalker Runbook Automation that run on Windows systems are referred to as the Windows version.
- The versions of Systemwalker Runbook Automation that run on Linux systems are referred to as the Linux version.

Export Restriction

If this document is to be exported or provided overseas, it is necessary to check Foreign Exchange and Foreign Trade Law, and take the necessary procedure according to these laws.

Trademarks

Linux is a registered trademark of Linus Torvalds.

Microsoft, Windows and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Red Hat and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are trademarks or registered trademarks of VMware, Inc. in the United States and other countries.

Other company names and product names are trademarks or registered trademarks of respective companies.

The company names, system names, product names and other proprietary names that appear in this document are not always accompanied by trademark symbols (TM or (R)).

This guide uses screenshots in accordance with Microsoft Corporation's guidelines.

September 2012

Revision history
March 2012: First edition
July 2012: Second edition

Revision history
September 2012: Third edition

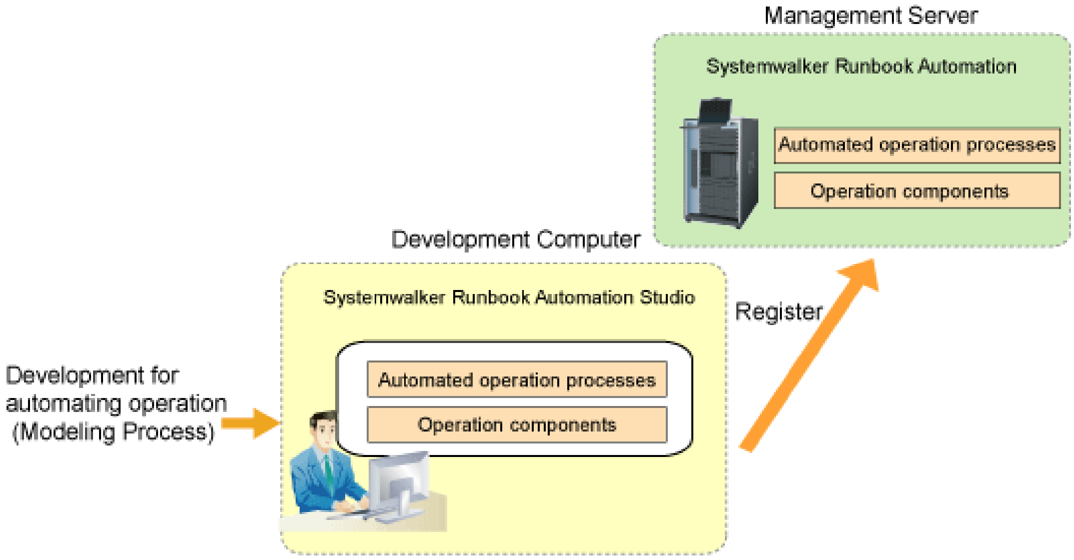
Contents

Chapter 1 Introduction

This chapter gives an overview of Systemwalker Runbook Automation Studio. It also explains basic concepts you need to know before getting started.

1.1 About Systemwalker Runbook Automation Studio

Systemwalker Runbook Automation Studio is a tool for developing Automated Operation Processes and operation components for Systemwalker Runbook Automation.


Management Server
Systemwalker Runbook Automation
Automated operation processes
Operation components
Development Computer
Systemwalker Runbook Automation Studio
Automated operation processes
Operation components
Register
Development for automating operation (Modeling Process)

Automated Operation Processes

An Automated Operation Process consists of an operation flow (process definition) that prescribes system operation procedures, and Web Console windows (forms) and other components displayed when input or confirmation tasks are performed in accordance with the operation flow.

Operation components

Operation components are collections of Ruby or Perl scripts that make it possible to automatically execute operations on business servers and linked products. Operation components are also provided in the standard Systemwalker Runbook Automation Studio product.

1.1.1 Overview of Automated Operation Processes

Automated Operation Processes are made up of the following two elements:

- Process definitions
- Definitions associated with process definitions

What are process definitions?

Process definitions are created as a flow of system operation procedures. With Systemwalker Runbook Automation Studio, process definitions can be created easily using the Process Definition Editor.

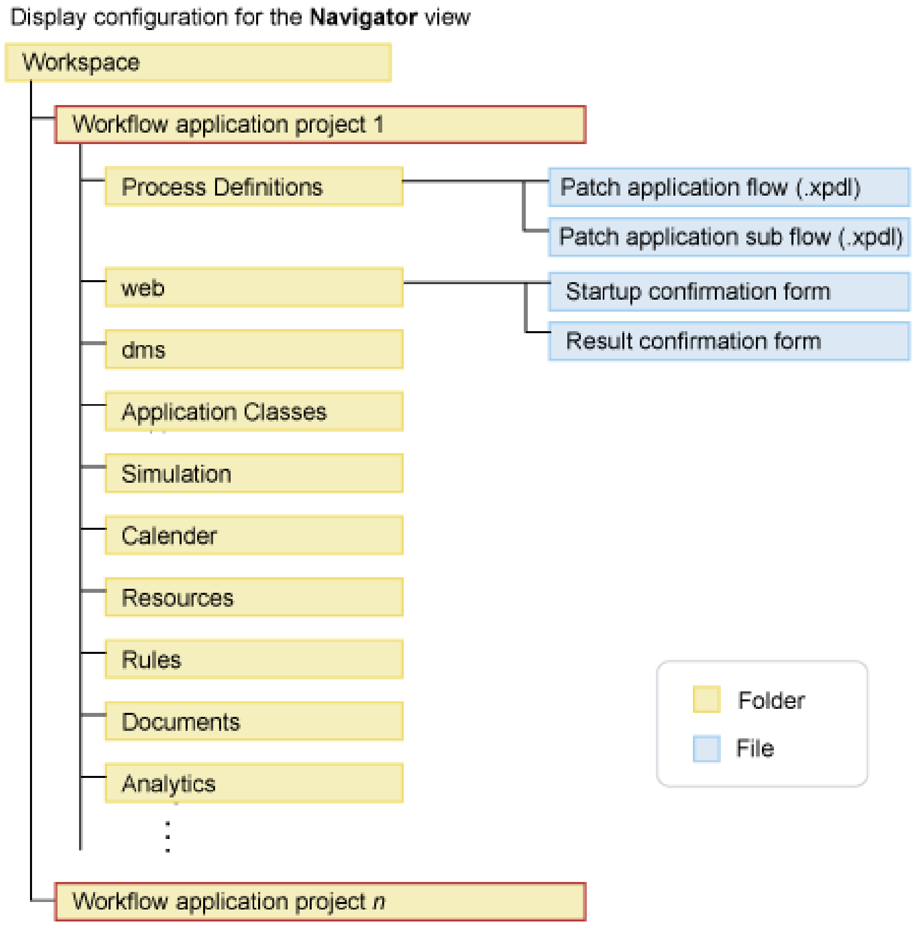
What are definitions associated with process definitions?

The following definitions are the main definitions associated with process definitions.

- Forms
  Define the windows for entering data, confirming results, and so on. With Systemwalker Runbook Automation Studio, forms can be created easily by using QuickForm from the Process Definition Editor.

Management configuration for Automated Operation Processes

Each Automated Operation Process is managed as a single workflow application project. The following diagram illustrates the configuration of an Automated Operation Process:

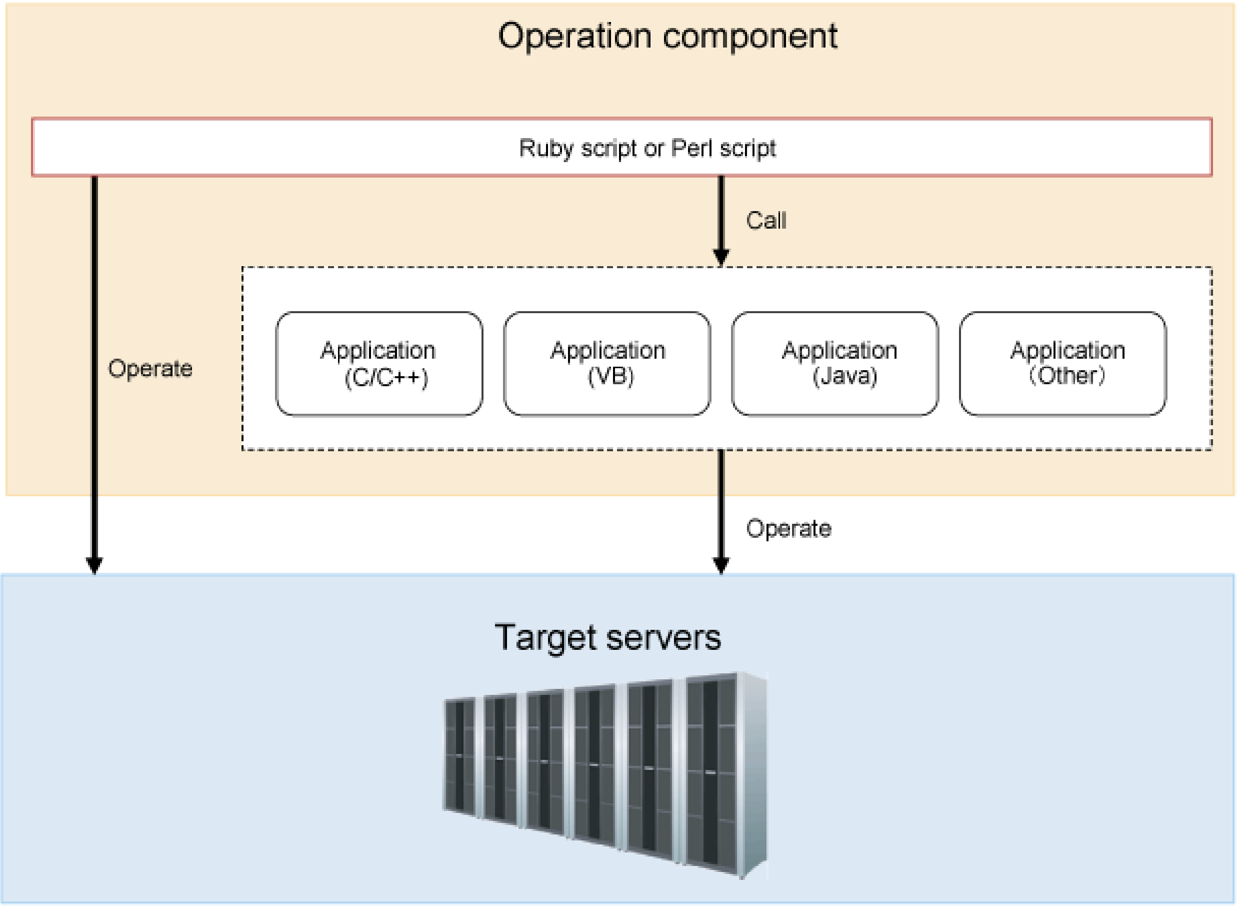

Display configuration for the **Navigator** view
Workspace
Workflow application project 1
Process Definitions
Patch application flow (.xpdl)
Patch application sub flow (.xpdl)
web
Startup confirmation form
Result confirmation form
dms
Application Classes
Simulation
Calender
Resources
Rules
Documents
Analytics
Workflow application project *n*
Folder
File

Information

Systemwalker Runbook Automation Studio stores projects in folders called "workspaces".

1.1.2 Overview of Operation Components

Operation components are collections of Ruby or Perl scripts that execute operations on business servers and linked products.



Operation component functions

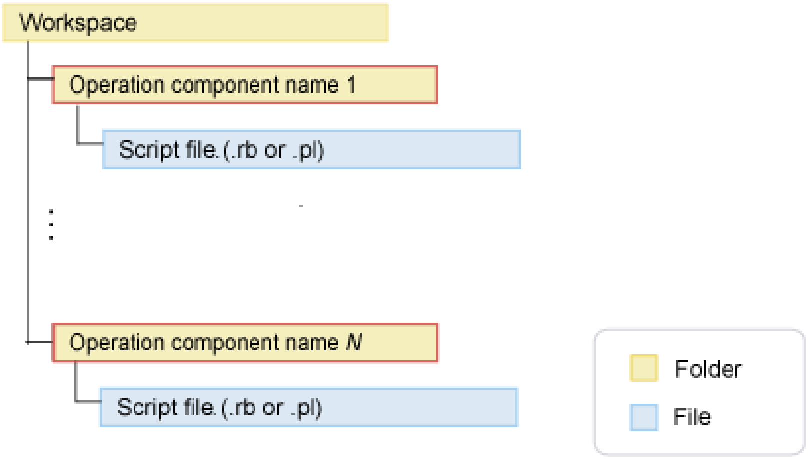
Operation components absorb the differences between different platforms. Operation components are designed so that they can be used easily without the need for detailed knowledge about system management tools or operating systems. Operation components can also perform target operations by calling existing applications.

Refer to the *Systemwalker Runbook Automation Reference Guide* for information on the structure and mechanisms of operation components.

Management configuration for operation components

Each operation component is managed as a single project. The following diagram illustrates the configuration of an operations component:

Display configuration for the **Operation Component Management** view



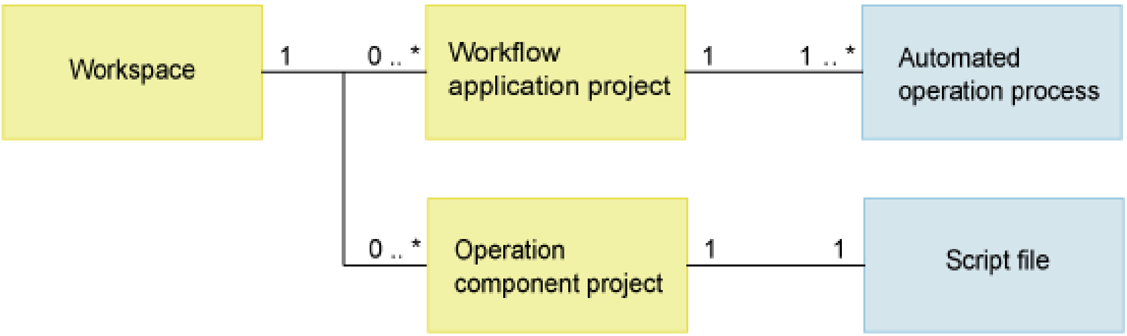
1.1.3 Relationship between Automated Operation Processes and Operation Components

This section explains the relationship between Automated Operation Processes and operation components.

Relationship between workspaces, projects and objects

Automated Operation Processes and operation components are managed as separate projects.

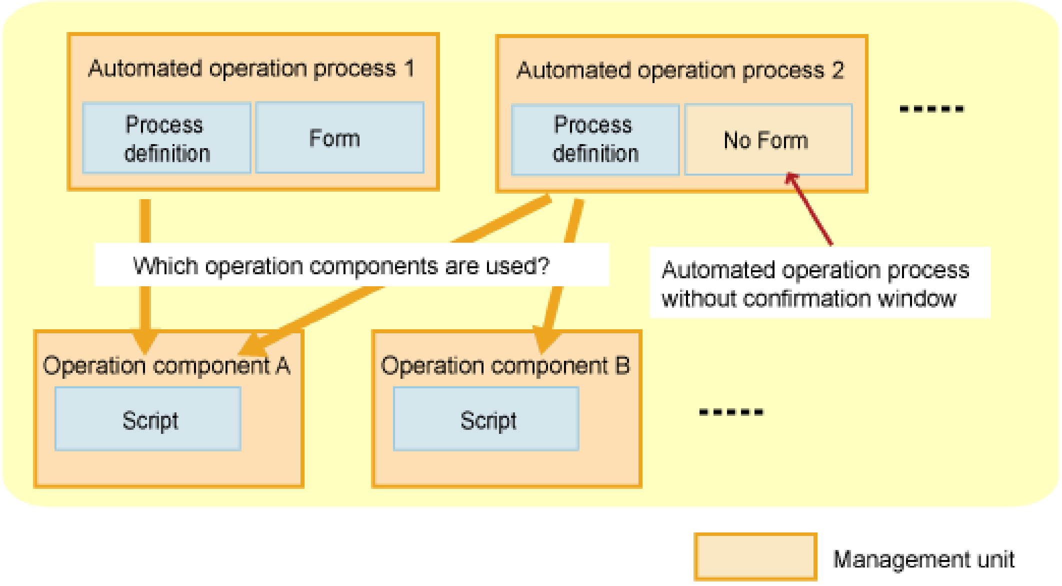
The following diagram illustrates the relationship between workspaces, projects and objects:


Workspace
1
0 .. *
Workflow application project
1
1 .. *
Automated operation process
0 .. *
Operation component project
1
1
Script file

Relationship between Automated Operation Processes and operation components

Automated Operation Processes do not actually contain operation components. Rather, Automated Operation Processes hold only information about which operation components are used from the Automated Operation Process.

Note that there are no forms for Automated Operation Processes that do not require human intervention (that is, those that do not have confirmation windows).

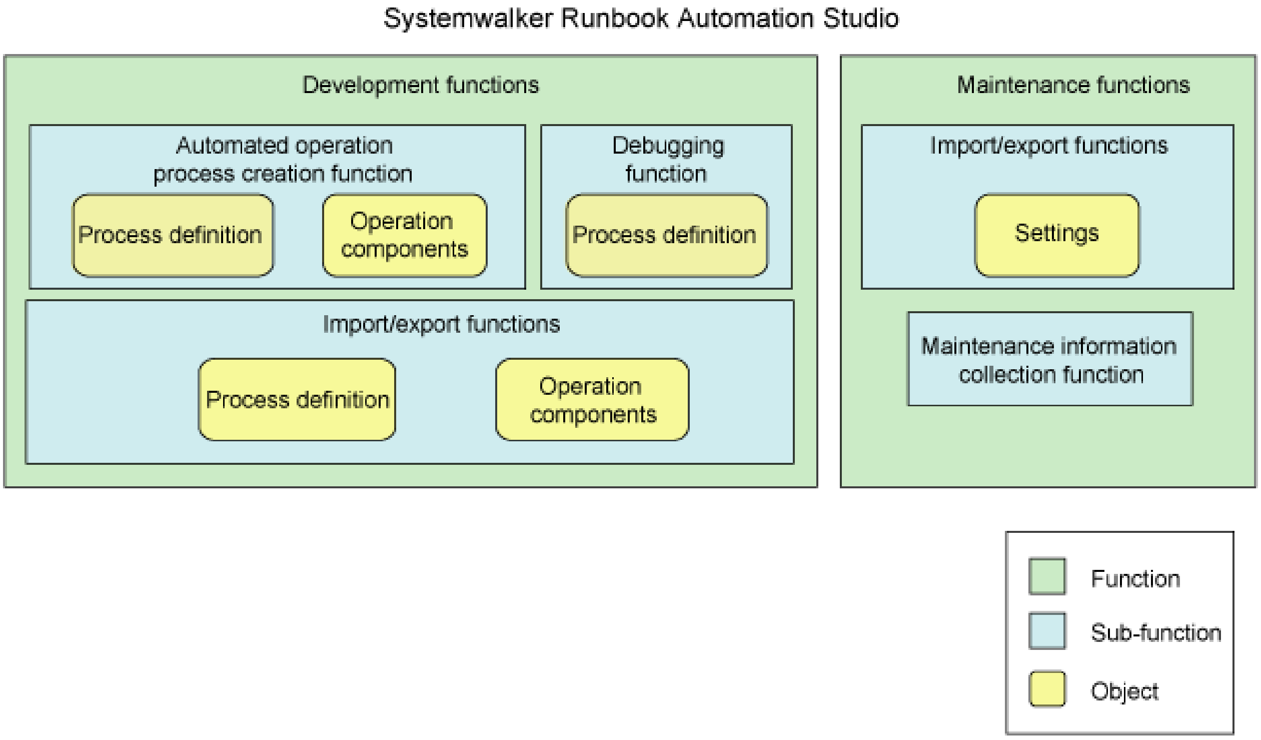

Automated operation process 1
Process definition
Form
Automated operation process 2
Process definition
No Form

Which operation components are used?
Automated operation process without confirmation window
Operation component A
Script
Operation component B
Script

Management unit

1.2 Functional Configuration of Systemwalker Runbook Automation Studio

The following diagram shows the functional configuration of Systemwalker Runbook Automation Studio:



Systemwalker Runbook Automation Studio consists of the following two functions:

- Development functions
- Maintenance functions

Each of these functions is explained below.

1.2.1 Development Functions

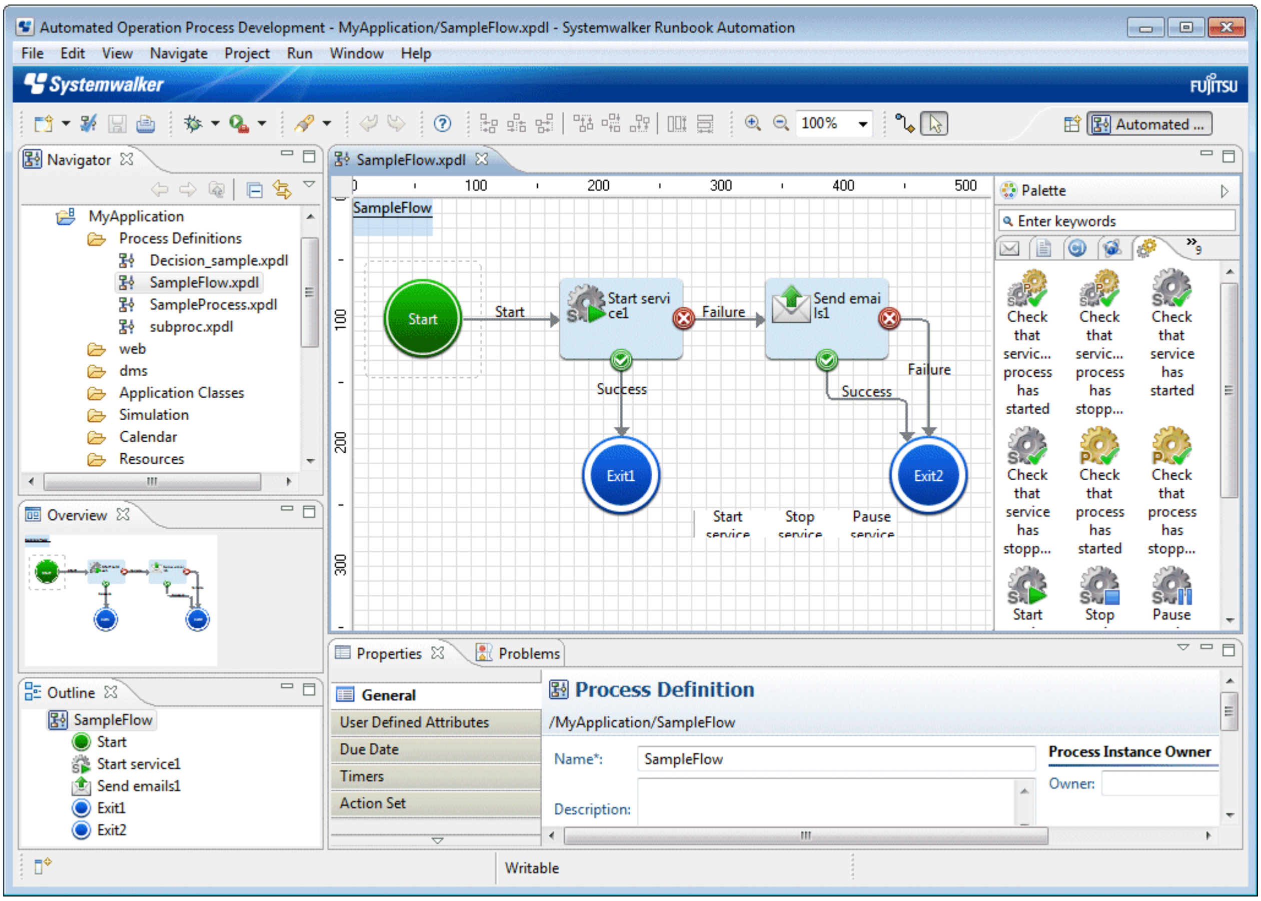
This section describes the following development functions:

- Creating Automated Operation Processes
- Creating operation components
- Importing
- Exporting

Creating Automated Operation Processes

The Automated Operation Process creation function includes tools for creating and editing the Automated Operation Processes that run on the Operation Management Server, as well as a tool for creating the forms for confirmation and other purposes that are displayed on the Web Console in accordance with the Automated Operation Process.

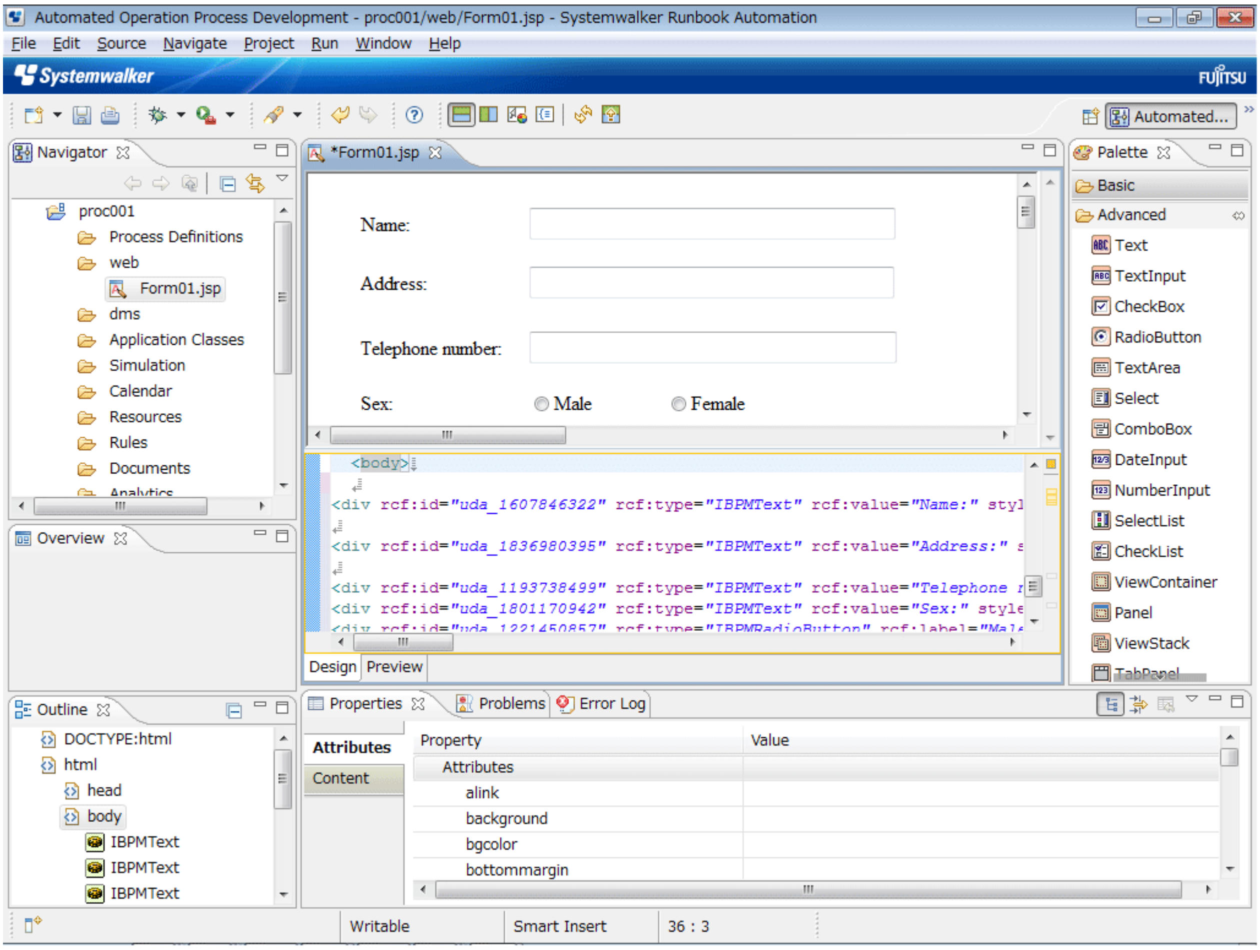
Figure 1.1 Window for developing Automated Operation Processes



Name*: SampleFlow

Writable

Figure 1.2 Window for developing forms



Name:

Address:

Telephone number:

Sex: ○ Male ○ Female

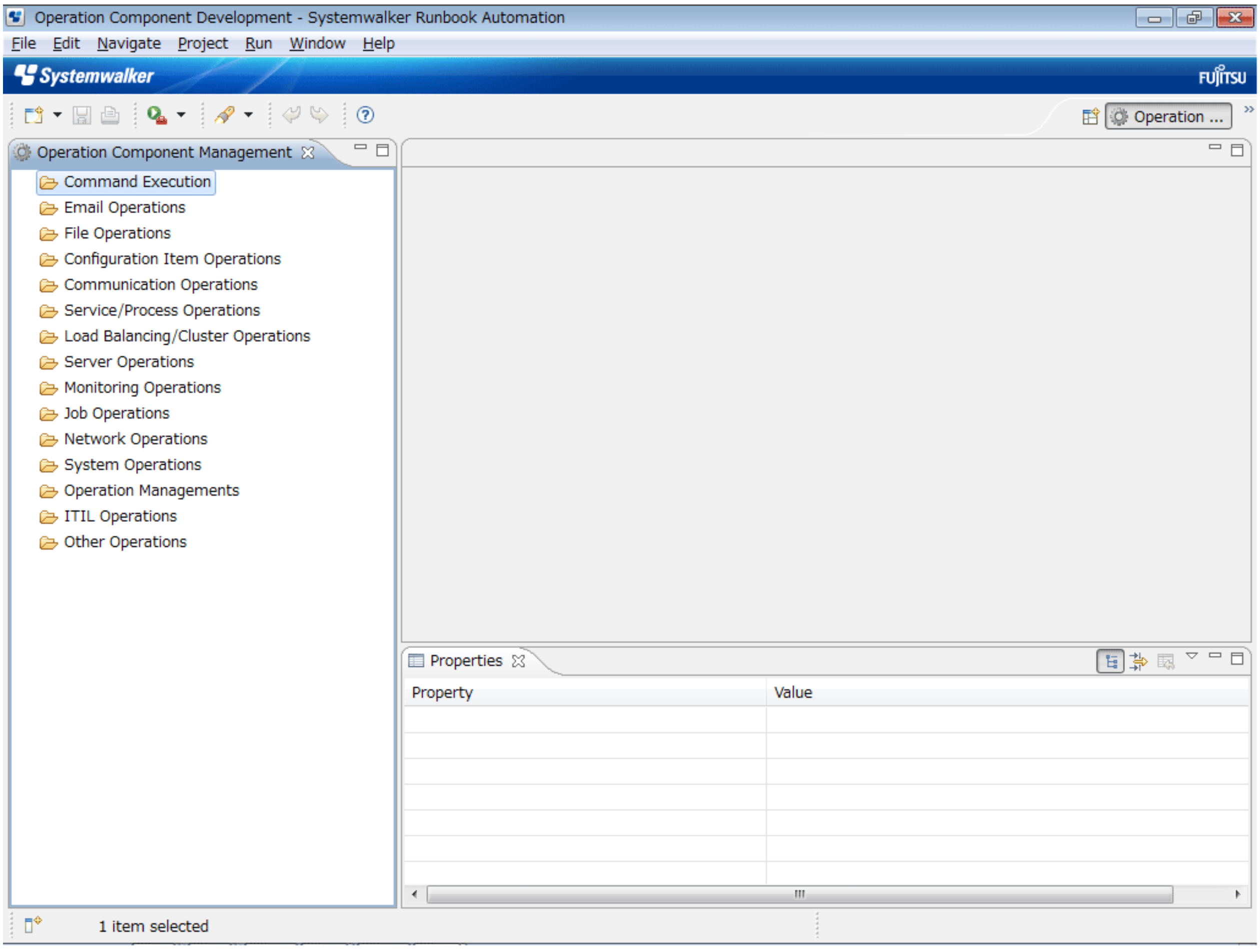
```
<body>
<div rcf:id="uda_1607846322" rcf:type="IBPMText" rcf:value="Name:" styl
<div rcf:id="uda_1836980395" rcf:type="IBPMText" rcf:value="Address:" s
<div rcf:id="uda_1193738499" rcf:type="IBPMText" rcf:value="Telephone r
<div rcf:id="uda_1801170942" rcf:type="IBPMText" rcf:value="Sex:" style
<div rcf:id="uda_1221450857" rcf:type="IBPMRadioButton" rcf:label="Male
```

Property	Value
Attributes	
alink	
background	
bgcolor	
bottommargin	

Creating operation components

The function for creating operation components is used to create components for automating operations (such as starting services or applying patches) that have been performed manually until now. If other operation components are required, they can be created by installing an external development environment. The operation components created can then be used for an Automated Operation Process.

Figure 1.3 Window for developing operation components



Importing

This function imports Automated Operation Process templates and Automated Operation Processes that have been obtained from the Management Server into Systemwalker Runbook Automation Studio on the local computer. This function can also take Automated Operation Processes and operation components that have been exported from Systemwalker Runbook Automation Studio on another computer and import them into Systemwalker Runbook Automation Studio on the local computer.

Exporting

This function exports Automated Operation Processes and operation components from Systemwalker Runbook Automation Studio. The Automated Operation Processes and operation components that have been exported can then be executed by registering them on the Management Server. These Automated Operation Processes and operation components can also be imported into Systemwalker Runbook Automation Studio on another computer.

Information

SaaS mode

Systemwalker Runbook Automation Studio supports multiple-tenant mode, but when developing Automated Operation Processes, use "SaaS" mode (the former of the two modes listed below). However, the tenant name can only be "Default".

SaaS mode:

- There are also other tenants, as well as the default tenant.
- Each tenant can retain multiple applications.
- Each tenant has a BaseURL. When tenants upload or download an application, they access this BaseURL.

Non-SaaS mode:

- There is only the default tenant.
- The default tenant can retain multiple applications.
- The default tenant has a BaseURL. When the default tenant uploads or downloads an application, it accesses this BaseURL.

1.2.2 Maintenance Functions

This section describes the following maintenance functions:

- Export settings
- Import settings
- Collect maintenance information

Export settings

This function exports settings for Systemwalker Runbook Automation Studio. The Process Definition Editor palette definition information is included in the settings information.

By exporting settings information, the settings for Systemwalker Runbook Automation Studio can be easily restored when Systemwalker Runbook Automation Studio is reinstalled. Settings information that has been exported can also be imported into Systemwalker Runbook Automation Studio on another computer.

Import settings

This function imports settings for Systemwalker Runbook Automation Studio. When Systemwalker Runbook Automation Studio is reinstalled, the settings for Systemwalker Runbook Automation Studio can be easily restored by importing the settings information that was exported beforehand.

Collect maintenance information

This function is used to collect various log files and settings files required for investigating problems that may occur when using Systemwalker Runbook Automation Studio.

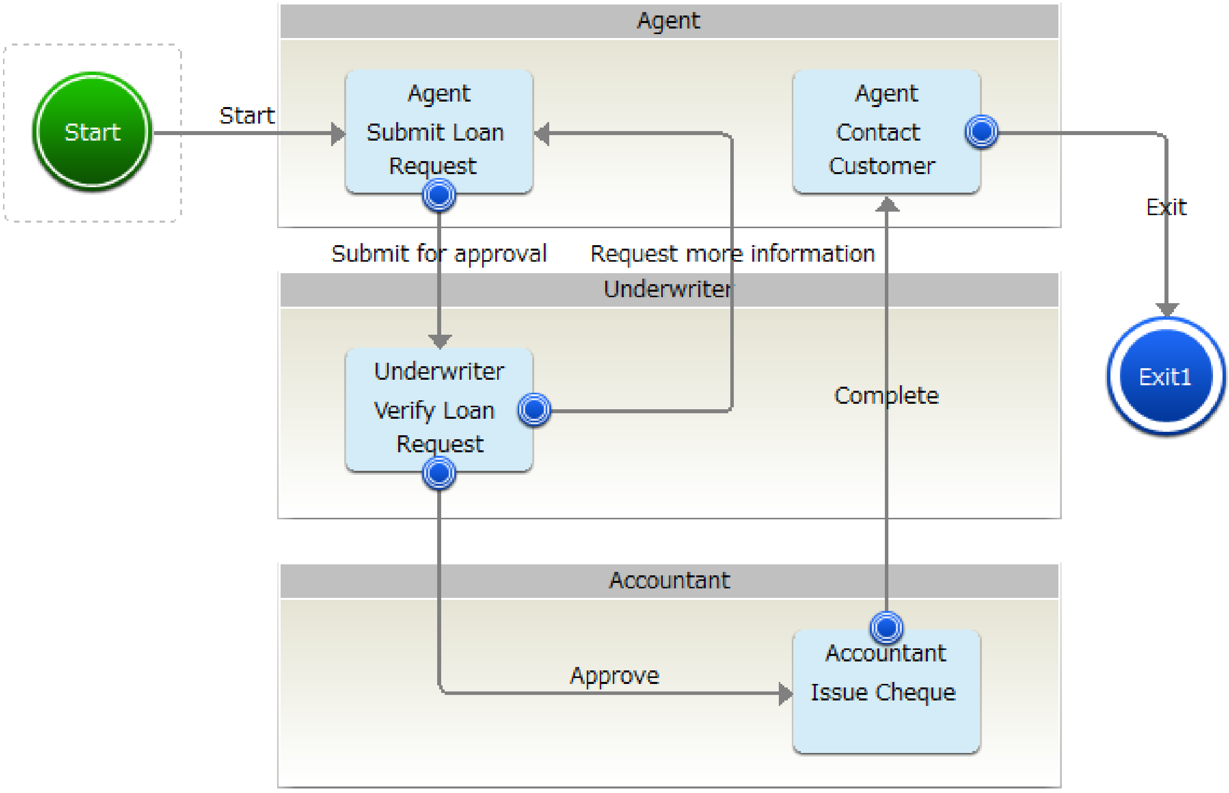
1.3 Process Definitions

Process definitions represent a business process in a form that supports automated manipulation.

Process definitions define the behavior and properties of the process instances created, including the flow of control within the process.

The following figure shows a sample process definition:

Figure 1.4 Sample Process Definition



1.3.1 Nodes

Nodes represent the steps in a process. A step could be an activity where process participants are assigned specific tasks to be completed. A step could also be a place where a decision on the process flow is made. In this case, no user action is required.

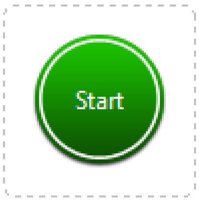
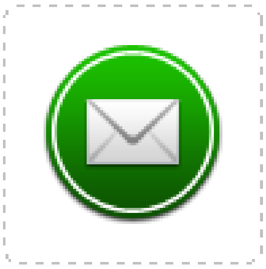
There are the following node types. Each node type is represented by its own graphical symbol.

The Process Definition Editor, which is the main window for process modeling, offers a palette containing all the nodes that can be used while modeling a process definition. These nodes are grouped based on their purpose:

- Start Node
- Activity Nodes
- Routes
- Event Nodes
- Customized Nodes

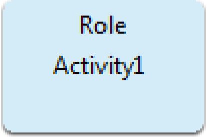
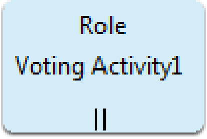
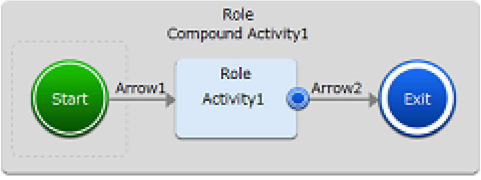
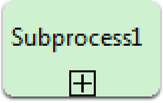
Start Node

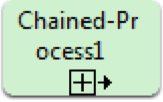
The Start Node identifies the beginning of a process. Every process definition has one and only one Start Node, which is created automatically whenever you create a new process definition. Depending on whether a trigger has been defined for the process definition, the Start Node is displayed as follows:

Node Type	Symbol	Description
● Start Node		Start Node of a process definition for which no trigger has been defined so far. This is the default notation when starting to create a process definition.
● Start Node		Start Node of a process definition for which one or more triggers have been defined.

Activity Nodes

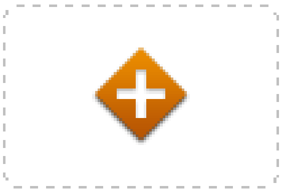
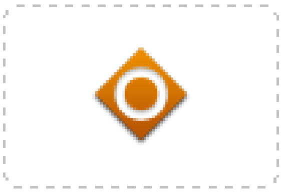
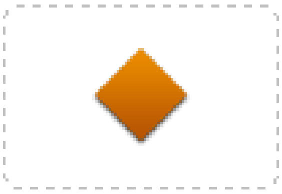
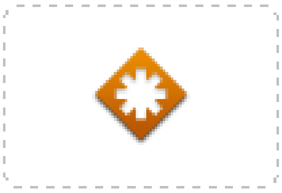
These are nodes that usually involve human activity in order to complete them:

Node Type	Symbol	Description
□ Activity Node		Activity Nodes are the main building elements of a process definition. They model tasks that require user actions or decision-making. They define the tasks, the forms associated with the task and the users assigned to perform the task.
II Voting Activity Node		A Voting Activity Node allows users to work on an activity in collaboration with one another. All users can make their own choice (or vote). All of their votes are polled. The winning vote, represented by one of the outgoing arrows, is determined by voting rules.
− Compound Activity Node	 Role Compound Activity1 Start Arrow1 Role Activity1 Arrow2 Exit	When you want to refer to the process state in terms that are larger in scope than its individual activities, a Compound Activity Node is used. A Compound Activity Node is a container that contains various nodes and arrows.
⊞ Subprocess Node		A Subprocess Node represents a complex task. The details of that task are defined in another process definition. When a Subprocess Node is reached, control is passed to the subprocess. The parent process waits for the subprocess to complete and the results to come back. Subprocesses are used to break tasks into a hierarchy of easier-to-handle units. Note: If you click the + sign on the node, an image of the subprocess definition will be displayed. If you double-click the + sign, the subprocess definition will be opened in a separate editor.

Node Type	Symbol	Description
Chained-Process Node		Like a Subprocess Node, a Chained-Process Node represents a complex task. However, this task can be accomplished independently from the tasks defined in the parent process definition. Once a chained process is started, the parent process continues with its own process flow without waiting for the chained-process to complete. Note: If you click the + sign on the node, an image of the chained process definition will be displayed. If you double-click the + sign, the chained process definition will be opened in a separate editor.

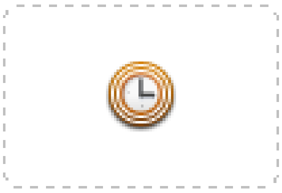
Routes

This area contains nodes that involve branching and process flow control. They are also called "Gateways". They are all automatic nodes that do not involve human interaction.

Node Type	Symbol	Description
AND Node		An AND Node synchronizes multiple branches in a process. When an AND Node is reached, the process does not continue until all activities that lead to this node are completed.
OR Node		An OR Node splits process flow into multiple parallel branches. When an OR Node is reached, it activates all subsequent nodes connected to it simultaneously.
Conditional Node		A Conditional Node controls the process flow by selecting 1 of multiple options based on the conditions that were specified. Conditional Nodes allow for automated decision-making. When a Conditional Node is reached, the process continues along the arrow that satisfies the criteria specified.
Complex Conditional Node		A Complex Conditional Node, in the same way as the Conditional Node, controls the process flow by selecting 1 of multiple options based on the conditions that were specified. The Conditional Node can only specify simple conditions that compare one value with another. However, the Complex Conditional Node can specify detailed conditions using JavaScript expressions.

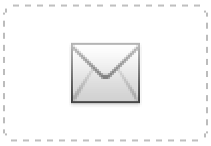
Event Nodes

This area includes nodes that involve some kind of event related to the process flow. They are automatic nodes that do not involve human interaction.

Node Type	Symbol	Description
🕒 Delay Node		A Delay Node suspends process execution for a certain amount of time. The amount of time is specified with one or more timers. When a Delay Node is reached, the process pauses until the first timer attached to the Delay Node fires. Then the Delay Node activates all subsequent nodes connected to it simultaneously.
● Exit Node		An Exit Node identifies the end of a process branch. Every Exit Node process definition has at least one Exit Node.

Customized Nodes

This area includes nodes that are customized to perform certain tasks. They are all automatic nodes that do not involve human interaction.

✉ Email Node		An Email Node sends out predefined emails. After that, it activates all subsequent nodes connected to it simultaneously. No user action is required with this node.

1.3.2 Property Symbols

You can assign specific properties to process definitions, nodes and arrows. These properties are represented by symbols. The property symbols help you to easily identify the properties that have been assigned to a process definition, node or arrow. Systemwalker Runbook Automation Studio displays these properties in the Process Definition Editor (for more information, refer to section 2.2.15 Displaying Properties).

You can turn off or on the display of the properties in the Process Definition Editor by selecting **Set** from **Window** menu and selecting **Show Details** from the **View** menu.

Depending on the specified properties, the following symbols are used:

Property Type	Symbol	Description
Trigger	⚑	Triggers move data from external systems into process instances. The trigger symbol is displayed when one or more triggers have been defined for a process definition or node. For more information on triggers, refer to section 11.9 Using Triggers. Triggers can be used with process definitions and Activity Nodes
Timer	🕒	Timers trigger Java Actions when they expire. The timer symbol is displayed when a due date or one or more timers have been defined. For more information on timers, refer to section 6.22 Using Due Dates and Timers. Timers can be used with process definitions, Activity Nodes, Voting Activity Nodes, and Compound Activity Nodes
Action	⚙ or ⚙	Java Action is executed when a process definition or node starts or completes. The Java Action symbol is displayed when one or more Java Actions have been defined. For more information on Java Actions, refer to section 11.1 Using Java Actions. Java Action properties can be used with process definitions and the following nodes: Activity, Voting Activity, Compound Activity, Subprocess, Chained-Process, Remote Subprocess, Conditional, Complex Conditional, Delay, AND, and OR Nodes

Property Type	Symbol	Description
Error	⚙	Error Actions are executed when a regular Java Action throws an exception. The Error Action symbol is displayed when Error Actions have been defined. For more information on Error Actions, refer to section 11.8 Defining Exception Handling. Error Actions can be used with process definitions and all nodes, except Start Nodes and Exit Nodes.
Form	▤	When forms are used, data can be acquired from the process and displayed for the user. If a form has been defined, the form symbol is displayed. Refer to Chapter 8 Using Forms for information on forms. Forms can be used with the following nodes: Activity Nodes, Voting Activity Nodes, Compound Activity Nodes and Start Nodes. Forms cannot be used with process definitions.
Agent	⚙	Java Agents are Java programs that perform well-defined tasks in the background. For more information on Java Agents, refer to section 11.10 Defining Java Agents. Java Agents can only be used with Activity Nodes. The Java Agent symbol is only displayed when an Activity Node is defined as an Agent.
Iterator (Parallel) Loop	|||	The Iterator (Parallel) Loop generates multiple node instances in parallel. The Iterator Loop symbol is displayed when an Iterator (Parallel) Loop is defined. Refer to "6.19 Looping Definitions" for details about Iterator (Parallel) Loops. Iterator (Parallel) Loops can be used with the Activity, Subprocess, and Chained-Process nodes.
Sequential Loop	↻	The Sequential Loop generates node instances sequentially till a condition is satisfied. The Sequential Loop symbol is displayed when a Sequential Loop is defined. Refer to "6.19 Looping Definitions" for details about Sequential Loops. Sequential Loops can be used with the Activity, Subprocess, and Compound Activity nodes.

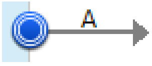
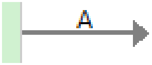
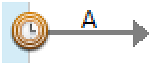
Refer to section 1.3.3 Arrows for a description of the symbols used with arrows.

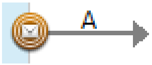
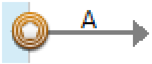
1.3.3 Arrows

Arrows define the flow of events. They are the connectors between nodes that guide the process flow from one node to another.

When an arrow originates from an Activity Node, it represents a choice that the activity assignee can make in response to the activity.

Depending on where an arrow originates and where it goes to, it can have different properties. Again, one symbol is associated with each property:

Property Type	Symbol	Description
"Normal" outgoing arrow		The arrow originates from an Activity Node, Voting Activity Node, or Compound Activity Node. This expression means that only one arrow is executed at runtime.
Outgoing arrow from a node except Activity, Voting Activity, or Compound Activity.		The arrow originates from a node except Activity, Voting Activity, or Compound Activity. This expression means that all arrows are executed at runtime.
Timer		The arrow originates from an Activity Node, Voting Activity Node, or Compound Activity Node for which a timer is defined. The timer must have at

Property Type	Symbol	Description
		least one Make Choice Java Action defined for its timer.
Trigger		The arrow originates from an Activity Node for which one or several triggers are defined.
Multiple events		The arrow originates from an Activity Node for which one or several triggers and timers are defined.

1.3.4 Forms

Processes may involve decisions and actions that are based on information. In Interstage BPM Studio for Systemwalker, "form" is used as an abstract concept. A form can be any technology that can take data from a process and present it to the user so that information can be provided and accessed.

Forms can be associated with Start Nodes, Activity Nodes, Compound Activity Nodes, and Voting Activity Nodes. Refer to Chapter 8 Using Forms for details.

1.3.5 Roles

In a typical business process, several people are performing tasks. These people are called Roles. A Role is a name given to a single person or to a group of users. Roles are defined according to the needs of the organization.

Most commonly, users are grouped according to the kind of work they perform. They can also be grouped according to their authority, responsibility, skill, or profession.

For example, a Manager Role might contain the first-line managers in an organization.

When you assign activities (Activity Nodes, Compound Activity Nodes, or Voting Activity Nodes) to Roles, you are recommended to assign group without assigning users directly. This will allow you to be flexible with change of organization If personnel changes occur, only the Role definition needs to be updated and not all the process definitions that use the Role

1.4 Workflow Application Projects

A Workflow Application project is a project for managing all files (process definitions, forms, icons, simulation scenarios, etc.) that are required for performing any task with your application. Systemwalker Runbook Automation Studio allows you to design Workflow application projects offline. Later you can deploy such an application to the Management Server from the Web Console.

Workflow Application projects help you to develop and deploy rich internet and composite applications very rapidly by providing a design and runtime environment that does not require skills in JavaScript, HTML and CSS. If you have larger projects and consequently have a high number of layouts and adapter classes, you might want to structure your development activities in a better way. For this reason, Systemwalker Runbook Automation Studio allows you to separate your project's resources in Workflow Application projects.

A Workflow Application project is represented in the file system by a single directory. It groups all the process definitions, forms, attachments, etc. which are specific to a business process. The default project directory contains all components that are required for running a process solution (refer to section Appendix B Project Components for a complete overview), for example:

- QuickForm JSP files
- HTML files generated from layout definitions
- Required images or required HTML files
- Java Script files
- Simulation scenarios and simulation result files
- Calendar files
- Configuration file for Java Agents used by the application
- FTP Agent files

- HTTP Agent files
- Custom Config files
- File Listener files
- Adapter classes
- Rule files
- Document files
- Help files

Also, refer to 2.2 Workbench Window for the Automated Operation Process Development Perspective for more information about these components.

1.5 Offline Editing and Transfer to Server

Systemwalker Runbook Automation Studio is a standalone tool. It allows you to model your Workflow Application projects together with process definitions offline, that is, without connection to the Management Server. All artifacts that you create with Systemwalker Runbook Automation Studio are stored in a file system folder of your choice.

You can export individual process definitions or entire Workflow Application projects from the Systemwalker Runbook Automation Studio and import or deploy them on the Management Server.

Transferring Individual Process Definitions

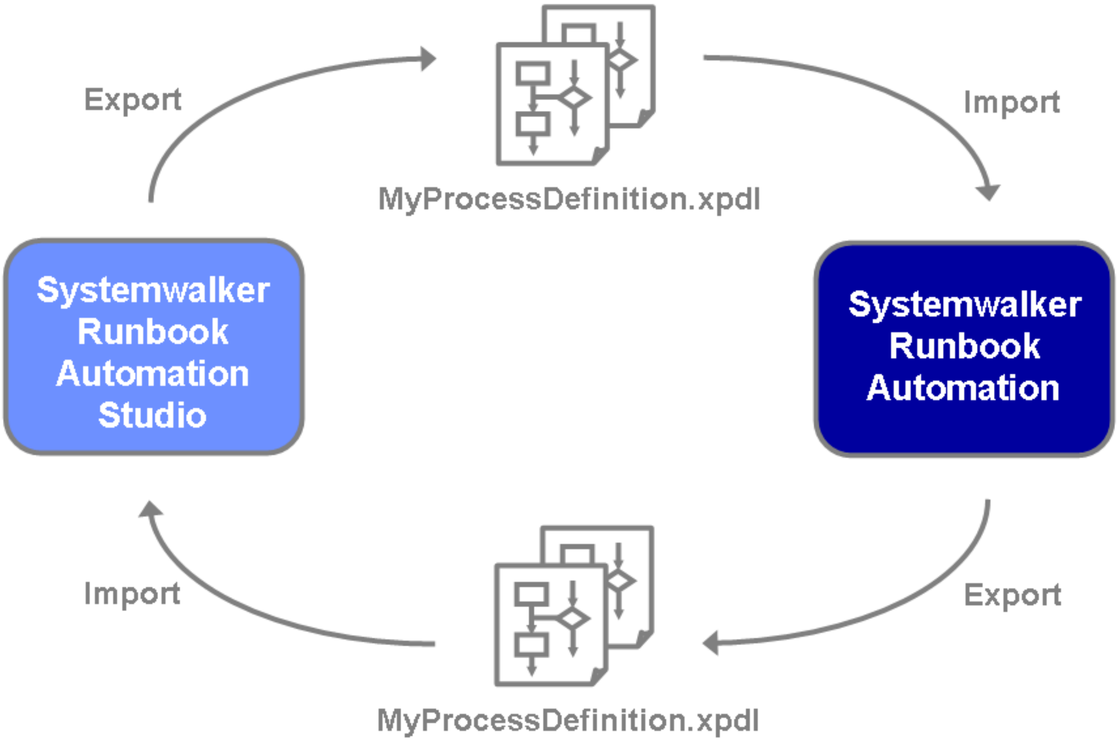
On the Management Server, you can publish process definitions and start process instances from it.

Vice versa, you can export process definitions from the Management Server and import them into Systemwalker Runbook Automation Studio. After completing your modifications, you transfer the modified process definition again to the Management Server.

XPDL is used to exchange data between Systemwalker Runbook Automation Studio and the Management Server. XPDL stands for XML Process Definition Language and is a standard file format for process definitions.

The following figure shows how process definitions are transferred between Systemwalker Runbook Automation Studio and the Management Server.

Figure 1.5 Transferring Process Definitions



Transferring Workflow Application Projects

Transferring Workflow Application Projects

Systemwalker Runbook Automation Studio allows you to transfer entire Workflow Application projects between your workspace, the local file system, and the Management Server.

These files contain everything that is required to run your application.

You can transfer projects as a whole, or you can select specific components to be transferred. You can then use the Web Console for deploying the application on the Management Server and thus make the application available for public use.

The name of a Workflow Application project must uniquely correspond to the application on the Management Server.

When transferring Workflow Application projects, you have the following options:

Download projects from an application server: This function downloads projects from a Management Server. This function allows you to download Workflow Application projects from a remote server to your workspace. You can select entire projects or specific components to be downloaded.

For more information, refer to section 3.1.14 Downloading Workflow Application Projects from a Server.

Upload projects to an application server: This function uploads projects to a Management Server. This function allows you to upload a local Workflow Application project from your workspace to a remote server. Similarly to the **Upload projects to an application server** function, you can upload entire projects or selected components.

For more information, refer to section 3.1.15 Uploading Workflow Application Projects to a Server.

Import applications: This function allows you to import a project or selected components from a .bar file that is stored on your local file system.

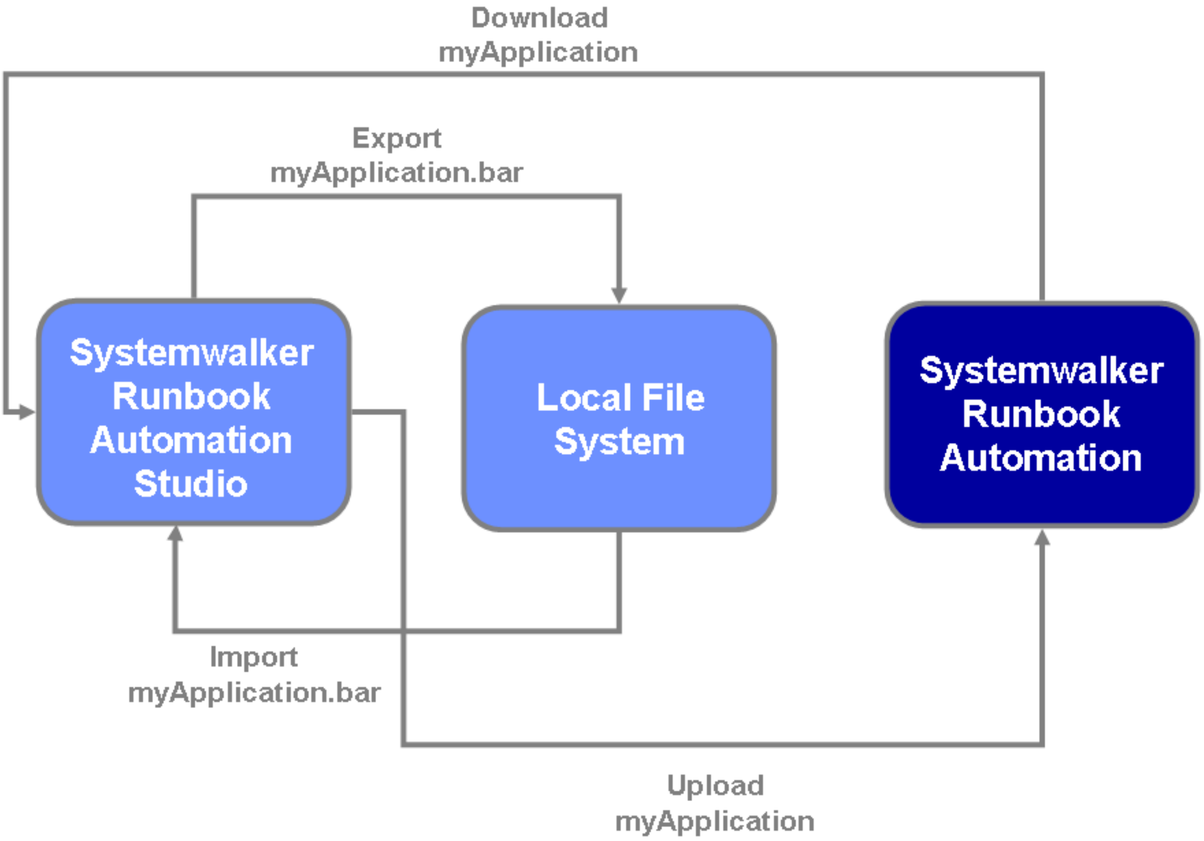
For more information, refer to section 3.1.12 Importing Workflow Application Projects.

Export applications: This function allows you to export a local Workflow Application project from your workspace to the local file system. You can select specific components to be exported, too. The exported .bar file can then be deployed on the Management Server.

For more information, refer to section 3.1.13 Exporting Workflow Application Projects.

The following figure shows how Workflow Application projects can be transferred between Systemwalker Runbook Automation Studio and the Management Server.

Figure 1.6 Transferring Workflow Application Projects



1.6 Online Editing: Server Projects

In addition to the possibility of working offline in Systemwalker Runbook Automation Studio as a standalone tool, you can work directly on a connected Management Server. Process definitions that you create with Systemwalker Runbook Automation Studio are stored directly in the database located on the Management Server.

You can also transfer process definitions created locally directly to the server.

You can only connect to system applications for the default tenant in SaaS mode or non-SaaS mode.

1.7 Simulation Scenarios and Results

A simulation scenario defines criteria for simulating the execution of a business process as defined in a process definition. The scenario defines the behavior and properties of the process instances created from the process definition including the flow of control within the process, for example, the probability with which a specific activity will be performed.

A simulation scenario consists of a process definition, criteria to indicate the start and end of the simulation run, and information about the individual activities, such as participants, the data to be handled, the estimates of an activity or resource. A simulation scenario file includes the details of a process simulation as well as - after running a simulation - the simulation results. You find more information on the simulation process in section 12.2 Defining a Simulation Scenario.

The data that is specified for a scenario, such as the start and end times for executing the simulation and the probability of executing a particular activity, can be acquired from the history data for process instances that is held on the server. For more information, refer to "12.2.2 Using Simulation Values From History".

The simulation of a process can help you in calculating cost and time for specific activities, and thus optimize your business processes.

Chapter 2 Basic Information and Customization for the User Interface

This chapter introduces the user interface of Systemwalker Runbook Automation Studio. You will learn how to work with user interface components and how to customize the user interface to your needs.

2.1 Perspectives

Systemwalker Runbook Automation Studio provides you with the following three perspectives:

- **Automated Operation Process Development** perspective
- **Operation Component Development** perspective
- **Debug** perspective
  For **Debug** perspective , refer to the "Chapter 13 Process Debugging"

2.2 Workbench Window for the Automated Operation Process Development Perspective

The **Automated Operation Process Development** perspective can be used to perform the following operations:

- Creating Workflow Application projects
- Creating process definitions using activities, arrows, swimlanes, and other elements

The following method is used to start the **Automated Operation Process Development** perspective.

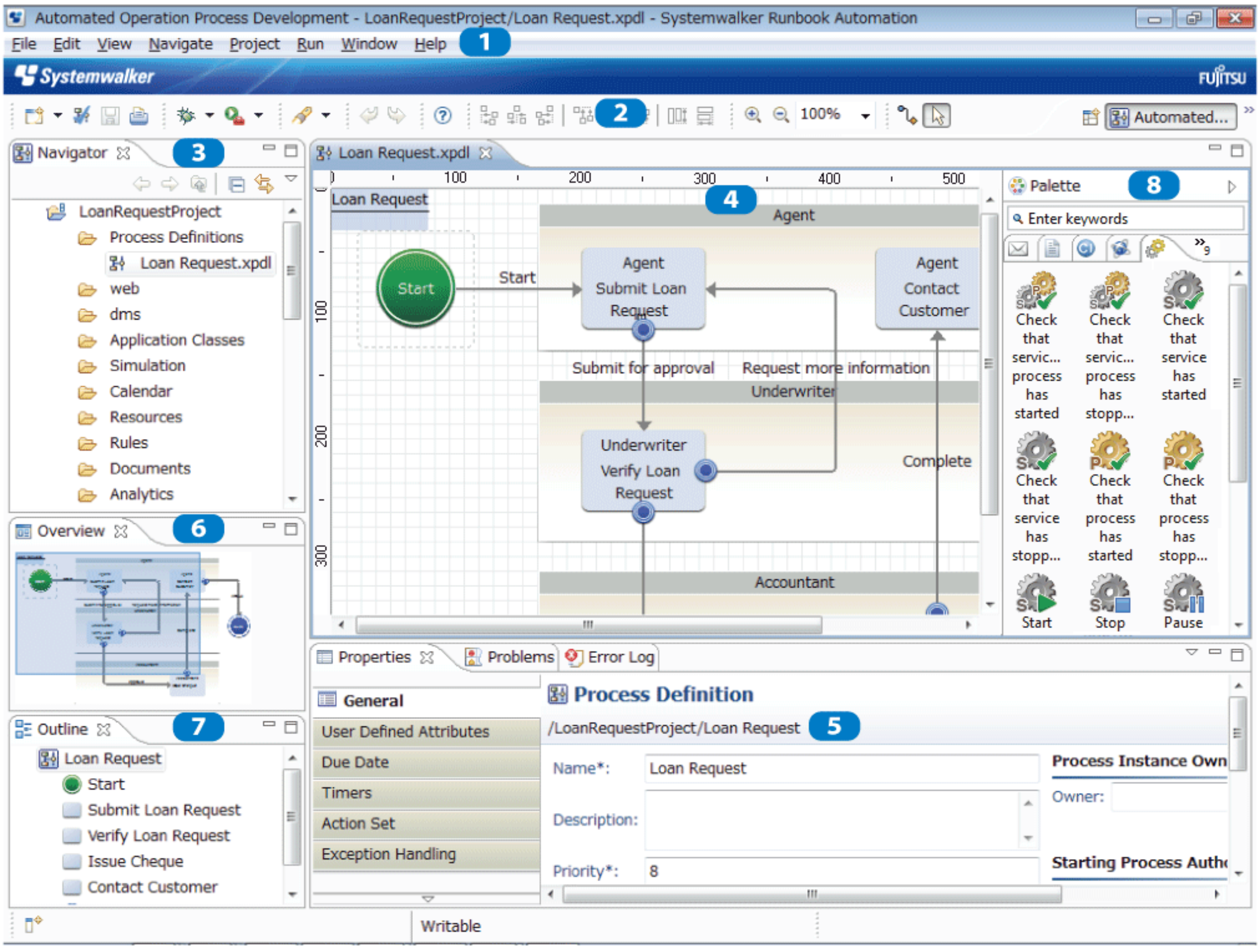
- Select **Window** >> **Open Perspective** >> **Automated Operation Process Development**.

Information

Another way to switch perspectives is by using the shortcut icons at the upper right of the Studio window.

The user interface for the **Automated Operation Process Development** perspective consists of a menu bar, a toolbar, several views, and a Process Definition Editor. These components are collectively referred to as the "workbench window". The following screenshot shows the workbench window:

Figure 2.1 Workbench window for the Automated Operation Process Development perspective



Name*: Loan Request

Description:

Priority*: 8

The numbers in the screenshot indicate the following screen components:

- 1: Menu bar
- 2: Toolbar
- 3: Navigator view
- 4: Process Definition Editor, Ajax Page Editor, etc.
- 5: **Properties** view, **Problems** view, or Error Log view (Used to display search results as well)
- 6: **Overview** view
- 7: **Outline** view
- 8: **Palette** view

The next few sections explain each of these components.

2.2.1 Menu bar

The menu bar contains menus and menu options with which you access the Systemwalker Runbook Automation Studio functions.

File menu

The functions available from the File menu differ depending on what is selected in the Navigator view:

- Project: Workflow Application project, local project, server project
- Process definition: located on the local machine (local process definition) or located on a server

- Simulation Scenarios project, process fragment project
- Folder of a Workflow Application project

Menu Option	Description
New >> Project >> General	It is not supported in Systemwalker Runbook Automation Studio
New >> Project >> Interstage BPM Studio >> Project	Create new Workflow Application projects and Server projects.
New >> Project >> Systemwalker Runbook Automation Studio >> Operation components	Create new Operation components.
New >> Sample	It is not supported in Systemwalker Runbook Automation Studio
New >> Other >> General	It is not supported in Systemwalker Runbook Automation Studio.
New >> Othe r >> Interstage BPM Studio	Create new FTP agent, HTTP agent, QuickForm, custom construction,Folder, Process fragment, Process definition, Rules set, Determination rule table, Workflow Application projects, Server projects.
New >> Other >> Systemwalker Runbook Automation Studio >> Operation components	Create new Operation components.
New >> Project >> Application	Create new Workflow Application projects.
New >> Project >> Server	Create Server projects.
New >> Process Definition	Creates a new process definition.
New >> Scenario	Creates a new simulation scenario.
New >> QuickForm	Creates a new QuickForm.
New >> Agents	Creates an agentsConfig.xml file where you can define agents for Workflow Application projects.
New >> FTP Agent	Creates an FTP Agent file where you can define FTP agents for Workflow Application projects.
New >> HTTP Agent	Creates an HTTP Agent file where you can define HTTP agents for Workflow Application projects.
New >> Custom Config	Creates a Configuration file for customer's JavaAction or Agent.
New >> File Listener	Creates a file ListenerConf.xml (fixed name) file for Workflow Application projects.
New >> Process Scheduler	Creates ProcessSceduler.xml file for Workflow Application projects.
New >> Java action >> Data Source	Create DataSourceDefinition.xml file to define data source of Workflow Application projects.
New >> Calendar	Creates a new calendar .cal file where you can define new business calendars.
New >> Rules >> Rules Set	Creates Rules Set for creating Decision Tables
New >> Rules >> Decision Table	Creates Decision Tables
New >> Folder	Creates a new subfolder beneath the selected folder.
New >> File	Creates a new file beneath the selected file.
New >> Process Fragment	Creates Process Fragments to define Process Fragment Project
Close	Closes the process definition that is displayed in the Process Definition Editor, the scenario that is displayed in the Scenario editor, the resource file that is opened in a text editor or the QuickForm displayed in the Ajax Page Editor.

Menu Option	Description
Close All	Closes all process definitions, scenarios, text editors, and QuickForms.
Save	Saves the process definition of the active Process Definition Editor, the scenario of the active Scenario editor, resource files opened in a text editor, or QuickForms in the Ajax Page Editor.
Save As	Saves a process definition to another location and/or to another file name, a resource file that has been changed using a text editor, a scenario to another file name, or a QuickForm to another name.
Save All	Saves all open process definitions, resource files, scenarios and QuickForms that have been changed.
Revert	It is not supported in Systemwalker Runbook Automation Studio
Move	It is not supported in Systemwalker Runbook Automation Studio
Change Name	Change the file name of Process definition, Scenario, Resource File, and QuickForm.
Refresh	Refreshes the Navigator view.
Print	Prints the process definition displayed in the Process Definition Editor.
Switch Workspace	Switches to a different workspace. This option restarts Systemwalker Runbook Automation Studio.
Restart	Restart Studio
Send to Server	Transfers a local process definition to a server project.
Upload Application	Uploads a Workflow Application project from your workspace to a remote server
Download Application from Server	Downloads a Workflow Application project from a remote server to your workspace
Import Bar File	Imports a Workflow Application project from the local file system to your workspace.
Import	Imports resources of a Workflow Application project from the local file system to your workspace.
Export	Exports resources of a Workflow Application project from your workspace to the local file system. When you export the process definition, you can select the following formats using the Export Process Definition dialog: - XPDL1.0 - XPDL2.0 - XPDL2.1
Generate Process Documentation	Generates the report of the Process Definition displayed in an active Process Definition Editor.
Run Simulation	Simulates a scenario and generates simulation result that can be used to replay the simulation.
Properties	Displays properties of projects, local process definitions, simulation scenarios, QuickForms, resource files, and other project-related properties.
Exit	Closes Systemwalker Runbook Automation Studio.

Note

The Process Scheduler function has not been supported in Systemwalker Runbook Automation

Edit menu

The **Edit** menu opens only if a process definition, a simulation scenario or a QuickForm is open.

Menu Option	Description
Undo	Undoes the last change to a process definition or scenario.
Redo	Redoes the change that has been previously undone.
Cut	Cuts selected process definition nodes, swimlanes or groups, or text strings defined in a scenario. If a "cut" operation is executed in the Source view of the Ajax Page Editor, the selected range will be cut and stored on the clipboard as text data. If a "cut" operation is executed in the Design view, the selected component will be cut and stored on the clipboard as text data.
Copy	Copies selected process definition nodes, swimlanes or groups, or text strings defined in a scenario. If a "copy" operation is executed in the Source view of the Ajax Page Editor, the selected range will be copied and stored on the clipboard as text data. If a "copy" operation is executed in the Design view, the selected component will be cut and stored on the clipboard as text data.
Paste	Pastes cut or copied process definition nodes, swimlanes or groups, or text strings defined in a scenario. If this command is executed in the Source view of the Ajax Page Editor, the data in the clipboard will be pasted at the cursor location. Similarly, if this command is executed in the Design view, the data in the clipboard will be pasted to the Design view and the component will be selected.
Delete	Removes selected nodes, swimlanes, groups, and arrows from the process definition, or selected text strings from a scenario. In the Ajax Page Editor, the selected range or element will be deleted. If the Navigator view is active, this command will delete the projects, process definitions, scenarios, QuickForms, resource files and other files and folders in the Workflow Application project. You cannot delete process definitions for server projects, folders that represent each element of Workflow Application projects, simulation folders, or the Process Fragments project.
Select All	Selects all elements of the process definition or text strings defined in the scenario currently displayed. This command can be used in the Process Definition Editor and the Scenario Editor. In the Ajax Page Editor, this command can be executed in the Source view, in which case the entire source code will be selected.
Find/Replace	Displays the Find/Replace dialog box, which can be used to search for a string and replace it with another one.
Add book mark	It is not supported in Systemwalker Runbook Automation Studio
Add task	It is not supported in Systemwalker Runbook Automation Studio
Find Next	Finds the next instance of the currently selected text, or the next instance of the element name selected in the Outline view.
Find Previous	Finds the previous instance of the currently selected text, or the previous instance of the element name selected in the Outline view.
Incremental Find Next	Finds the next instance in incremental search mode.
Incremental Find Previous	Finds the previous instance in incremental search mode.
Align (Submenu)	Contains menu options for aligning nodes, swimlanes, and groups. This command can be used in the Process Definition Editor and the Scenario Editor.
Same Size (Submenu)	Contains menu options for adjusting the height and width of swimlanes to the same size. This command can be used in the Process Definition Editor and the Scenario Editor.

Menu Option	Description
Swimlane Style (Submenu)	Contains menu options for switching between the alignment of the swimlane title (at the top or to the left). This command can be used in the Process Definition Editor and the Scenario Editor.
Expand Selection To	This command can be used with the Source view of the Ajax Page Editor. It selects the selector or property at the position where the cursor is currently located, or just before or after it.
Toggle Insert Mode	Cannot be used.
Content Assist	Can be used with the Source view of the Ajax Page Editor. This command opens a list of input candidates that can be used at the cursor position in the Source view.
Show Tooltip Description	Can be used with the Source view of the Ajax Page Editor. This command displays explanations of properties or tags when the cursor is hovered over them.
Word Completion	Can be used with the Source view of the Ajax Page Editor. This command completes the word input at the cursor position in the Source view.
Quick Fix	Can be used with the Source view of the Ajax Page Editor. This command applies a quick fix to the word at the cursor position in the Source view.
Update	Can be used with the Ajax Page Editor. This command refreshes the display in the Design view.

Source menu

The **Source** menu can only be used with the Source view of the Ajax Page editor.

Menu Option	Description
Toggle Comment	Comments out the selected line or the line containing the cursor.
Add Block Comment	Comments out everything between the start tag and the end tag of the element that contains the cursor.
Remove Block Comment	Cancels a block comment.
Shift Left	Shifts the indentation to the left.
Shift Right	Shifts the indentation to the right.
Cleanup Document	Displays the Cleanup dialog box, which can be used to clean up the document according to rules, such as unifying the case used for tags and attributes.
Format	Formats the content of the editor.
Format Active Elements	Formats the content of the active element.
Occurrences in File	Finds the occurrence locations in the file.

View menu

The **View** menu opens only if a process definition is open.

Menu Option	Description
Zoom In	Increases the magnification of the Process Definition Editor.
Zoom Out	Decreases the magnification of the Process Definition Editor.
Display Mode (Submenu)	Contains menu options for switching between Enhanced View and Classic BPMN View.
Show Details	Turns the display of property symbols (node icons) and Action Annotations on or off.
Show Role	Turns the display of Roles on Activity, Compound Activity, and Voting Activity nodes on or off.

Menu Option	Description
Show Grid	Turns the grid on or off.
Show Ruler	Turns rulers on or off.
Snap to Geometry	Turns feedback lines on or off. Feedback lines help to align elements when dragging.
Palette Settings	Opens a dialog for customizing the palette.

Navigate menu

Menu Option	Description
Display	It is not supported in Systemwalker Runbook Automation Studio

Project menu

Menu Option	Description
Open Project	Open Project
Close Project	Close Project
Built Working Sets >> Select Working Sets	It is not supported in Systemwalker Runbook Automation Studio
Property	Display property of project, process definitions, simulation scenario, QuickForm, resource file, and other relation project.

Window menu

Menu Option	Description
New Window	Opens the new perspective.
New Editor	Opens the process definition, simulation scenario, or resource file that is currently displayed in a new editor.
Open Perspective	Allows you to switch between the **Automated Operation Process Development** perspective, the **Operation Component Development** perspectives and the Debug perspectives.
Show View (Submenu)	Contains menu options to open a view that is not included in the current perspective.
Reset Perspective	Resets the Workbench window to its default settings.
Navigation (Submenu)	Contains menu options for navigating between views, Process Definition Editors, Scenario editors, Resource editors and Ajax Page editors. Also use this submenu for maximizing views, Process Definition Editors, Scenario editors and Ajax Page editors, and for displaying or hiding the system menu and view menus.
Preferences	Opens a dialog where you can set your preferences for working with Systemwalker Runbook Automation Studio. The preferences include appearance of graphical display settings, editor, navigator, startup and shutdown, and server connection settings

Help menu

Menu Option	Description
Welcome to Studio	Displays the Welcome to Studio panel.
Help Contents	Opens the help browser and displays the table of contents for the online Studio User's Guide.
Search	Opens a Help view where you can type the terms you are searching for.

Menu Option	Description
Dynamic Help	Opens a Help view where you can browse for help topics.
About Systemwalker Runbook Automation	Opens a dialog where you can access version information and technical information about Systemwalker Runbook Automation Studio components.

2.2.2 Toolbar

The toolbar contains buttons for frequently used functions.

Button	Description
📄 ▾	Opens a wizard that allows you to create a new project, process definition, scenario, QuickForm, Java Agent, Calendar file, FTP Agent, HTTP Agent, Custom Config, File Listener, Process Scheduler, Decision Table, Process Fragment and a new folder or file.
✓	Validates the process definition, scenario or decision table displayed in the Process Definition Editor, Scenario editor or Decision Table editor respectively, and displays the errors in them in the Problems view.
💾	Saves the process definition, scenario or QuickForm that is being edited.
🖨	Prints the process definition displayed in the Process Definition Editor, or the resource file displayed in the Text editor.
↶	Undoes the last change to a process definition, scenario, or resource file.
↷	Redoes the change that has been previously undone.
	Aligns selected nodes and swimlanes or groups to the left edge, to their center, or to the right edge in the Process Definition Editor.
	Aligns selected nodes and swimlanes or groups to the top edge, to their middle, or to the bottom edge in the Process Definition Editor.
	Adjusts the height and width of swimlanes or groups to the same size in the Process Definition Editor.
⊕	Increases the magnification of the Process Definition Editor.
⊖	Decreases the magnification of the Process Definition Editor.
100% ▾	Sets the magnification of the Process Definition Editor to the specified zooming level.
↳	Allows you to connect nodes with arrows in the Process Definition Editor.
↖	Allows you to select elements in the Process Definition Editor. You can select multiple elements by holding down the <Ctrl> key.
?	Opens the Help Window of *Systemwalker Runbook Automation Studio*
🔦 ▾	Allows you to search for folders and files within your workspace, in selected resources, or in enclosing projects.
⟳	Refreshes the display in the Design view. If it becomes impossible to perform edit operations in the Design

Button	Description
	view because an error has occurred, it is necessary to refresh the view after modifying the source code.
⊟	Displays the Design and Source views with one on top of the other.
◫	Displays the Design and Source views side by side.
🎨	Displays only the Design view.
{≡	Displays only the Source view.
⇧	Sets whether to automatically reflect changes made while editing in the Source view to the Design view. When changes are reflected automatically, the button is displayed as pressed.
🐞 ▾	Start process debug in last using debug consistent.
▶ ▾	Start outside tool process debug in last using debug consistent.

Note

The Process Scheduler function has not been supported in Systemwalker Runbook Automation

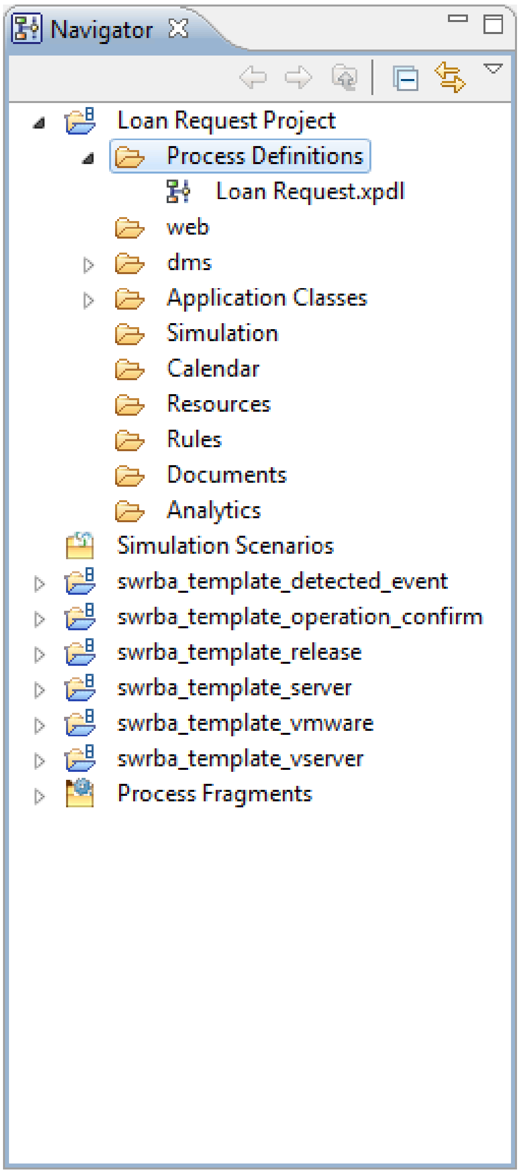
If display Multiple work bench windows, do not open and edit.

2.2.3 Navigator View

The Navigator view shows your Workflow Application and server projects, as well as the Simulation Scenarios project, the file names or names of the process definitions, scenarios, QuickForms, and folders contained in these projects, and Process Fragments.

Use this view to manage your projects, process definitions, and scenarios. For example, you can open process definitions for editing or select them for copying, renaming, exporting, and so on.

Figure 2.2 Navigator View



Navigator

- Loan Request Project
  - Process Definitions
    - Loan Request.xpdl
  - web
  - dms
  - Application Classes
  - Simulation
  - Calendar
  - Resources
  - Rules
  - Documents
  - Analytics
- Simulation Scenarios
- swrba_template_detected_event
- swrba_template_operation_confirm
- swrba_template_release
- swrba_template_server
- swrba_template_vmware
- swrba_template_vserver
- Process Fragments

Navigator View Toolbar

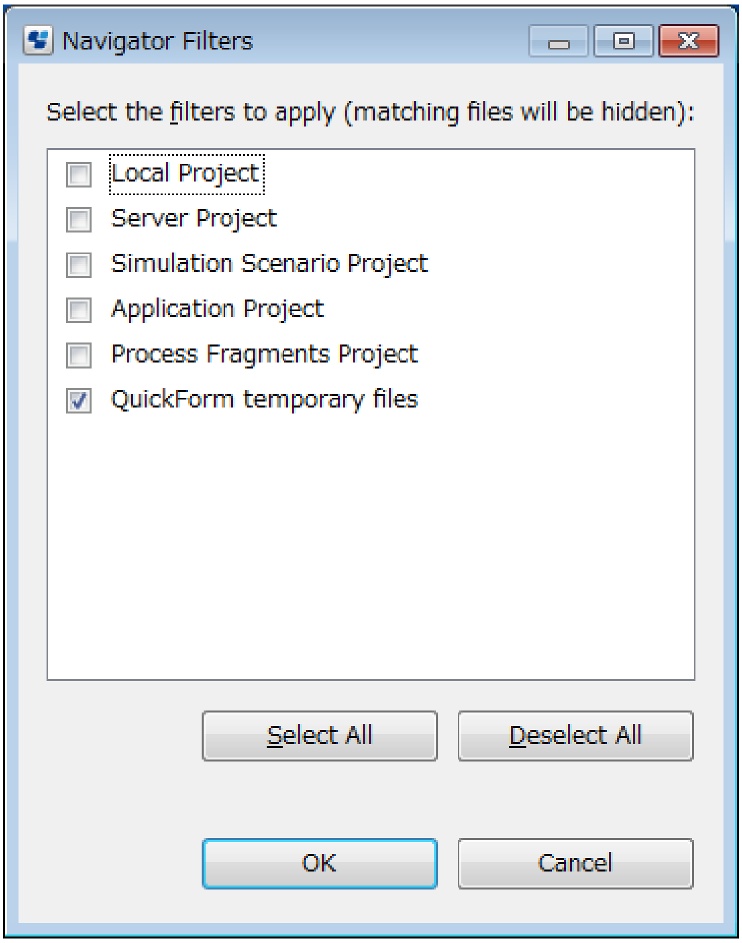
The toolbar of the Navigator view contains buttons for frequently used functions.

Button	Description
⇦	Go back one level in the hierarchy of projects, folders, etc.
⇨	Go forward one level in the hierarchy of projects, folders, etc.
⇧	Go up one level in the hierarchy of projects, folders, etc.
⊟	Collapse all projects.
⇆	Link the selected item with the editor.
▽	Opens a menu for selecting the **Filters** menu option. See below for details.

Filtering Projects

You can filter the projects displayed in the **Navigator** view by selecting **Filters** from the Menu icon in the **Navigator** toolbar. The following dialog is opened:

Figure 2.3 Filtering Projects in the Navigator View



Select the filters to apply (matching files will be hidden):

- □ Local Project
- □ Server Project
- □ Simulation Scenario Project
- □ Application Project
- □ Process Fragments Project
- ☑ QuickForm temporary files

Select the filters to apply to the **Navigator** view. For example, if you select the **Simulation Scenario Project** checkbox, all projects except the Simulation Scenarios project will be listed in the **Navigator** view.

Note

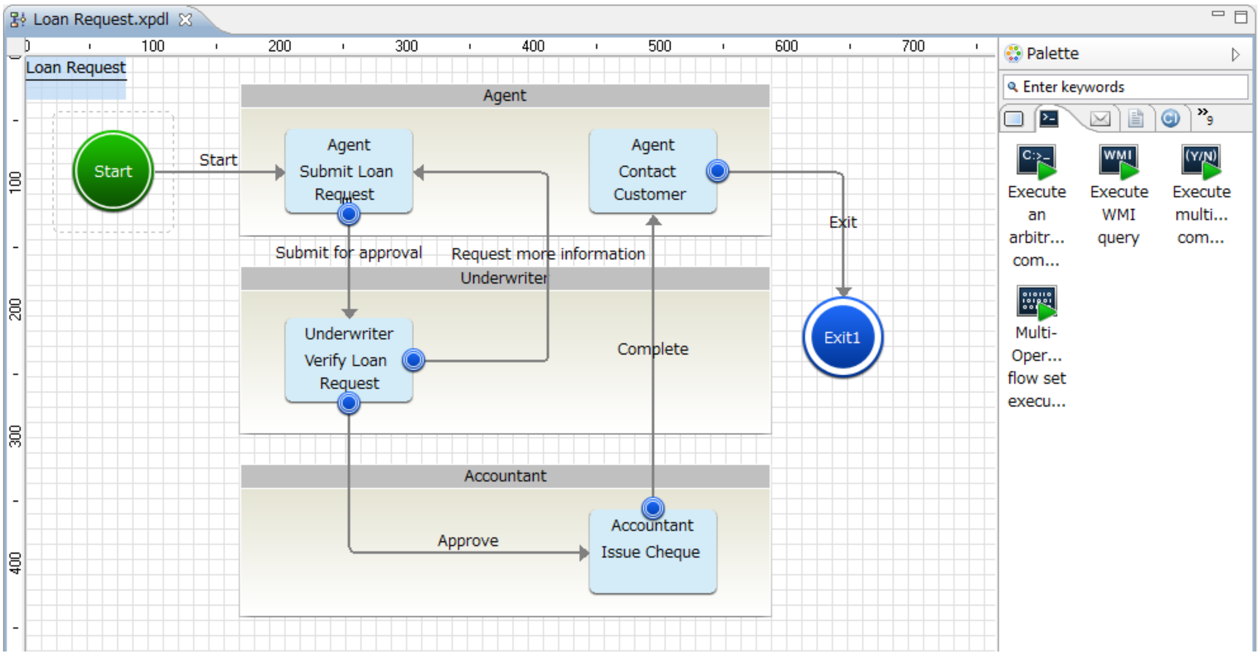
Do not select the **Local Project** check box.

2.2.4 Process Definition Editor

The Process Definition Editor is the main window for process modeling. Use the editor to add and edit all elements that belong to your process definition.

When you open a process definition, the Process Definition Editor is automatically opened. You can open several process definitions in parallel. Each process definition is displayed in a tab. Depending on your preferences settings for the Navigator view, the tab shows the file name or name of the process definition; an asterisk (*) to the left of the file name or name indicates that there are unsaved changes.

Figure 2.4 Process Definition Editor



The Process Definition has a palette with buttons for adding nodes.

- To temporarily open the palette, hover the cursor over the collapsed palette. The palette quickly expands.
- To open the palette and keep it open, click the Show Palette button (<) on the palette.
- To collapse the palette, click the Hide Palette button (>) on the palette.
- You can place the palette on the left-hand side or right-hand side of the Process Definition Editor. Click the palette header and drag it to the desired location.
- To resize the palette, click the border that faces the editing area. Drag the border.

By default, the palette is collapsed on the right side of the Process Definition Editor. You can change the default settings using **Window >> Preferences**. Refer to "2.4.2 Display Settings" for more information.

When you open a process definition that is read-only, the palette is not displayed. The status bar indicates that the file is read-only.

Information

Items can be placed from the palette in the Process Definition Editor, in either of the following two ways.

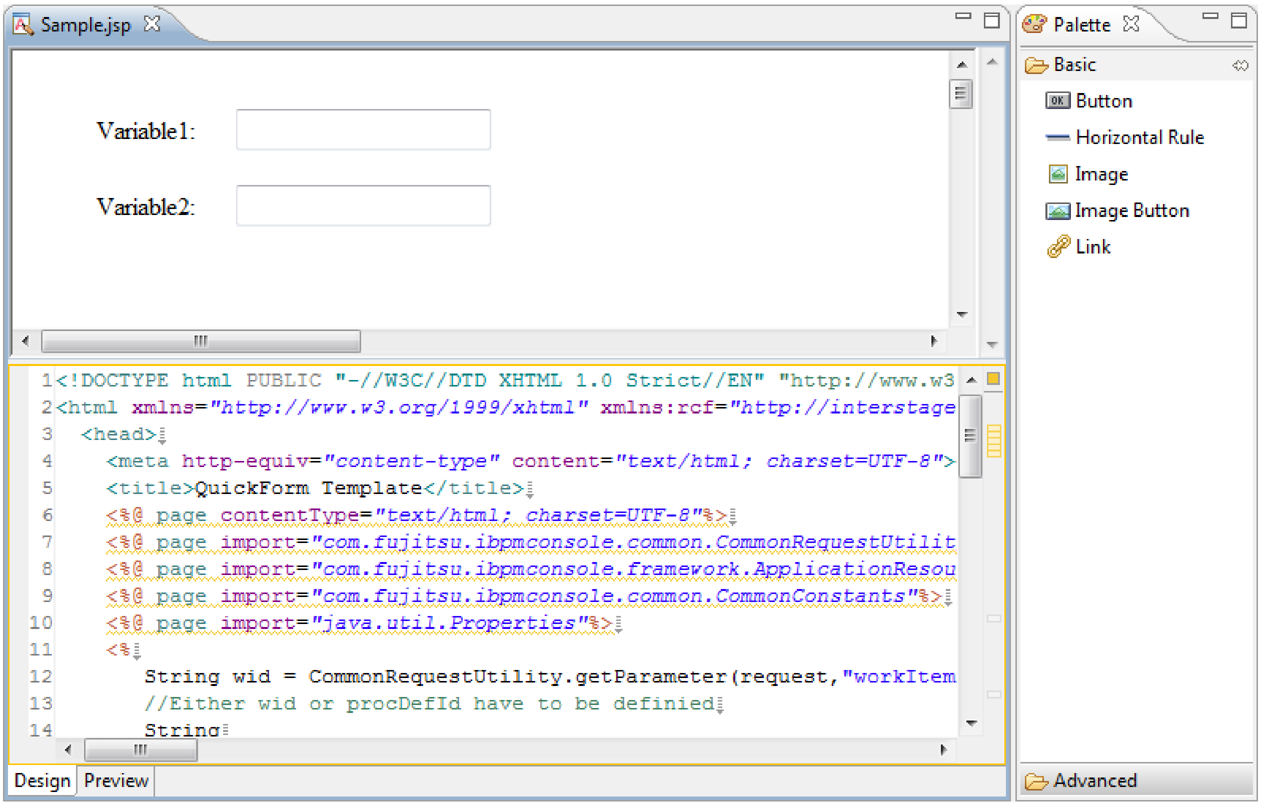
- Drag and drop items from the palette to the Process Definition Editor.
- Select an item in the palette by clicking on it. Then move the mouse cursor to the location in the Process Definition Editor where the item is to be placed, and click again.

2.2.5 Ajax Page Editor and Palette View

The Ajax Page Editor is the main window used to create QuickForms. Use the editor to design QuickForms. When a QuickForm is opened, the Ajax Page Editor opens automatically. Multiple QuickForms can be opened at the same time.

Each QuickForm is displayed in a separate tab. The Quick Forms are displayed in the tabs. Asterisks (*) displayed to the left of a name or file name indicate that there are changes that have not been saved.

Figure 2.5 Ajax Page Editor and Palette view



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:rcf="http://interstage
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>QuickForm Template</title>
    <%@ page contentType="text/html; charset=UTF-8"%>
    <%@ page import="com.fujitsu.ibpmconsole.common.CommonRequestUtilit
    <%@ page import="com.fujitsu.ibpmconsole.framework.ApplicationResou
    <%@ page import="com.fujitsu.ibpmconsole.common.CommonConstants"%>
    <%@ page import="java.util.Properties"%>
    <%
       String wid = CommonRequestUtility.getParameter(request,"workItem
       //Either wid or procDefId have to be defiend
       String
```

Information

Items can be placed from the Palette view in the Ajax Page Editor, in either of the following two ways.

- Select an item in the Palette view by clicking on it. Then move the mouse cursor to the location in the Ajax Page Editor where the item is to be placed, and click again.

2.2.6 Scenario Editor

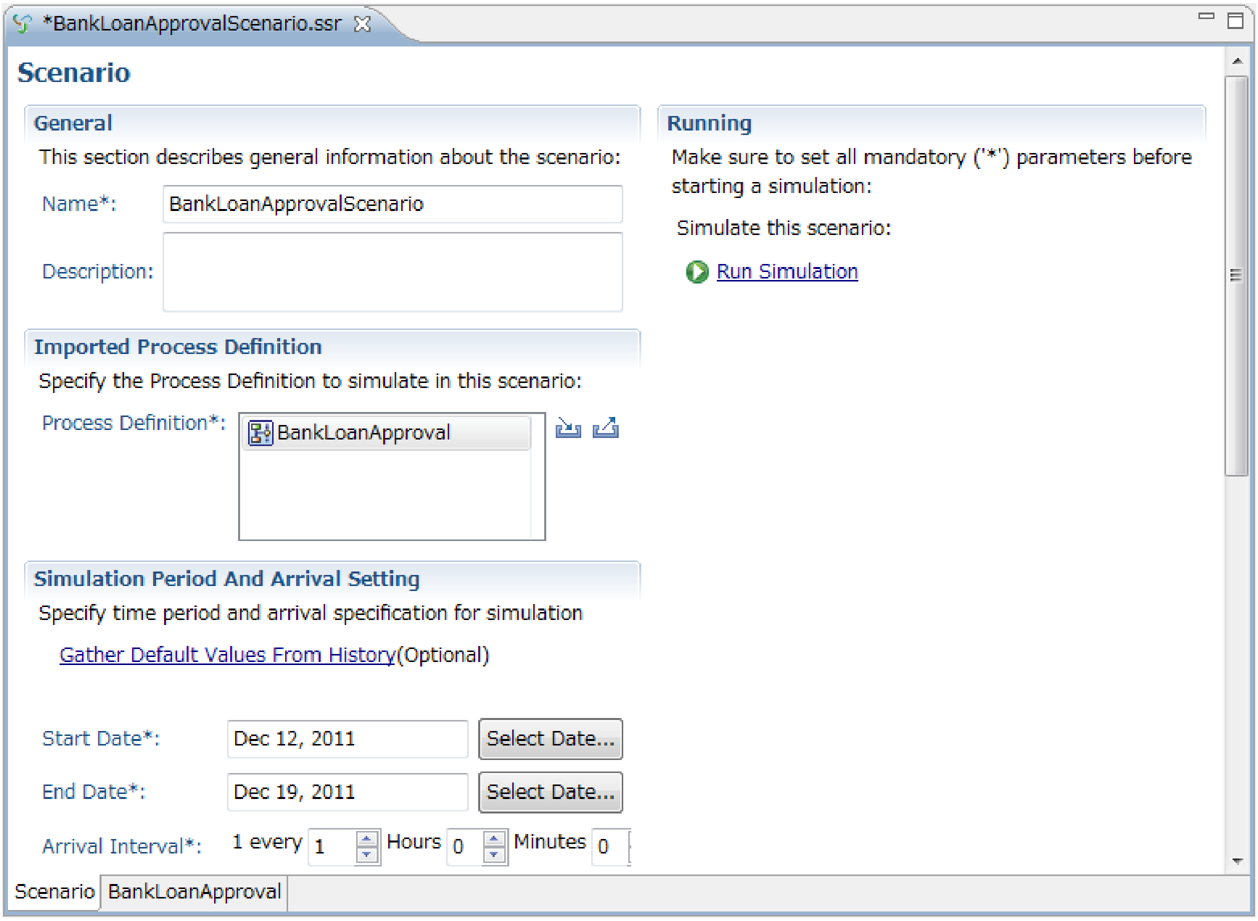
The Scenario editor is the main window for defining a simulation scenario and storing simulation results. Use the Scenario editor for defining simulation properties and display and report simulation results.

Every scenario must import a process definition from either a Workflow Application or a server project. The process definition used by a scenario can, in turn, be exported to a local or server project. In addition, for every activity defined in the process definition, simulation properties can be specified. Use the Scenario editor to add and edit all elements that belong to your process definition.

When you open a simulation scenario, the Scenario editor is automatically opened. You can open several scenarios in parallel. Each scenario is displayed in a tab on top of the Scenario editor. Depending on your preferences settings for the Navigator view, the tab shows the file name or name of the scenario; an asterisk (*) to the left of the file name or name indicates that there are unsaved changes.

Depending on whether an opened scenario already imports an existing process definition, the process definition is also opened in a separate tab that you can access at the bottom of the Scenario editor.

Figure 2.6 Scenario Editor



*BankLoanApprovalScenario.ssr

Scenario

General

This section describes general information about the scenario:

Name*: BankLoanApprovalScenario

Description:

Imported Process Definition

Specify the Process Definition to simulate in this scenario:

Process Definition*: BankLoanApproval

Simulation Period And Arrival Setting

Specify time period and arrival specification for simulation

Gather Default Values From History(Optional)

Start Date*: Dec 12, 2011

End Date*: Dec 19, 2011

Arrival Interval*: 1 every 1 Hours 0 Minutes 0

Running

Make sure to set all mandatory ('*') parameters before starting a simulation:

Simulate this scenario:

The process definition imported by the scenario in the Scenario editor is opened on a separate tab (if it exists), and you can change it, if required: Click the tab labeled with the name of the process definition at the bottom of the Scenario editor to do so. In the sample above, this is the **BankLoanApproval** tab.

The **Gather Default Values From History** link indicates that you can fetch "historical" process definitions from a remote server and use them for simulation.

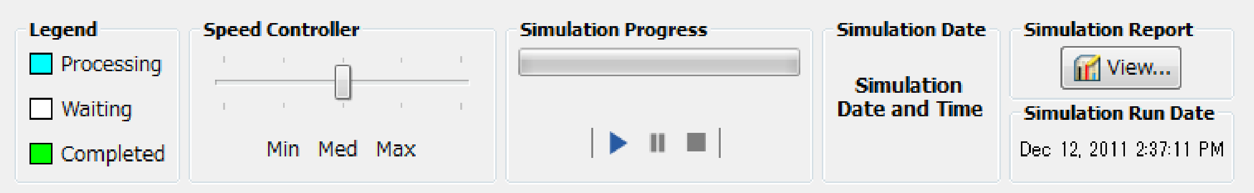
Scenario results are also displayed in the Scenario editor. In this case, the Simulation Controller is displayed below the Scenario editor.

Simulation Controller

The Simulation Controller is used for starting, stopping and pausing a simulation run, and for controlling the speed with which activities are processed and animated during the simulation. In addition, the Simulation Controller also provides a tab for generating simulation reports.

When you have executed **Run Simulation** on a selected simulation scenario, the Simulation Controller is displayed just below the Scenario editor.

Figure 2.7 Displaying the Simulation Controller



Legend

- 🟦 Processing
- □ Waiting
- 🟩 Completed

Speed Controller

Simulation Progress

Simulation Date

Simulation Date and Time

Simulation Report

Simulation Run Date

Dec 12, 2011 2:37:11 PM

2.2.7 Decision Table Editor

Decision Tables allow you to design advanced rules for decision making without programming. Decision Tables use a simple yet powerful table based approach for managing rules dynamically.

Decision Table editor is used to define Decision Table.

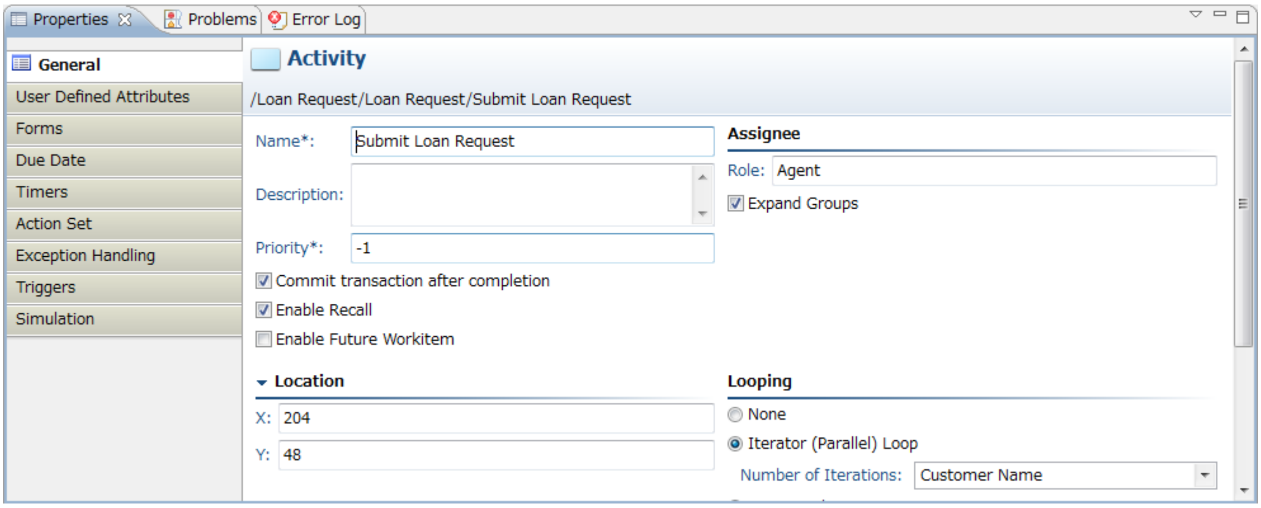
2.2.8 Properties View

The **Properties** view displays all the properties for a process definition or for a selected node and allows you to edit them.

When you select a project, process definition, scenario, folder, QuickForm or decision table in the **Navigator** view, the Properties view usually displays folder or file properties (**File name** setting). Depending on your setting of the preferences for the **Navigator** view, the name of the process definition, scenario, Muckworm or decision table and its description (**Name** setting) may be displayed.

When you click the empty space in the **Process Definition Editor**, the **Properties** view displays the properties of the process definition like its name and description, and information about the **User Defined Attributes**, **Forms**, **Due Dates**, **Timers**, **Triggers**, **Action Set**, **Exception Handling**, **Voting Rules**, **Data Mapping** and **Decisions** associated with it. When you select an element of a process definition, the properties of that element are displayed. As an example, the following figure shows the **Properties** view for an Activity Node.

Figure 2.8 Properties View for an Activity Node



Activity

/Loan Request/Loan Request/Submit Loan Request

Name*: Submit Loan Request

Description:

Priority*: -1

☑ Commit transaction after completion

☑ Enable Recall

☐ Enable Future Workitem

Assignee

Role: Agent

☑ Expand Groups

Location

X: 204

Y: 48

Looping

○ None

● Iterator (Parallel) Loop

Number of Iterations: Customer Name

If a component or tag information item is selected in the Ajax Page Editor, the Properties view displays the properties of that component or tag information item.

Switching the Properties tab display

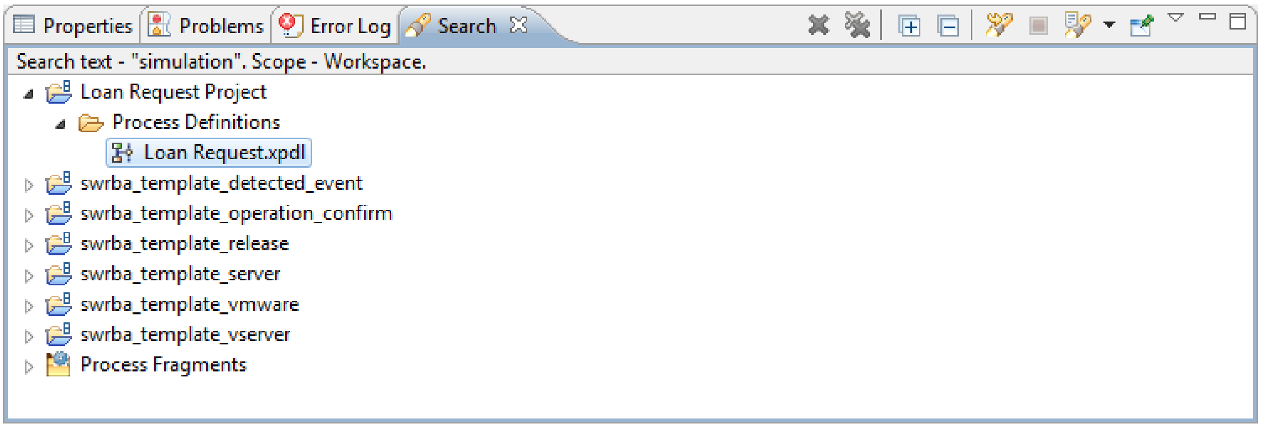
The **Properties** view tab can be switched to display only the icon - this will cause the tab name to be displayed as a tooltip when the icon receives the focus.

2.2.9 Search View

The **Search** view displays search results and provides several functions for analyzing the search results. This view is displayed only after searching for specific items.

The toolbar provides, among others, symbols for displaying selected matches and showing the search history. The following figure shows the **Search** view after searching for the word "simulation":

Figure 2.9 Search View



Search text - "simulation". Scope - Workspace.

- Loan Request Project
  - Process Definitions
    - Loan Request.xpdl
- swrba_template_detected_event
- swrba_template_operation_confirm
- swrba_template_release
- swrba_template_server
- swrba_template_vmware
- swrba_template_vserver
- Process Fragments

2.2.10 Problems View

The Problems view displays the errors for a particular process definition, scenario, decision table or QuickForm.

When you open a process definition and click **Validate** on the toolbar or right click the empty space in the Process Definition Editor and select **Validate** from the pop-up menu, the Problems view displays all the errors relevant to that process definition. Similarly errors in a scenario or a decision table can also be displayed in the Problems view.

In the Process Definition Editor, nodes relevant to errors are emphasized.

In Navigator view, the error icon is added to the resource file that has the error.

If the Problems view is closed, select **Window** >> **Show View** >> **Problems**, to display it. It displays the following information for each error:

Field	Description
Description	Describes the nature of the error.
Resource	Displays the name of the resource where the error occurred.
Path	Displays the name of the project to which the resource belongs.
Location	Displays the precise element in the resource where the error occurred.
Type	Displays the type of the error that occurred.

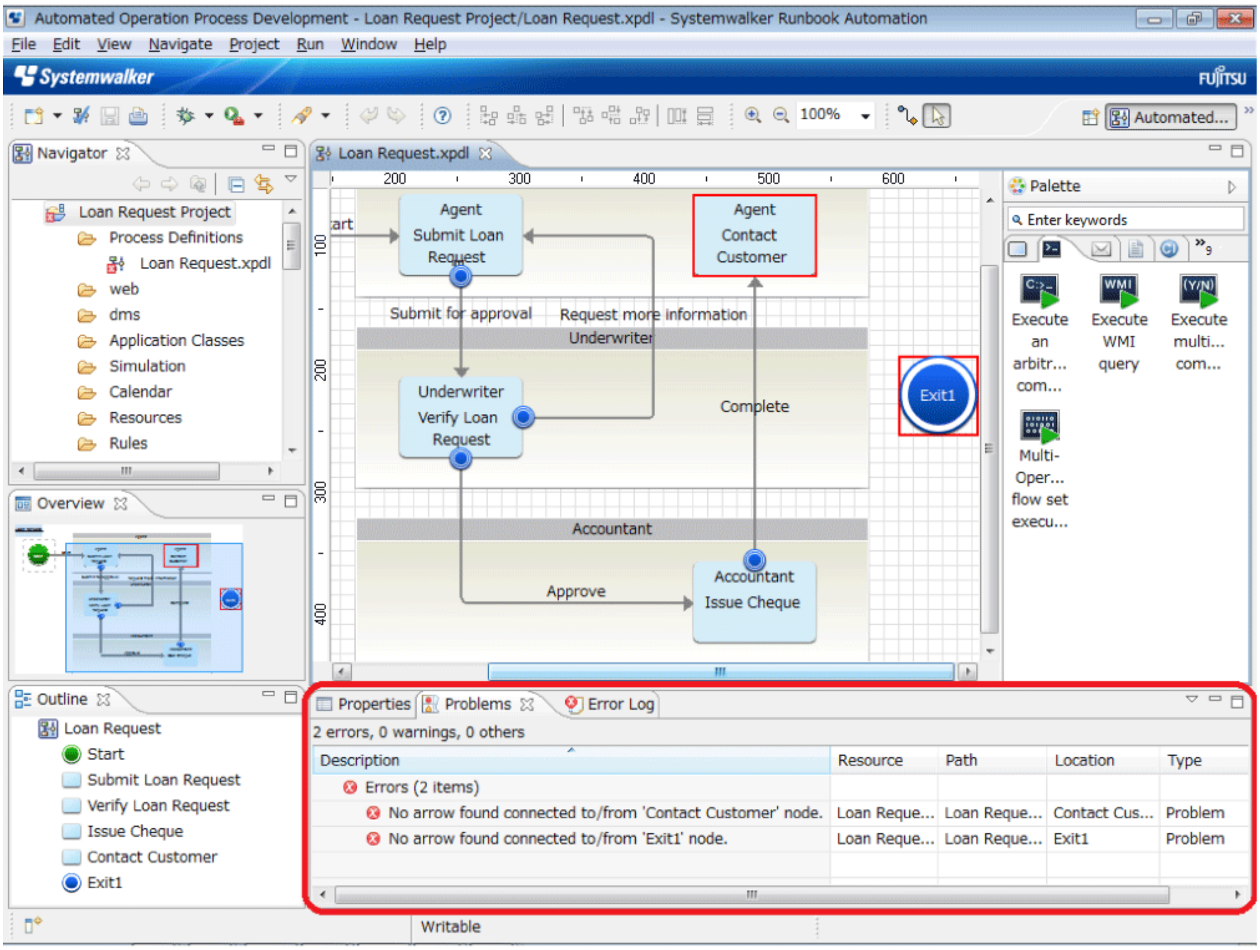
When you double click on a particular row, the precise element in the process definition, where the error occurred gets selected in the Process Definition Editor so that the error can be corrected. Similarly, the precise element in a scenario or Decision Table gets selected in the Scenario editor or Decision Table editor respectively.

Note

In Navigator view, when you copy the resource file having the error icon and create a new resource file, the error icon is not added to the new resource file.

If a process definition file, scenario file or Decision Table file is deleted, then the errors and warnings associated with them are also deleted from the this view.

Figure 2.10 BPM Problems View for a process definition



2 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
No arrow found connected to/from 'Contact Customer' node.	Loan Reque...	Loan Reque...	Contact Cus...	Problem
No arrow found connected to/from 'Exit1' node.	Loan Reque...	Loan Reque...	Exit1	Problem

2.2.11 Error Log View

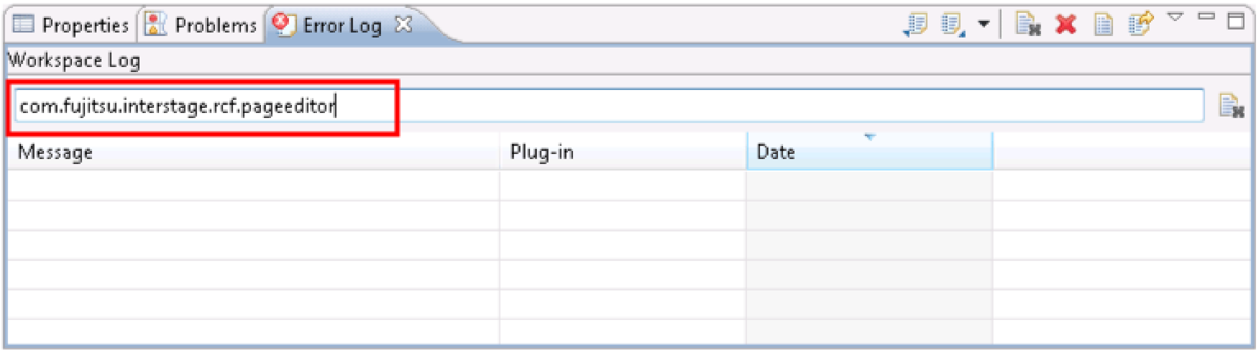
The Error Log view displays errors and warnings for QuickForms.

If the Error Log view is not displayed, select Windows >> View Display >> Error Log to display it. The details of each field are shown below.

Field	Description
Messages	Displays an error or warning message.
Plug-in	Displays the name of the plug-in where an error or warning occurred.
Date	Displays the date when an error or warning occurred.

Note

- Specify " com.fujitsu.interstage.rcf.pageeditor" for the filter.



com.fujitsu.interstage.rcf.pageeditor

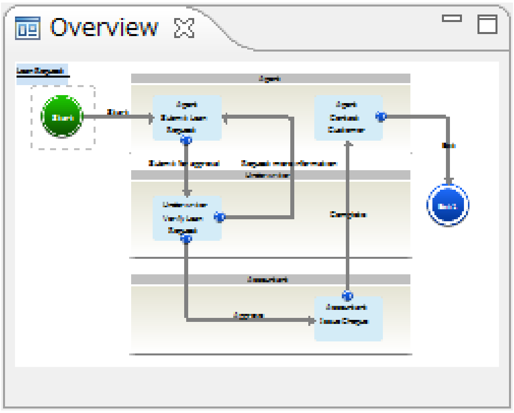
- If the log file was opened using the **Open Log** button of the **Error Log** view, close it before exiting the workbench.

2.2.12 Overview View

The Overview view is useful with large process definitions. It gives an overview of the whole process definition and allows you to quickly navigate to the area you want to work on.

If a process definition is too large to be displayed as a whole in the Process Definition Editor, the Overview view highlights the area that is currently visible. To navigate to another part, move the highlighted area.

Figure 2.11 Overview View



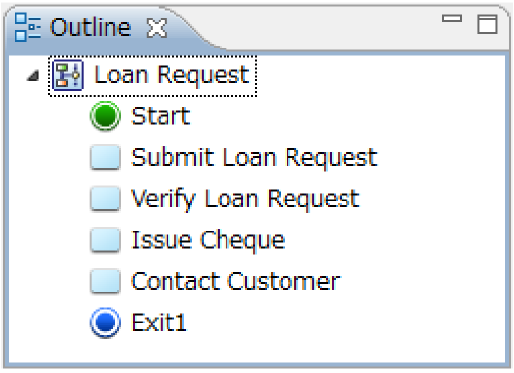
2.2.13 Outline View

The Outline view displays an outline of the active process definition or process fragment. It displays all nodes the process definition or fragment is made of.

Using the Outline view, you can quickly navigate to particular nodes in large process definitions or fragments. You can also use this view for node operations like cutting, copying, and pasting, and for defining node properties.

Nodes are represented by an icon and the node name. Refer to section 1.3.1 Nodes for an explanation of the icons. Note that any annotations to Actions or Nodes are not visible in Outline view.

Figure 2.12 Outline View



Outline

- Loan Request
  - Start
  - Submit Loan Request
  - Verify Loan Request
  - Issue Cheque
  - Contact Customer
  - Exit1

If a simulation scenario is active, the Outline view only displays the scenario name. Similarly, if the Ajax Page Editor is active, a list of the components in the Ajax Page Editor will be displayed in the Outline view.

2.2.14 Pop-Up Menu

Most elements have a pop-up menu associated with them that contains frequently used functions. The contents and availability of the pop-up menu depends on the type of element. To display the pop-up menu, select an element in a view, in the Process Definition Editor, in the Scenario editor, or in the Form editor, and right click.

2.2.15 Displaying Properties

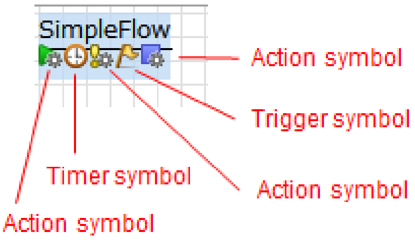
Properties help you specify process definitions, nodes and arrows (for more information on properties and their symbols, refer to section 1.3.2 Property Symbols).

By default, properties are automatically displayed in the Process Definition Editor. When you modify the properties of a process definition or node through the **Properties** view, Systemwalker Runbook Automation Studio automatically checks if there are any properties assigned. If there are, the property symbols are displayed.

Property symbols are put in different places, depending on whether they are assigned to a process definition, a node or an arrow:

- **Process Definition Symbols**: Process-related symbols are displayed on the top-left of the Process Definition Editor. The name of the process definition will be underlined. The following example shows a process definition for which Java Actions, a timer, Error Actions, and a trigger have been defined:

Figure 2.13 Displaying property symbols for process definitions

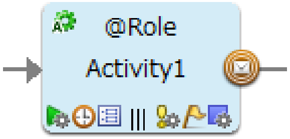


In the example, the property symbols are displayed in the following order (from left to right):

No.	Symbol	Meaning
1	🚩⚙	Owner actions or initializing actions are defined in the Process Definition.
2	🕒	A timer has been defined for the process definition.
3	⚙	One or more Java Actions have been defined for dealing with errors during process execution.
4	🚩	A trigger has been defined for the process definition.
5	⚙	Commit actions are defined in the Process Definition.

- **Node symbols**: Node-related symbols are placed within a node. The following example shows an Activity Node to which a Java Agent has been assigned. All properties provided in the Activity Node are set.

Figure 2.14 Displaying property symbols for an Activity Node



In the example, the property symbols are displayed in the following order (from top left to bottom right):

1	⚙	This symbol indicates that you have assigned a Java Agent to the Activity Node.
2	▶	This symbol indicates that at least one Role or Prologue Action has been defined for the Activity Node. This Java Action is executed when the Activity Node is initialized.
3	🕒	This symbol indicates that a due date or timer has been defined for the Activity Node.
4	☰	This symbol indicates that one or more Forms have been defined for the Activity Node.
5	⚙	This symbol indicates that an Error Action has been defined for the Activity Node.
6	⚑	This symbol indicates that one or more triggers have been defined for the Activity Node.
7	⚙	This symbol indicates that at least one Epilogue Action has been defined for the Activity Node. This Java Action is executed when the Activity Node is completed.
8	|||	This symbol indicates that the Iterator (Parallel) Loop has been defined for the Activity Node.
9	↻	This symbol indicates that the Sequential Loop has been defined for the Activity Node.

Note

Java Action symbols for nodes are displayed only if you have defined Prologue or Role Actions (for initializing an activity), and Epilogue Actions (for completing an activity).

2.2.16 Displaying or Hiding Rulers

Rulers help you align elements in the Process Definition Editor. They use pixels as measurement unit.

By default, rulers are displayed. You can change this setting for any open Process Definition Editor. You can also set your preference for all Process Definition Editors that you open in the future.

To display or hide rulers for an open Process Definition Editor:

1. Click the Process Definition Editor to make it active.

2. Do one of the following:

    - To hide rulers, select **View** and uncheck **Show Ruler**.

    - To display rulers, select **View** and check **Show Ruler**.

To set your ruler preference, select **Window** >> **Preferences** and then **Interstage BPM Studio** >> **Editor**.

2.2.17 Displaying or Hiding the Grid

The grid helps you align elements in the Process Definition Editor.

By default, the grid is displayed. You can change this setting for any open Process Definition Editor. You can also set your preference for all Process Definition Editors that you open in the future.

To display or hide the grid for an open Process Definition Editor:

1. Click the Process Definition Editor to make it active.
2. Do one of the following:
    - To hide the grid, select **View** and uncheck **Show Grid**.
    - To display the grid, select **View** and check **Show Grid**.

Refer to "2.4.2 Display Settings" for information on how to set your grid preference.

2.2.18 Displaying or Hiding Roles

By default, the Role that is supposed to complete an activity is displayed on the Activity or Compound Activity or Voting Activity Node. You can hide Roles so that they are not displayed on the nodes.

You can change this setting for any open Process Definition Editor. You can also set your preference for all Process Definition Editors that you open in the future.

To change role settings for an open Process Definition Editor:

1. Click the Process Definition Editor to make it active.
2. Do one of the following:
    - To hide Roles, select **View** and uncheck **Show Role**.
    - To display Roles, select **View** and check **Show Role**.

To set your preference for the display of Roles in Activity Nodes, select **Window** >> **Preferences** and then **Interstage BPM Studio**.

2.2.19 Changing Display Mode

Systemwalker Runbook Automation Studio supports two display modes for the Process Definition Editor:

- Classic BPMN View

  Elements in the Process Definition Editor are shown in black and white only.

- Enhanced View

  Elements in the Process Definition Editor have different colors.

By default, the Enhanced View is set. You can change this setting for any open Process Definition Editor. You can also set your preference for all Process Definition Editors that you open in the future.

To change display mode for an open Process Definition Editor:

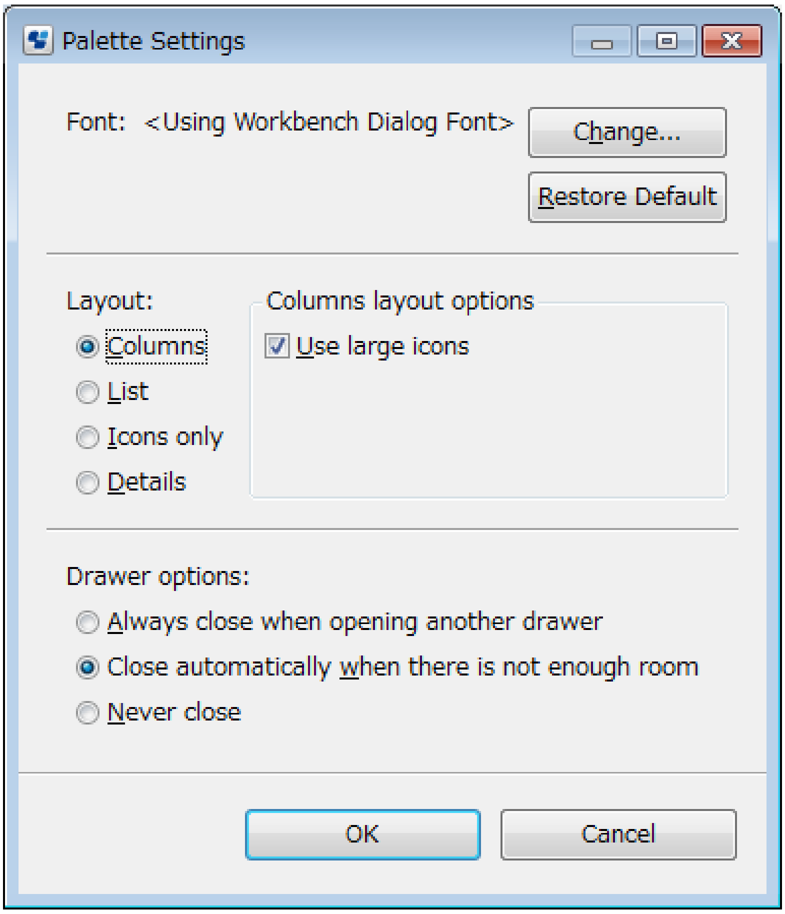
1. Click the Process Definition Editor to make it active.
2. Select **View** >> **Display Mode**. Select the display mode you want to use.

To set your preferred display mode, select **Window** >> **Preferences** and then **Interstage BPM Studio**.

2.2.20 Modifying the Palette Settings for the Process Definition Editor

You can change the appearance of the palette for each Process Definition Editor individually. You can also set preferences for all Process Definition Editors that you open in the future.

Figure 2.15 Palette Settings



Font: <Using Workbench Dialog Font>

Layout:

● Columns
○ List
○ Icons only
○ Details

Columns layout options

☑ Use large icons

Drawer options:

○ Always close when opening another drawer
● Close automatically when there is not enough room
○ Never close

To change the settings for Palette:

Font:

1. To change the **Font** settings for the icon texts:
    a. Right click the palette and select **Settings** from the pop-up menu.
    b. Click **Change**.
    c. Change font properties like font family or font size.
    d. Close the dialog by clicking **OK**.

Layout:

1. To change the palette **Layout**, select the desired layout, for example **Icons only**.
2. For every layout, you can choose between large and small icons be checking or unchecking the **Use large icons** check box.

Drawer options:

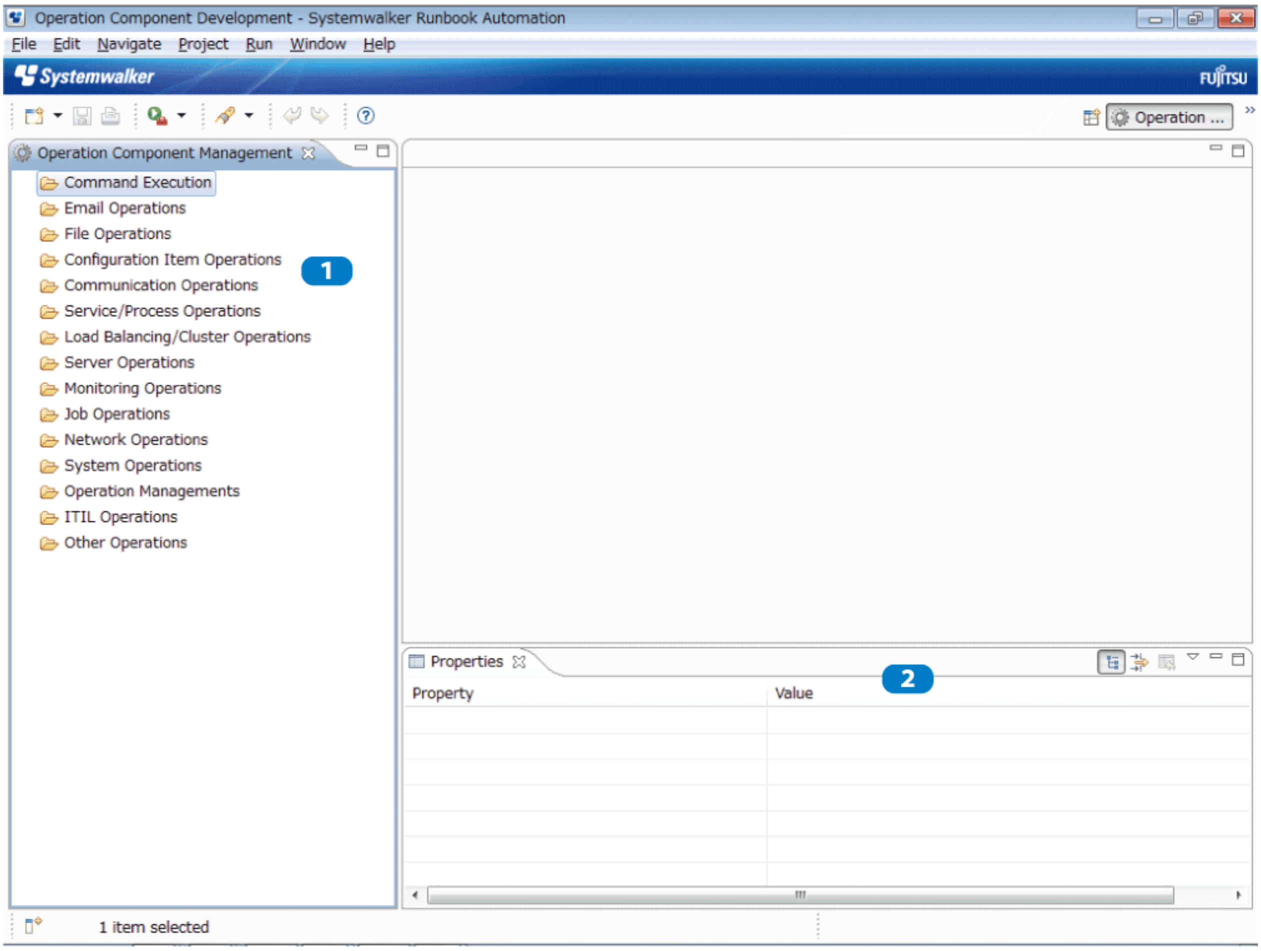
- The palette is opened in a tab rather than a drawer, so these options are ignored.

You can set palette preferences for Process Definition Editors that you open in the future. Select **Window** >> **Preferences** and then **Interstage BPM Studio** >> **Editor**.

2.3 Workbench Window for the Operation Component Development Perspective

The **Operation Component Development** perspective provides ways of creating and managing operation components. This perspective consists of a menu bar, a toolbar, several views, and a text editor. These components are collectively referred to as the "workbench window". The following screenshot shows the workbench window:

Figure 2.16 Workbench window for the Operation Component Management perspective



The numbers in the screenshot indicate the following screen components:

- 1: **Operation Component Management** view
  This is used for operations such as the creation of new operation component projects.

- 2: **Properties** view
  This displays information about the selected item.

2.4 Configuring Systemwalker Runbook Automation Studio

This section explains the following settings:

- Configurations for the Process Definition Editor

- Display method for process definition elements.

2.4.1 Server Connection Settings

Use the following procedure to set or change the server connection information used for Systemwalker Runbook Automation Studio:

1. Select **Window** >> **Preferences**.

   The **Preferences** dialog box is displayed.

2. Click **Server connection settings** for **Interstage BPM Studio**.

   Server connections is displayed in the **Server connection settings** page.

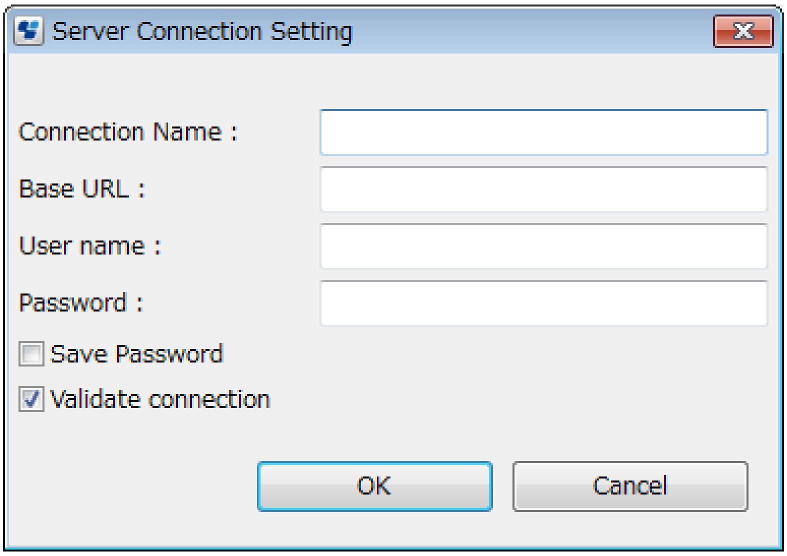
3. Use the following procedure to add new server connection information to the list or to edit server connection information already in the list:

   1. Click **New** or **Edit**.

      The **Server Connection Setting** dialog box opens.

      Enter the parameters for the server connection to be added or updated.

      Figure 2.17 Server Connection Setting

      Server Connection Setting

      Connection Name :

      Base URL :

      User name :

      Password :

      □ Save Password

      ☑ Validate connection

      Enter the following parameters:

      - **Connection Name**: The specific server connection name.

      - **Connection Information**: The base URL is the URL used to connect to the server. Specify the base URL using the following format:

        http://<hostname>:<port>

        The following is an example URL where the hostname is "rbaserver" and the port number is "80".

        http://rbaserver:80

        If the port number is omitted, it is assumed that "80" is specified.

      - **User name**: The user name that is authenticated by Systemwalker Runbook Automation.

      - **Password**: The password associated with the user name.

      - **Save Password**: Check to save the password.

      - **Validate connection**: Check to verify the connection when **OK** is clicked.

        If a parameter is invalid and connection to the server is not possible, for instance, the server stops, an error message is displayed and it will not be possible to set the server connection information. In this case, clear the check box to set the server connection information.

   2. Click the **OK** button.

4. Click **Apply** in the **Server connection settings** page.

2.4.2 Display Settings

This section describes how to change the settings for the following items:

- Layout of the Process Definition Editor
- Process Definition element display

These settings become effective for all Process Definition Editors that you open in the future. Process Definition Editors that are currently open are not affected.

Information

You can change some settings for currently open Process Definition Editors using the **View** menu and the pop-up menu of the Palette.

In addition to the above, the following items can be set:

- Display mode for the Navigator view
- Whether the names or file names of local process definitions or scenarios are displayed
- Defining which editor is to be used for opening a process definition, simulation file, or resource file
- Messages that appear when starting and exiting Systemwalker Runbook Automation Studio

To set your preferences:

1. Select **Window** >> **Preferences**.

   The **Preferences** dialog box is displayed.

2. To set your preferred **Display Mode** for elements in the Process Definition Editor:

   a. Click **Interstage BPM Studio**.

   b. Choose between Enhanced View (elements have different colors) and Classic BPMN View (elements are black and white only).

3. To set your preference for the display of property symbols in nodes, and to define how many of the defined Java Actions are displayed in a Java Action annotation.

   a. Ensure that the **Interstage BPM Studio** page is open.

   b. Check or uncheck the **Show Details** check box.

   c. Enter the desired number of Java Actions to be displayed in the annotation for Java Actions.

4. To set your preference for the display of Roles on Activity nodes:

   a. Ensure that the **Interstage BPM Studio** page is open.

   b. Check or uncheck the **Show Role** check box.

5. To set your preferred position of swimlane titles:

   a. Expand **Interstage BPM Studio** and click **Appearance**.

   b. To place the title on the left-hand side, select **Left**. To place the title on the upper side, select **Top**.

6. To set your preferences for the palette:

   a. Expand **Interstage BPM Studio** and click **Editors**.

   b. Specify your preferences for the width and position of the palette. Choose whether the palette is collapsed or open (**Lock Palette**).

   c. Click **Palette Settings** and specify your preferences for the layout and drawers.

7. To set your preferences for the grid:

   a. Ensure that the **Editor** page is open.

  b. Choose whether to display or hide the grid (**Show Grid**).

  c. In the **Grid horizontal spacing** and **Grid vertical spacing** fields, type the distance in pixels between grid points.

     The lower the value, the more grid points you see in the Process Definition Editor.

8. To set your preferences for the rulers:

   a. Ensure that the **Editor** page is open.

   b. Choose whether to display or hide the rulers (**Show Ruler**).

9. To set your preferences for alignment support:

   a. Ensure that the **Editor** page is open. Choose whether to turn **Snap to Geometry** on (turns on feedback lines) or off (turns off feedback lines).

   b. Feedback lines help to align elements when dragging.

10. To set your preferences for the Navigator view:

    a. Click **Navigator**.

    b. Choose whether to display the names or the file names of process definitions that are stored in local projects, or the names or the file names of simulation scenarios.

11. To set the Process Outline Editor preferences:

    a. Click **Process Outline editor**.

    b. Set the column information that will be displayed when the Process Outline Editor is opened.

    Note

    This setting is applied to all the Process Outline Editors that are currently open.

12. To set your preferences for the editor you want to use for opening a file:

    - Select **General** >> **Editors**.

      When **File Associations** is clicked, the **File Associations** page displays all the available file types.

    - Select a file type, and assign it to an editor from the **Associated editors** list.

13. To set your preferences for messages that appear when starting and exiting Systemwalker Runbook Automation Studio:

    a. Select **General** >> **Startup and Shutdown**.

    b. Change the settings as required.

14. To set the color used to emphasize nodes with syntax errors:

    a. Select **General** >> **Appearance** >> **Colors and Fonts**.

    b. Open the **Basic** tree

    c. Select **Error text color**.
       The button that indicates the current color is displayed.

    d. Click the button to open the **Color** dialog box.

    e. Select the color to be set, and click **OK**.

Chapter 3 Managing Projects

Projects are containers that help you organize your process definitions, resource files, forms, simulation scenarios, attachments, etc. On file system level, a project corresponds to a folder. As a default, projects are stored in your workspace, but you can store them in any location that is accessible from your computer.

Systemwalker Runbook Automation differentiates the following types of project:

- **Workflow Application project**: Systemwalker Runbook Automation Studio allows you to design entire workflow applications offline. Process definitions, for example, are created and designed on your local machine. You can store just one or several related process definitions in a Workflow Application project. Workflow applications that have been designed on the local computer can then be registered with the Systemwalker Runbook Automation Management Server. A Workflow Application project is represented in the file system by a single directory. It is used to structure the components that make up an application: process definitions, resources, forms, attachments, icons, simulation scenarios, etc.
- **Scenario project**: Project containing scenarios for locally simulating the execution of a process definition. A scenario is stored locally on your machine and can import either a local process definition or a process definition located on a Management Server. By default, Systemwalker Runbook Automation Studio shows a project named "Simulation Scenarios" in the **Navigator** view after installation. You can add your local scenarios in this project, or directly in a Workflow Application project, which contains a default folder named "Simulation". In addition, you can also use previously created process definitions ("historical values") for simulation. Refer to section Chapter 12 Simulating Processes for details.
- **Server project**: Project displaying the process definitions stored on a Management Server. Before you can work with these process definitions, you need to log in to the Management Server as a Systemwalker Runbook Automation user. You can transfer any local process definition to the server or export one from the server to your local machine.

This chapter explains how to create projects and work with them.

3.1 Managing Workflow Application Projects

This section explains the functions available for managing Workflow Application projects.

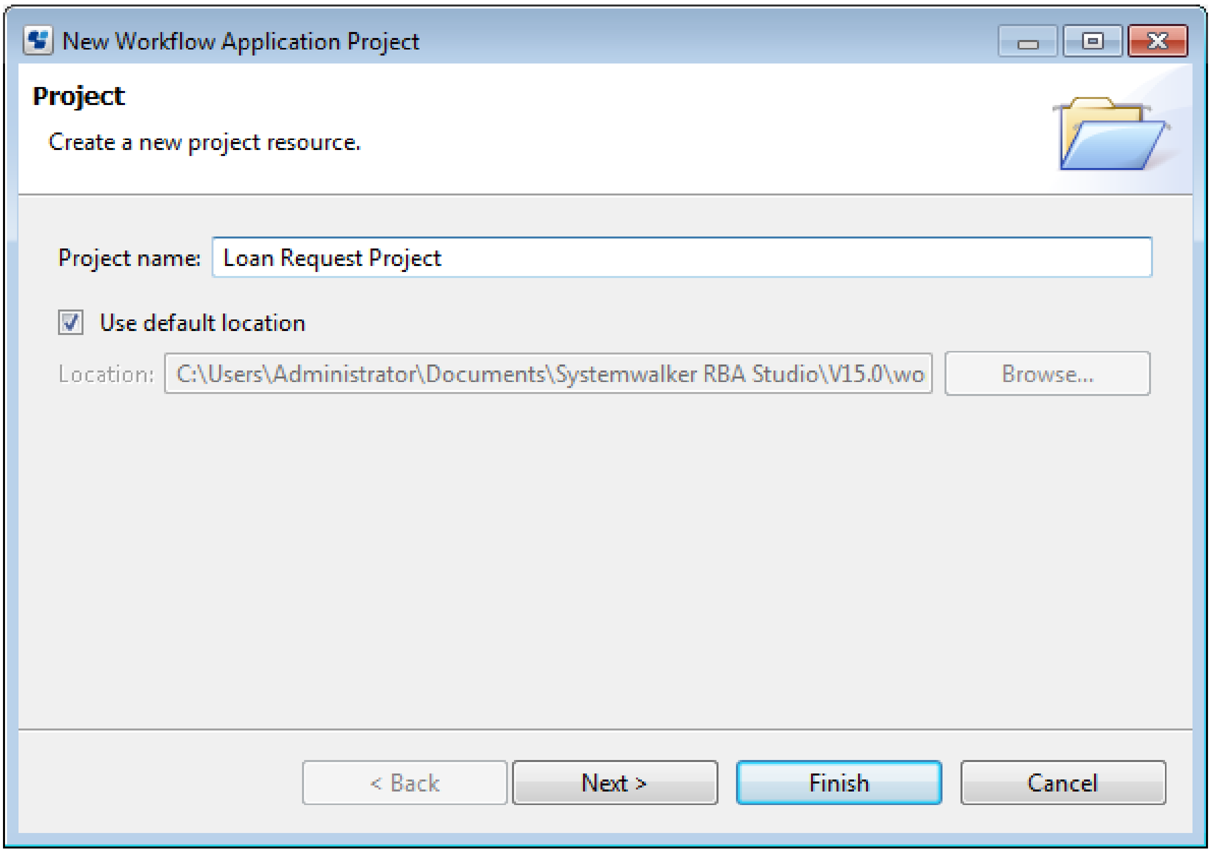
3.1.1 Creating Workflow Application Projects

To create a Workflow Application project on your local machine:

1. Select **File** >> **New** >> **Application**.
2. In the **Project name** field, type a name for your project. This is usually the name of your application.

3. The project that you create corresponds to a folder in the file system. As a default, the project is created in your workspace.

Figure 3.1 Creating a Workflow Application Project



New Workflow Application Project

Project

Create a new project resource.

Project name: Loan Request Project

☑ Use default location

Location: C:\Users\Administrator\Documents\Systemwalker RBA Studio\V15.0\wo

4. If you want to change the location of your project:

   a. Clear the **Use default location** check box.

   b. Click **Browse**. Select an existing folder, or create a new one. Click **OK**.

5. Click **Next**.

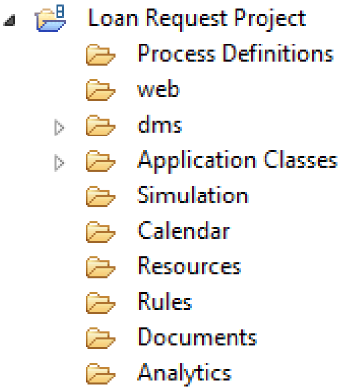
   Type in the required information:

   - Description: Provide a brief description of your project.
   - Owner: This setting is not required.

6. Click **Finish**.

   The new Workflow Application project is listed in the **Navigator** view. The folders are still empty.

Figure 3.2 Creating a new Workflow Application project

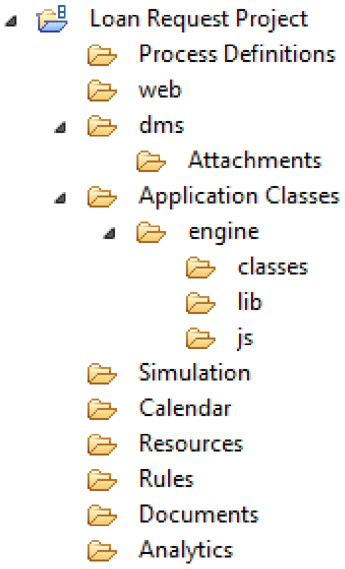


```
Loan Request Project
    Process Definitions
    web
    dms
    Application Classes
    Simulation
    Calendar
    Resources
    Rules
    Documents
    Analytics
```

3.1.2 Components of a Workflow Application Project

By default, the following folder structure is created for a Workflow Application project:

Figure 3.3 Workflow Application Project Folder Structure



- Loan Request Project
  - Process Definitions
  - web
  - dms
    - Attachments
  - Application Classes
    - engine
      - classes
      - lib
      - js
  - Simulation
  - Calendar
  - Resources
  - Rules
  - Documents
  - Analytics

The default folder structure contains everything necessary for managing your Workflow Application project. Your Workflow Application project folder contains the following components:

- **Process Definitions**: This is where you store the process definition(s) of your application.
- **web**: This is where you store all materials that are to be made available through the Web Console. The files placed in this folder are made available on the Web. For example, all your QuickForms associated with the process definitions of your Workflow Application project, any help pages, etc. should be stored in this folder. You can create additional folders beneath the **web** folder that will also be exposed to the web, such as subfolders for images and styles.
- **dms**: This is where you store, for example, any attachment that is part of your Workflow Application project. For this purpose, there is a predefined **Attachments** folder where you can create additional folders, if required.

  The dms folder will, together with any subfolders, be copied to the corresponding folder on the Management Server when your application is installed. You can create additional folders beneath the dms folder.
- **Application Classes**: This is where you store class files and jar files used by your application.
- These files are stored in the engine subfolder. Java Script files are stored in js folder. js folder too is a subfolder of the engine folder.
- **Simulation**: This is where you store the simulation scenarios for your application. This folder includes scenario files (provide information about process simulation parameters) and simulation result files (provide information about simulation results). Both scenario and simulation result files use the .ssr file extension. For more information on simulating processes, refer to section 12.3 Executing a Simulation Scenario.
- **Calendar**: This is where you store the calendar (.cal) files of your application. For more information on business calendars, refer to section 6.22.4 Creating Your Own Business Calendars.
- **Resources:** This is where you store resource files as listed below. For more information on these resource files, refer to section 3.1.6 Working With Resources. The resource files are as follows:
  - Configuration files for Java Action Agents (refer to 11.10 Defining Java Agents)
  - FTP Agent files
  - HTTP Agent files
  - Custom Config files
  - Java agent files

    - File listener files

- **Rules**: Rules folder contains Rules Set folder and Rules Set folder further contains Decision Tables. Refer to Chapter 10 Decision Tables for more information about Rules Set.

- **Documents**: The Process Definition reports are stored in this folder. Refer to 4.8 Using Process Definition Reports.

- **Analytics**: This is the Analytics function folder. Systemwalker Runbook Automation Studio does not use the Analytics function.

The Cost Application project in the sample above is the application root folder. All components of the application are contained in this folder. The contents of this folder can be zipped into one file which is used for transferring the application from one machine to another. For a complete overview of all components of a Workflow Application project, refer to section Appendix B Project Components.

Note

To specify the skin or logo for an application in the Web Console, the following operations must be executed in Studio.

1. Create a "styles" folder under the "web" folder in the application project.

2. Import the style sheet (*.css) files to be used with the skin settings for the application to this "styles" folder.

3. Create an "images" folder under the "web" folder in the application project.

4. Import the image files to be used with the logo settings for the application into this "images" folder.

3.1.3 Using Folders

In any Workflow Application project, you can create additional folders or structure the existing folders to your needs to reflect the hierarchy of your application.

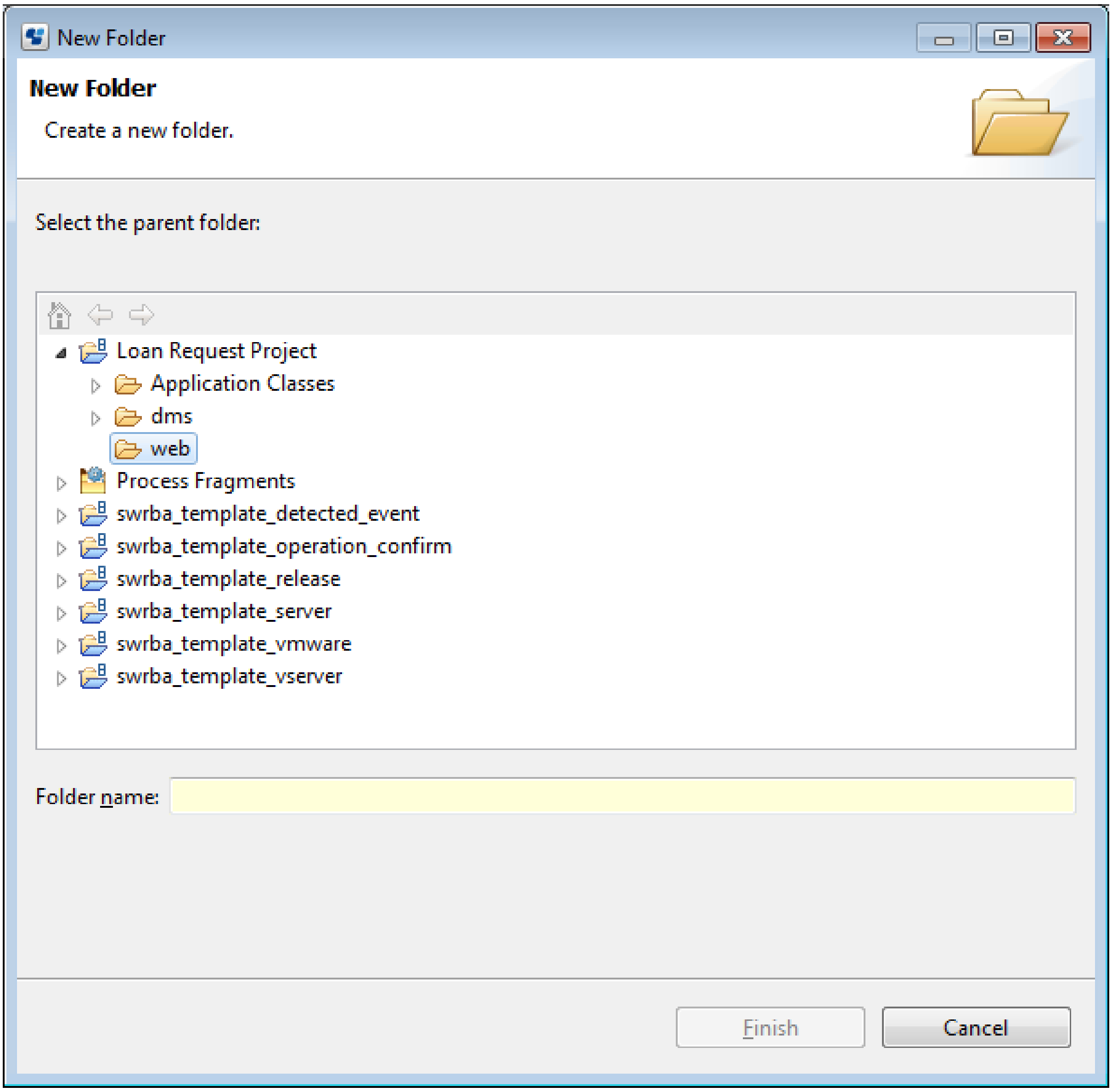
Note

You can create new folders beneath the default **dms**, **web**, and **classes** folders.

To create a folder:

1. Right click the Workflow Application project folder beneath which you want to create a subfolder in the **Navigator** view. Select **New** >> **Folder** from the pop-up menu.

Figure 3.4 Creating a new folder



2. Enter a name for the new folder, and click **Finish**.

You can proceed with importing existing folders and files from your file system into the folder you have just created. Refer to section 3.1.4 Importing Folders and Files for details.

In addition, you can use the default functions for folders that are also available for standard file system folders, such as copy and paste, rename, and delete. Note that these functions are only available for folders that you created, not for the predefined ones.

3.1.4 Importing Folders and Files

This section explains the procedure for importing folders or files from a file system

Prerequisite:

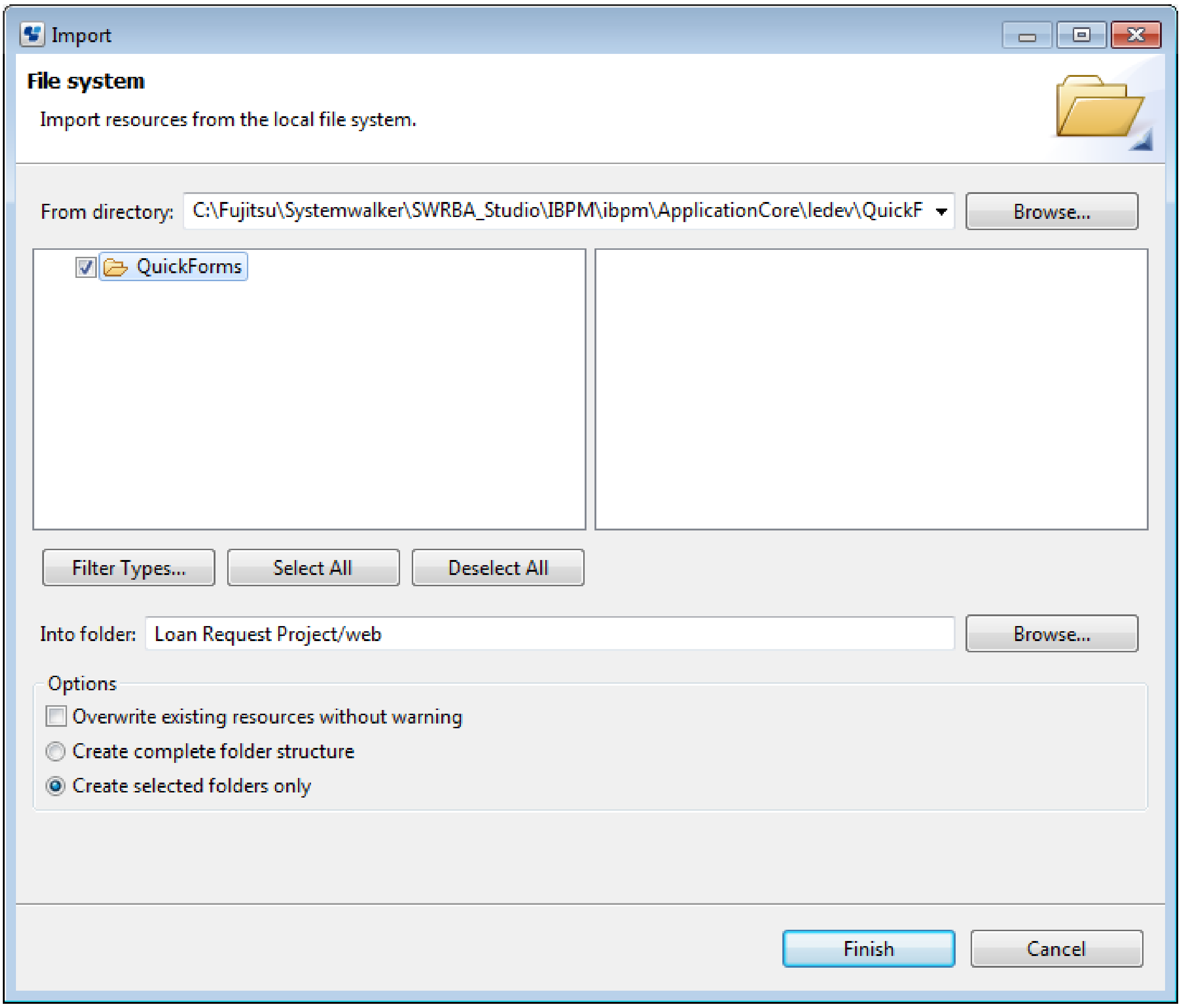
- You have created a Workflow Application project where you can import folders and files.

Note

Existing folders and files can only be imported into the default **web** folder or the **dms** folder and their subfolders.

1. In the **Navigator** view, right click the Workflow Application project folder where you want to import the files. Select Import from the popup menu. In the following example, two QuickForms are to be imported to your workspace:

   Figure 3.5 Importing Folders and Files

   File system

   Import resources from the local file system.

   From directory: C:\Fujitsu\Systemwalker\SWRBA_Studio\IBPM\ibpm\ApplicationCore\ledev\QuickF

   ☑ QuickForms

   Into folder: Loan Request Project/web

   Options

   □ Overwrite existing resources without warning

   ○ Create complete folder structure

   ● Create selected folders only

2. Click **Browse** and navigate to the directory where the folders and files are stored that you want to import in your Workflow Application project. Select an existing folder and click **OK**.

3. Select the folders and files to be imported.

4. The **Into folder** field shows the path of the folder you selected when starting the import function. You can change this path if you want to import the folders and files into a different project or folder of your Workflow Application project.

5. Decide whether you want to overwrite existing folders and files without warning, and whether your Workflow Application project is to contain the complete folder structure or the selected folder or file only.

6. Click **Finish.**

The selected files (QuickForms, in the example above) contained in the local directory will be imported into your Workflow Application project.

Note

If the **Create Complete folder structure** option has been selected, confirmation messages for overwriting files will not be displayed, regardless of whether the **Overwrite existing resources without warning** option is selected.

Note

In addition to importing folders and files from the local file system, you can also import entire Workflow Application projects to your workspace. For more information, refer to section 3.1.12 Importing Workflow Application Projects.

3.1.5 Searching for Folders and Files

The selected files (QuickForms, in the example above) contained in the local directory will be imported into your Workflow Application project.

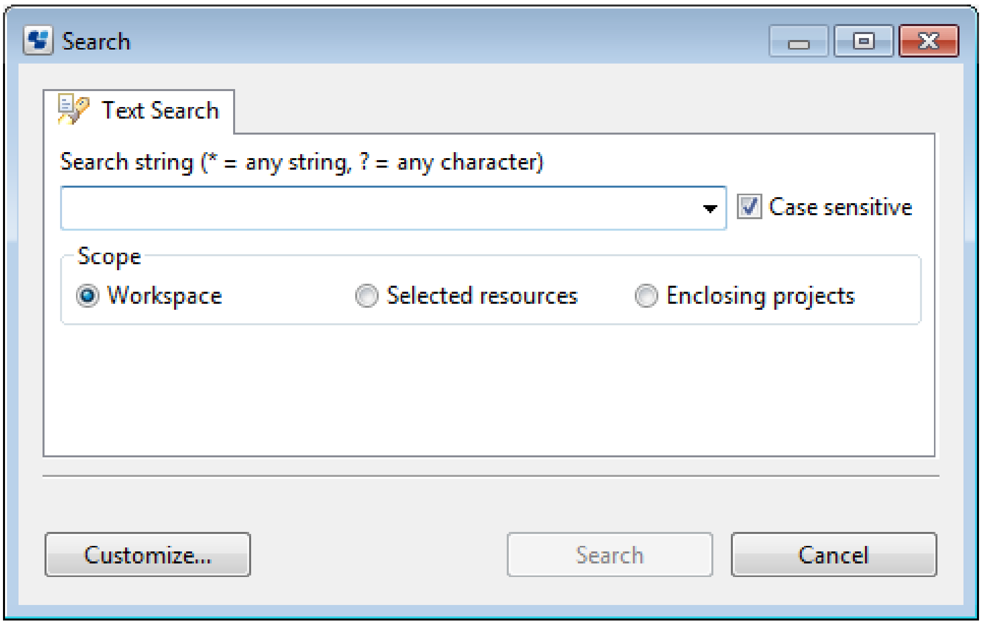
Systemwalker Runbook Automation allows you to search for folders and files in your workspace or in a specific Workflow Application project. Search is based on the text search method. This method specifies that the files you are searching for contain certain text strings in the title, contents, or properties. You can also specify where to look for folders and files and thus limit the range of your search.

To search for specific folders and files:

1. Select **Search** from the Toolbar( ).

   The **Search** dialog is displayed.

   Figure 3.6 Searching for files

   Search string (* = any string, ? = any character)

   ☑ Case sensitive

   Scope

   ● Workspace ○ Selected resources ○ Enclosing projects

2. Type in a search string or select a string from the drop-down list.

   The **Search** dialog shows the string you have selected. You can search by the following criteria:

   - All or part of a folder's or file's name
   - A word or phrase in the name

3. If you want Interstage BPM Studio to distinguish between uppercase and lowercase letters, select the Case sensitive check box.

4. Select one of the Scope check boxes to narrow the scope of search. You have the following options:

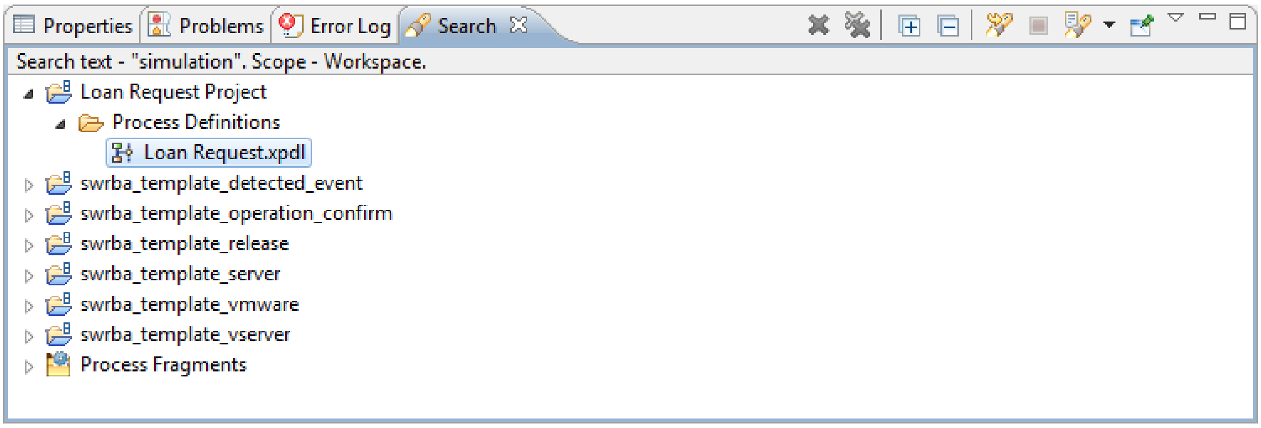
   - **Workspace**: Sets the search range to the file system folder where your work is stored.
   - Selected resources: Sets the search range to the Workflow Application project you have selected in the **Navigator** view.

- Enclosing projects: Sets the search range to the selected project, including the resources that are opened in the Resource editor.

5. Click **Search** to start searching for the specified folders or files.

   The search results are displayed in the **Search** view. The following example shows the search results for the term "simulation" in your workspace:

Figure 3.7 Displaying search results



Search text - "simulation". Scope - Workspace.

- Loan Request Project
  - Process Definitions
    - Loan Request.xpdl
- swrba_template_detected_event
- swrba_template_operation_confirm
- swrba_template_release
- swrba_template_server
- swrba_template_vmware
- swrba_template_vserver
- Process Fragments

From the results list, you can take further actions on the files you find, such as viewing or editing the files. To do this, double-click the files to open them in the editor window above the **Search** view.

3.1.6 Working With Resources

Systemwalker Runbook Automation Studio allows you to add new resources to Workflow Application projects.

Prerequisite:

- You have created a Workflow Application project that contains all components that are required for managing your project.

Resources indicate the following files:

- FTP Agent files
- HTTP Agent files
- Custom Config files
- Configuration file for Java Agents used by the application (agentsConfig.xml)
- File Listener file (fileListenerConf.xml)

The above resource files are allocated in the Resource folder under the project folder.

Note

You can add only one Java Agent and one File Listener file to your project.

To add new resources to a project:

1. Right-click your project in the **Navigator** view and select **New** from the pop-up menu.
2. Select the resource that you want to add to your project.

   You have the following options:

   - **FTP Agent**: Use this option to create a new FTP Agent.
   - **HTTP Agent**: Use this option to create a new HTTP Agent.
   - **Custom Config**: Use this option to create new Custom Configuration files.
   - **Agents**: Use this option to create a new Java Agent.

- **File Listener**: Use this option to create new File Listener file.
- **Process Scheduler File**: Use this option to create a new Process Scheduler file.

3. If you want to edit existing resources:

   a. Open any of the following files in the Resource editor.

      - FTP Agent file
      - HTTP Agent file
      - Custom Config file
      - Java Agent file (agentsConfig.xml)
      - File Listener file (fileListenerConf.xml)

   b. Edit the files.

      - To edit FTP Agent files
        Refer to "11.11 Defining FTP Agents".
      - To edit HTTP agent files
        Refer to "11.12 Defining HTTP Agents".
      - To edit Custom Config files
        Refer to "C.5 Custom-Built Files".
      - To edit the Java Agent file (agentsConfig.xml)
        Refer to "11.10 Defining Java Agents".
      - To edit the File Listener file (fileListenerConf.xml)
        Refer to "11.9 Using Triggers" and "11.13 Defining File Listener".

   c. Click the **Save** button in the toolbar to save your changes.

4. If you want to copy and paste existing resource files:

   a. In the **Navigator** view, right-click a resource file and select **Copy** from the pop-up menu.

   b. Navigate to the Resources folder of another project, right-click this folder, and select Paste from the pop-up menu.

   The resource file is pasted to the new location. If the same file already exists at the new location, Interstage BPM Studio will ask you to overwrite the file. Click Yes to complete the paste procedure.

   Note

   You can only copy and paste the contents of one Resources folder to another. You cannot

   copy the Resources folder itself.

3.1.7 Copying Workflow Application Projects

You can copy Workflow Application projects with all of their contents from one location to another.

Note

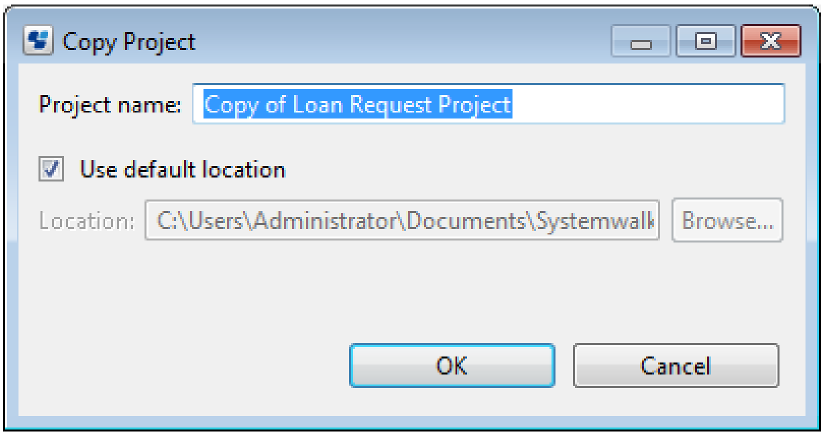
You can copy Workflow Application projects only. Server projects and scenario projects cannot be copied.

To copy a project:

1. Right click the project in the **Navigator** view. Select **Copy** from the pop-up menu.
2. Right click again and select **Paste** from the pop-up menu.

3. In the **Copy Project** dialog, type a name for the copy.

Figure 3.8 Copying a Project



Copy Project

Project name: Copy of Loan Request Project

☑ Use default location

Location: C:\Users\Administrator\Documents\Systemwalk

4. As a default, the project is stored in your workspace. If you want to change the location:

   a) Clear the **Use default location** check box.

   b) Click **Browse**. Select an existing folder or create a new one.

5. Click **OK**.

   The copy is listed in the **Navigator** view.

You can copy several projects at once. Hold down the <Shift> or <Ctrl> key to select all projects to be copied.

3.1.8 Renaming Workflow Application Projects

Note

You can rename Workflow Application projects only. Server projects and the scenario project cannot be renamed.

To rename a local project:

1. Right click the project in the **Navigator** view. Select **Rename** from the pop-up menu.
2. Type the new name for the project.
3. Press the <Enter> key.

3.1.9 Closing Workflow Application Projects

When you close a project, it is still displayed in the **Navigator** view, but you cannot change it any more. Its content, for example process definitions, is no longer visible.

To close a Workflow Application project:

- Right click the project in the **Navigator** view. Select **Close Project** from the pop-up menu.

The icon 📁 left to the project name indicates that it is closed.

You can close several projects at once. Hold down the <Shift> key or <Ctrl> key to select all projects to be closed.

Note

The closing procedure applies to server projects, too. For more information on server projects, refer to section 3.2 Managing Server Projects.

3.1.10 Opening Workflow Application Projects

You can reopen a project that is currently closed. The icon left to the project name indicates whether it is closed (📁) or open (📂).

Note

You can reopen server projects, too. For general information on server projects, refer to section 3.2 Managing Server Projects.

To open a project:

- Right click the project in the **Navigator** view. Select **Open Project** from the pop-up menu.

3.1.11 Removing Workflow Application Projects

You can remove projects with all of their contents from Systemwalker Runbook Automation Studio.

Note

You cannot remove the projects named "Simulation Scenarios" and "Process Fragments". When you remove a Workflow Application project, all files in the project are also removed and no longer accessible.

To remove a Workflow Application project:

1. Right click the project in the **Navigator** view. Select **Delete** from the pop-up menu.
2. Confirm the removal of the project by clicking **Yes**.

3.1.12 Importing Workflow Application Projects

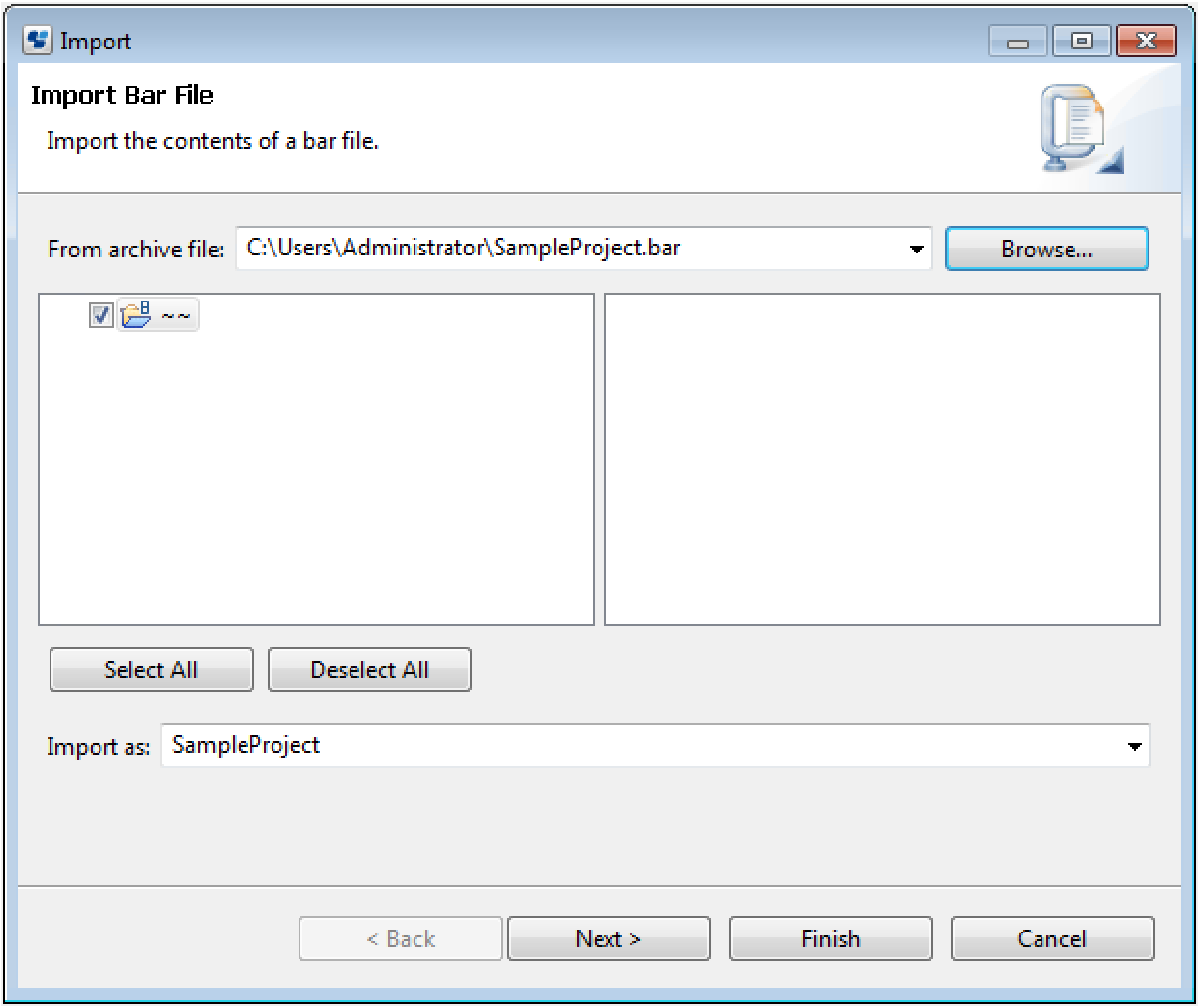
You can download Workflow Application projects packaged into .bar files from the local file system to your workspace.

To import a Workflow Application project from your local file system:

1. In the **Navigator** view, right-click the project name. Select **Import Bar File** from the pop-up menu.

   The **Import Bar File** dialog is displayed.

   Figure 3.9 Displaying the Import Bar File dialog



Import

Import Bar File

Import the contents of a bar file.

From archive file: C:\Users\Administrator\SampleProject.bar

☑ ~~

Import as: SampleProject

2. Specify the project to be imported. To do so:

   a. Select a .bar file from the **From archive file** drop-down list, or click **Browse**. From your local file system or a file server, select the .bar file that you want to import, and click **OK**. The selected file is added to the **From archive file** field. The .bar file name is displayed with the absolute path.

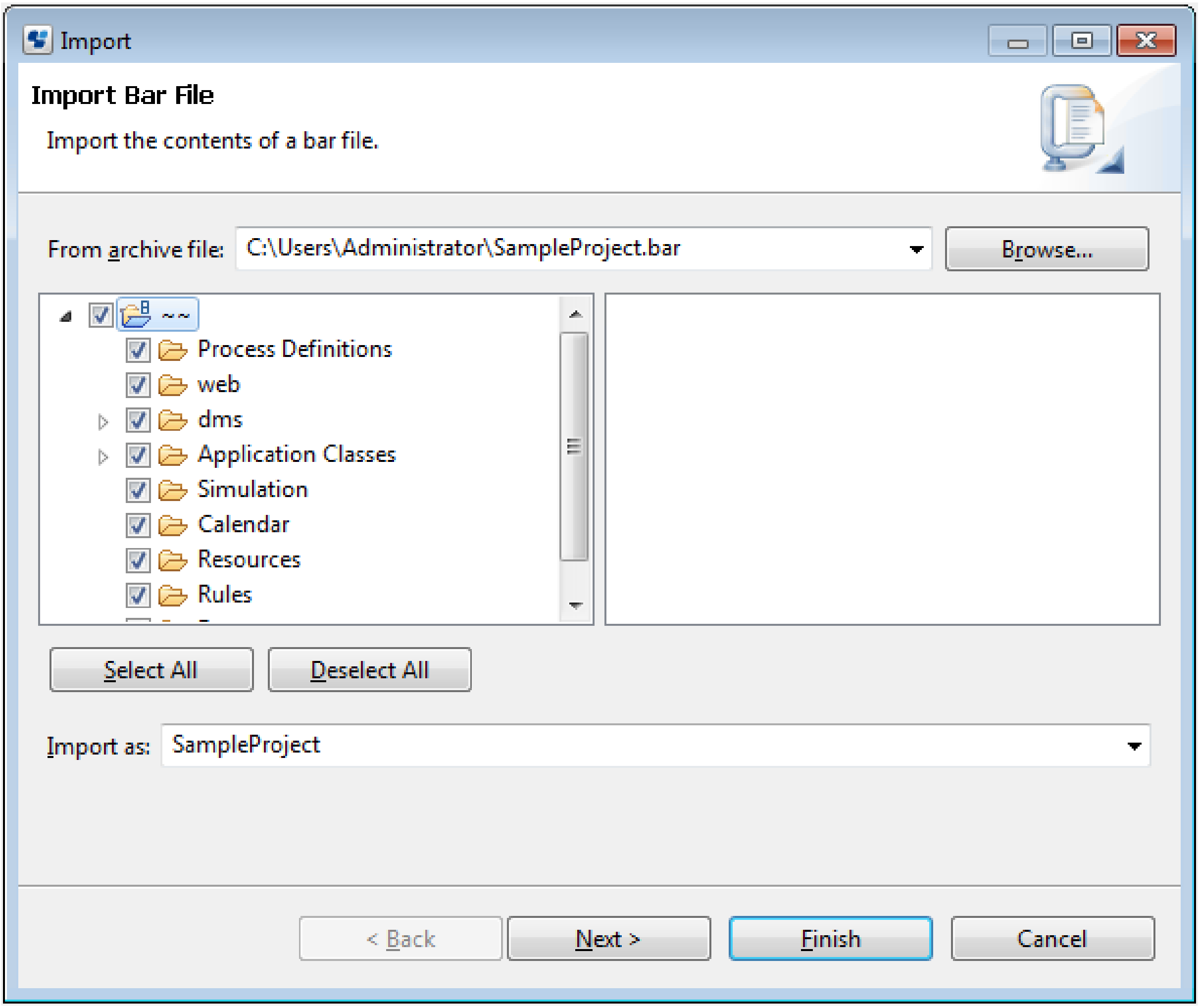
      If the project that you are importing has the same name as another project in your workspace, you will encounter an error message.

   b. Use the check boxes to select or deselect components to be imported. You can select or deselect all project components at once, using the **Select All** or **Deselect All** tabs.

   c. From the **Import as** drop-down list, select a name for the project to be imported. If you are trying to import a project that has the same name as a project in your workspace, an error message is displayed. Change the project name to avoid any conflicts.

After specifying all import settings, the **Import Bar File** dialog looks as follows:

Figure 3.10 Specifying import settings



Import

Import Bar File

Import the contents of a bar file.

From archive file: C:\Users\Administrator\SampleProject.bar

☑ ~~
- ☑ Process Definitions
- ☑ web
- ☑ dms
- ☑ Application Classes
- ☑ Simulation
- ☑ Calendar
- ☑ Resources
- ☑ Rules

Import as: SampleProject

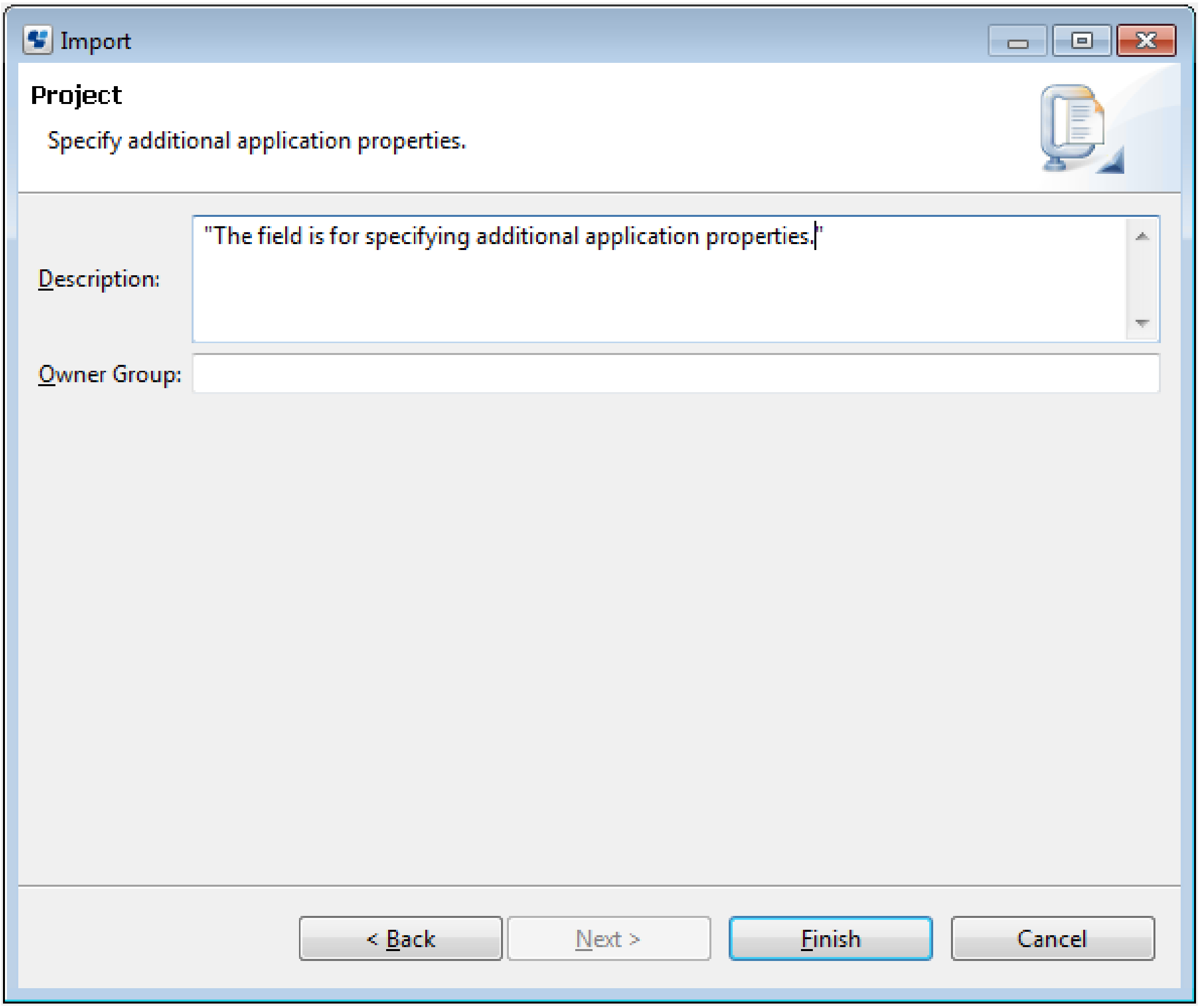
3. You can provide additional information about your project. To do so:

   a. Click **Next**.

   b. In the **Project** dialog, type in additional information about your project.

Providing additional project information is optional. You can type in a brief description of the project as well as the name of the project owner. However, the setting for project owner is not required.

Figure 3.11 Providing additional information



Description: "The field is for specifying additional application properties."

Owner Group:

4. Click **Finish** to import the selected project components to your workspace.

   While the application is being imported, a temporary project is listed in the **Navigator** view. You can identify this temporary project by its name starting with two tildes and the name you entered for the application project, e.g. **~~MyApp**. As soon as the import is finished, the temporary application project is deleted and the imported project is listed in the **Navigator** view.

3.1.13 Exporting Workflow Application Projects

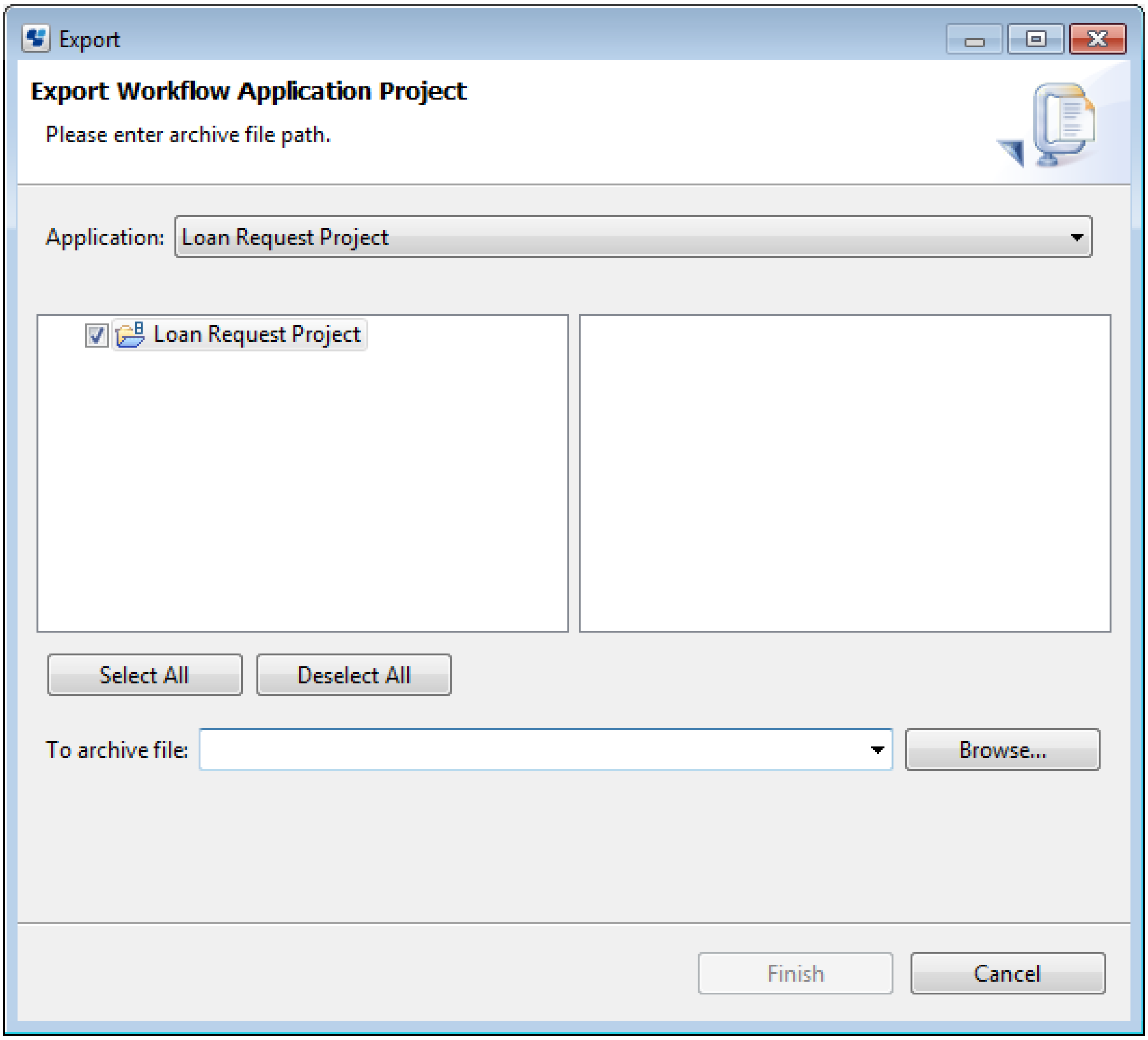
You can export Workflow Application projects from your workspace to the local file system and package all files that make up your application into a .bar file. This .bar file can then be registered on the Management Server from the Web Console.

To export a Workflow Application project to the local file system:

1. In the **Navigator** view, right-click the project name. Select **Export** from the pop-up menu. As an alternative, you can also select this option from the **File** menu.

   The **Export Workflow Application Project** dialog is displayed. It shows your local file system and the name of the .bar file you have selected in the workspace.

   Figure 3.12 Displaying the Export Workflow Application Project dialog



Export

Export Workflow Application Project

Please enter archive file path.

Application: Loan Request Project

☑ Loan Request Project

To archive file:

2. Specify the project components to be exported. To do so:

   a. Select an application to be exported from the **Application** drop-down list. The default application is the name of the project you have selected in the **Navigator** view.

      All files that make up your application are displayed.

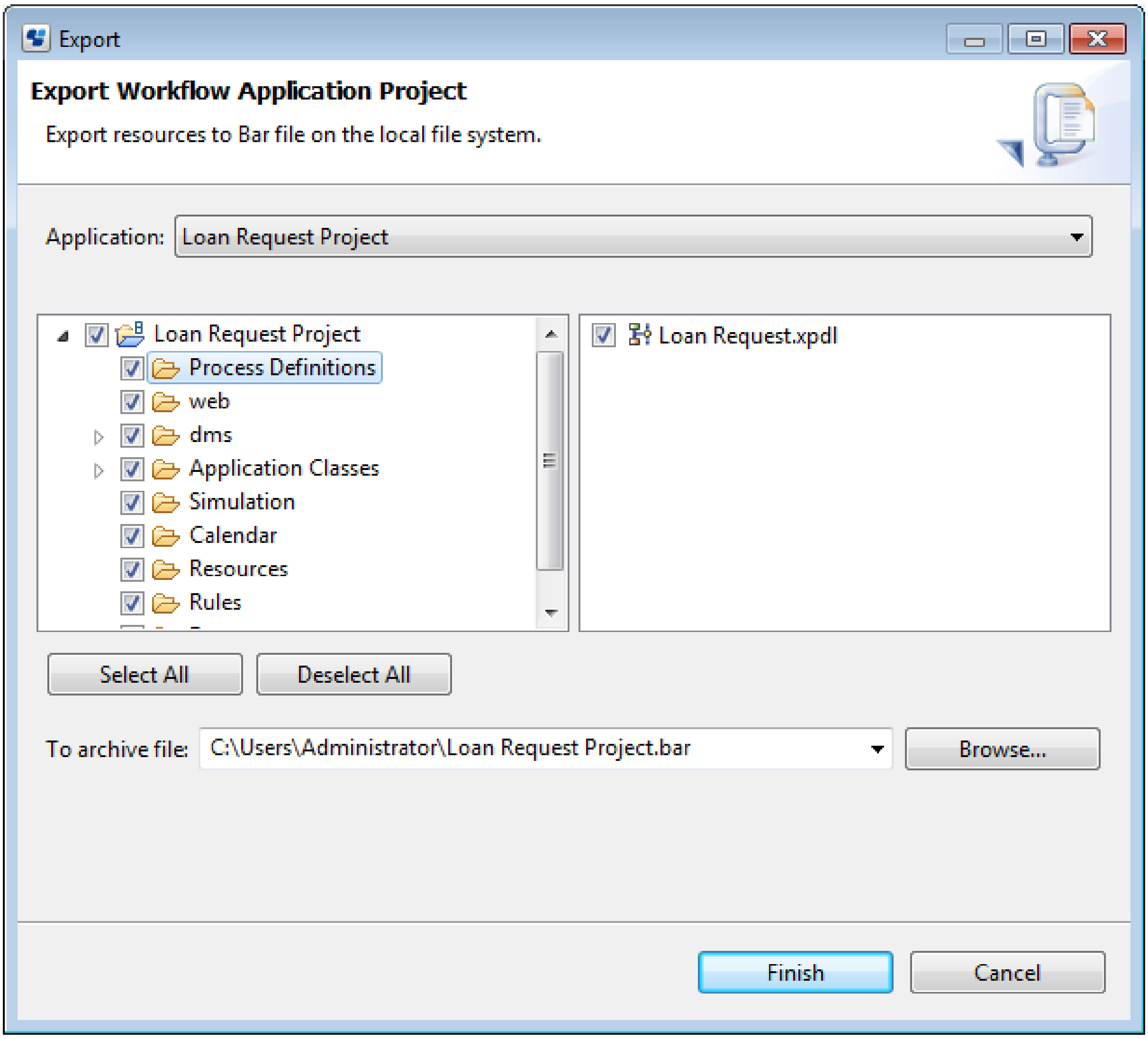
   b. Use the checkboxes to select or deselect specific files to be exported. You can select or deselect all files at once, using the **Select All** or **Deselect All** buttons.

   c. Use the **To archive file** drop-down list to specify a file to archive the exported project. You can also browse for an archive file using the **Browse** tab. The default archive file takes the name of the project you are exporting (in the example, BankLoan.bar file).

      The **To archive file** drop-down list contains file paths (for example C:\Fujitsu\Systemwalker\SWRBA_Studio\workspace\Archive).

The following figure shows how to select specific project components and specify the default archive file:

Figure 3.13 Specifying project components and archive file



Export

Export Workflow Application Project

Export resources to Bar file on the local file system.

Application: Loan Request Project

- ☑ Loan Request Project
  - ☑ Process Definitions
  - ☑ web
  - ☑ dms
  - ☑ Application Classes
  - ☑ Simulation
  - ☑ Calendar
  - ☑ Resources
  - ☑ Rules

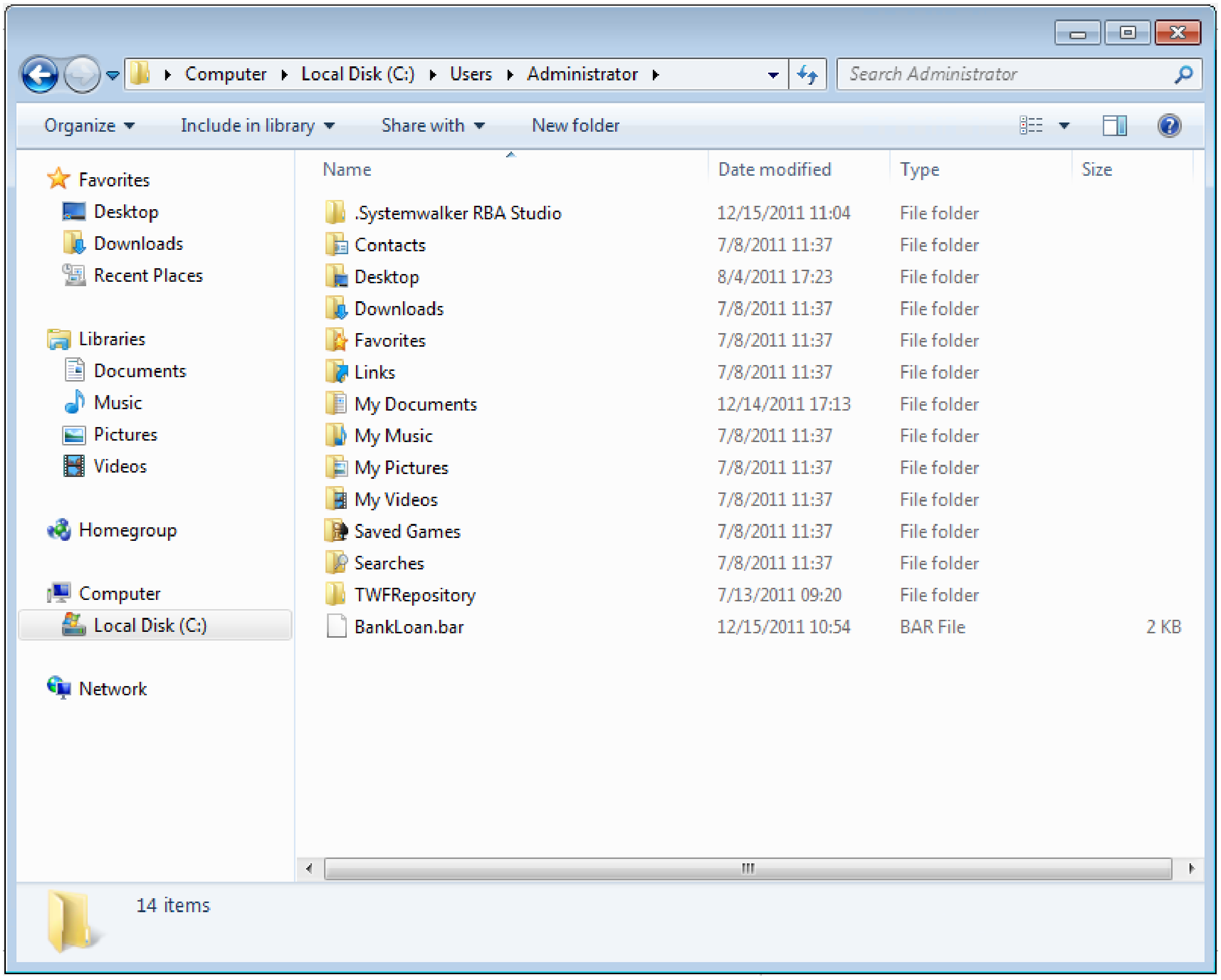
☑ Loan Request.xpdl

To archive file: C:\Users\Administrator\Loan Request Project.bar

3. Select the project components to be exported, specify an archive file, and click **Finish** to export your project.

   Your Workflow Application project is now stored as .bar file in the specified directory in the local file system.

   Figure 3.14 Location of the exported project



Name	Date modified	Type	Size
.Systemwalker RBA Studio	12/15/2011 11:04	File folder	
Contacts	7/8/2011 11:37	File folder	
Desktop	8/4/2011 17:23	File folder	
Downloads	7/8/2011 11:37	File folder	
Favorites	7/8/2011 11:37	File folder	
Links	7/8/2011 11:37	File folder	
My Documents	12/14/2011 17:13	File folder	
My Music	7/8/2011 11:37	File folder	
My Pictures	7/8/2011 11:37	File folder	
My Videos	7/8/2011 11:37	File folder	
Saved Games	7/8/2011 11:37	File folder	
Searches	7/8/2011 11:37	File folder	
TWFRepository	7/13/2011 09:20	File folder	
BankLoan.bar	12/15/2011 10:54	BAR File	2 KB

3.1.14 Downloading Workflow Application Projects from a Server

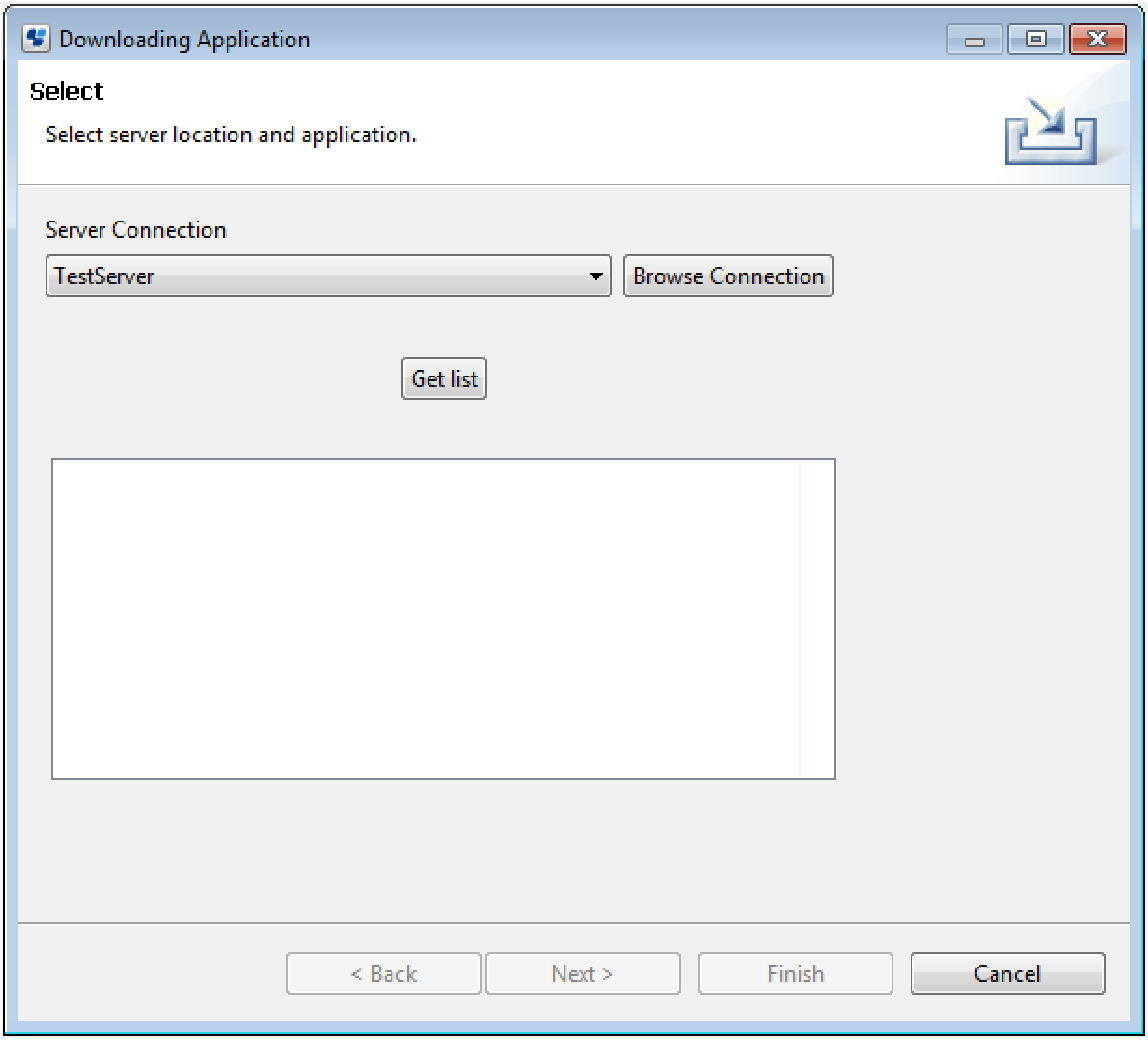
You can download Workflow Application projects packaged into .bar files from a Management Server to your local workspace. A wizard walks you through the downloading procedure.

To download a Workflow Application project from a server:

1. In the **Navigator** view, right-click the project name. Select **Download Application from Server** from the pop-up menu.

   The **Downloading Application** dialog is displayed. It automatically shows the last active server connection:

   Figure 3.15 Displaying the Downloading application dialog



Downloading Application

Select

Select server location and application.

Server Connection

TestServer

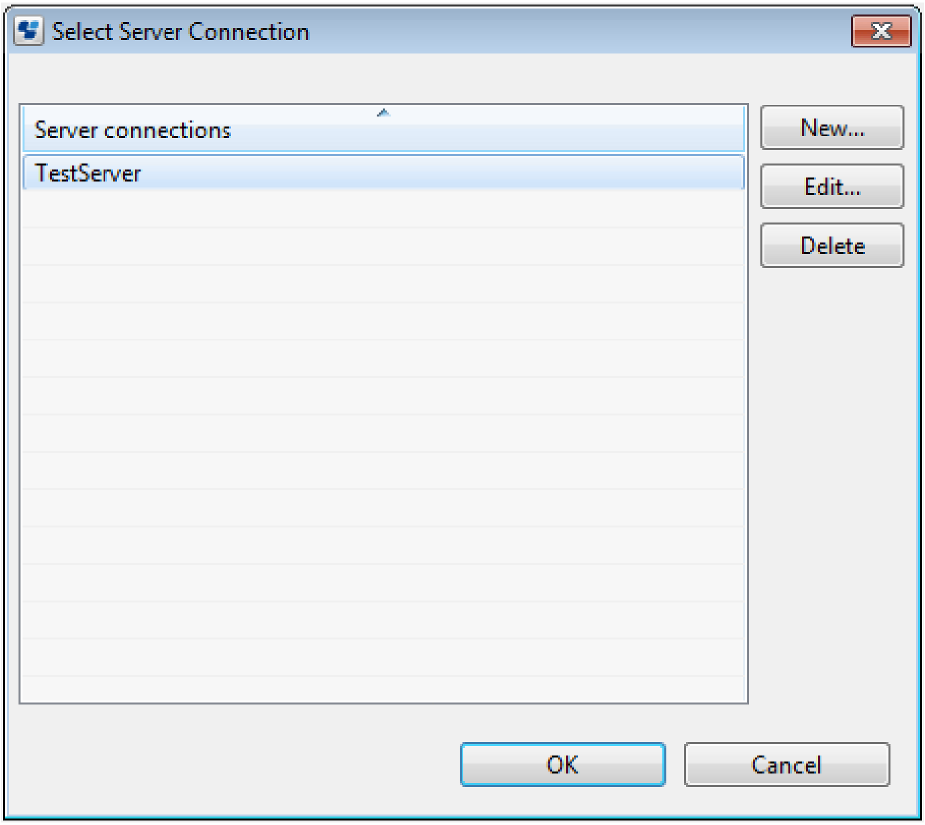
2. If you want to use an available server connection, select a server from the **Server Connection** drop-down list and proceed with Step 5.

3. If you want to edit one of the server connections or specify a new server:

   a. In the **Downloading Application** dialog, click **Browse Connection**.

      The **Select Server Connection** dialog is displayed. The available server connections are listed in alphabetical order. By default, the first server on the list is always highlighted. If you have not yet specified any server connection, the server connection field is empty. In the following example, only one server connection is available.

      Figure 3.16 Displaying server connections



   b. In the **Select Server Connection** dialog, click **New**.
      The **Server Connection Setting** dialog is displayed.

   c. In this dialog:

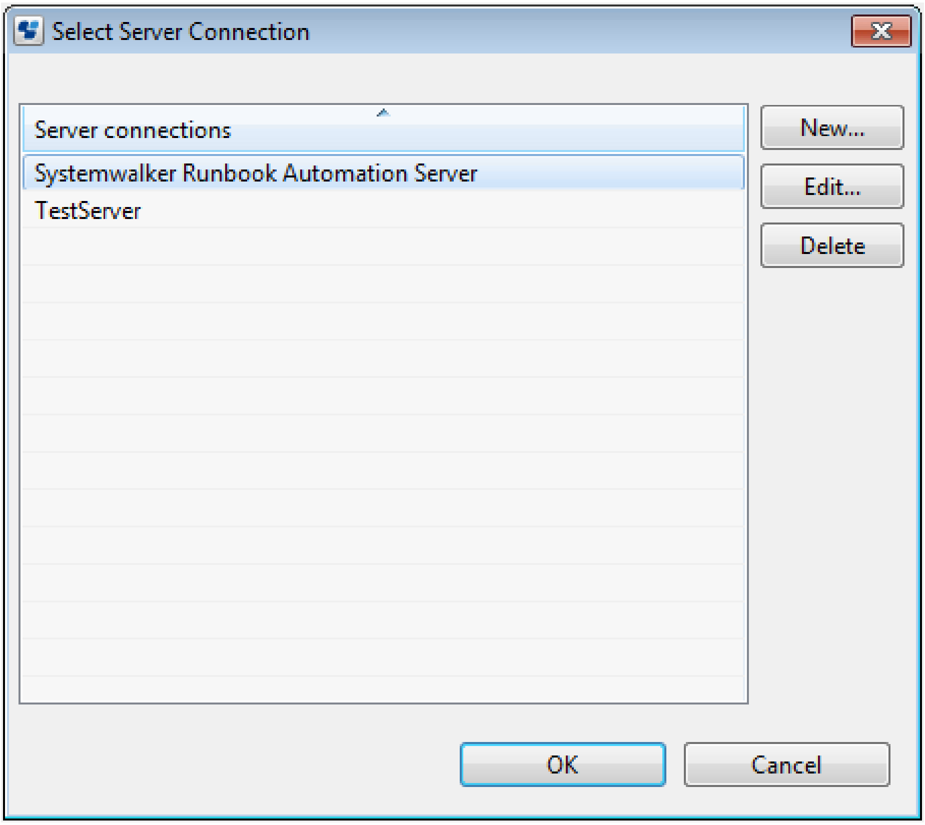
      1. Specify parameters for the new server connection. Refer to "2.4.1 Server Connection Settings" in Chapter 2, "Basic Information and Customization for the User Interface" for details about the required server parameters.

      2. Optional: Select the **Save Password** checkbox if you do not want to enter it again the next time you log into the Management Server.

d. Click **OK** to confirm the server settings.

The new server connection is added to the server connections list. In the following example, the Management Server has been specified:

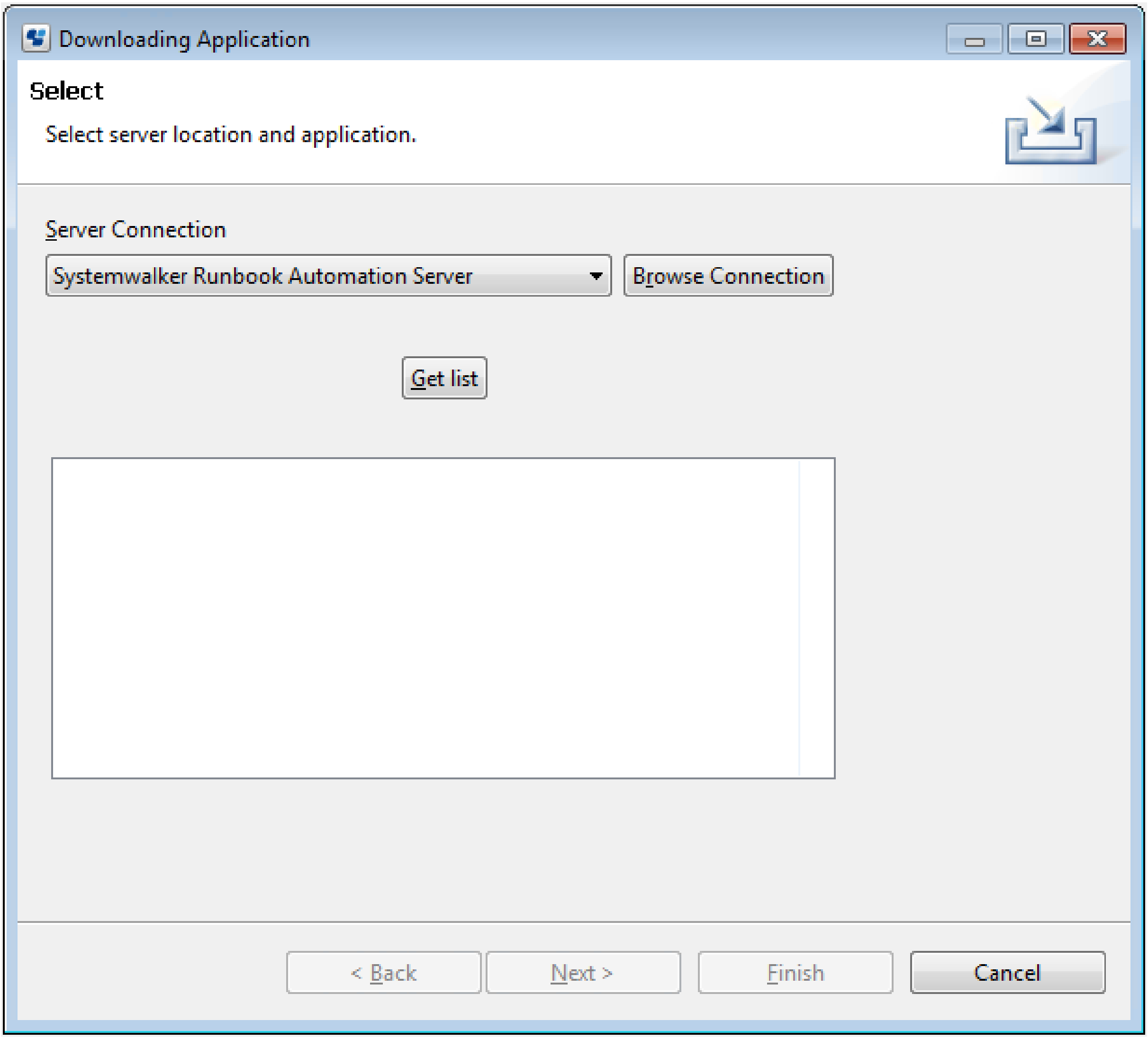
Figure 3.17 Displaying new server connection



e. In the **Select Server Connection** dialog, click **OK** to confirm your server choice.

The **Downloading Application** dialog displays the selected server connection (Management Server, in the example):

Figure 3.18 Displaying the new server connection



Downloading Application

Select

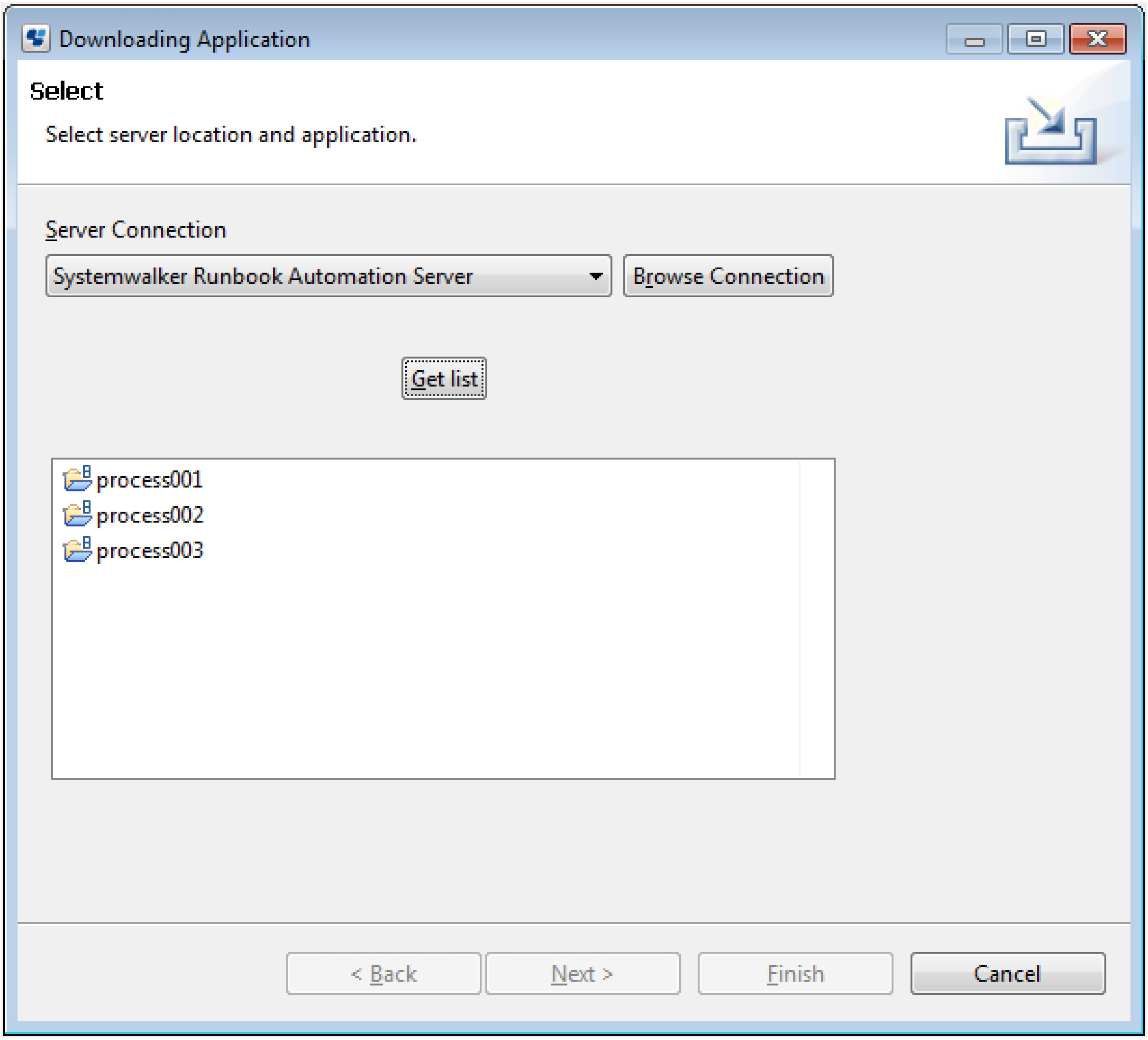
Select server location and application.

Server Connection: Systemwalker Runbook Automation Server

4. In the **Downloading Application** dialog, click the **Get List** tab.

   All projects using the specified server connection are displayed in the empty space beneath the tab. In the example, three projects are displayed.

Figure 3.19 Displaying Workflow Application projects



Downloading Application

Select

Select server location and application.

Server Connection

Systemwalker Runbook Automation Server

process001
process002
process003

5. Select the project you want to download from the server, and click **Next**.
   The **Downloading Application** dialog is displayed. It shows all components of the selected project.

   You can use the check boxes to select or deselect components to be downloaded. You can also select or deselect all project components at once, using the **Select All** and **Deselect All** tabs.

   Use the **Import as** drop-down list to select a name for the project that you want to download. The default name is the name of the project in your workspace. Change this name to avoid any name conflicts. The following example shows how to select project components to be downloaded:

Note

The **Download as** list contains the names of all available projects. However, it does not contain any file paths.

6. Select project components to be downloaded, and click **Next**.
   The **Project** dialog is displayed. Here you can enter additional information about the project.

7. Type in a project description. The setting for project owner is not required.

Note

Providing additional project information is optional. You can leave out this step and immediately finish the download process.

8. Click **Finish** to download the project from the specified server.

   While the application is being downloaded, a temporary project is listed in the **Navigator** view. You can identify this temporary project by its name starting with two tildes and the name you entered for the application project, e.g. **~~MyApp**. As soon as the download is finished, the temporary application project is deleted and the downloaded project is listed in the **Navigator** view.

   Note

   When a workflow application project is downloaded from a server using Systemwalker Runbook Automation Studio, if multiple version exist in Process Definitions for download, the Process Definition will be determined by the following rules.

   - The state for download will be published and draft.
   - If there is published state one, it will be for download.
   - If there is not Process definition of published state, The process definition which the biggest number of draft state is intended Process Definition.

   Draft Process Definition that are part of the application, only user can download it.

   Except user can download Published Process Definition.

3.1.15 Uploading Workflow Application Projects to a Server

You can upload Workflow Application projects from Systemwalker Runbook Automation Studio to the Management Server and package all files that make up your application into a .bar file. An uploading wizard walks you through the procedure of uploading a project from your workspace to a remote server.

Note

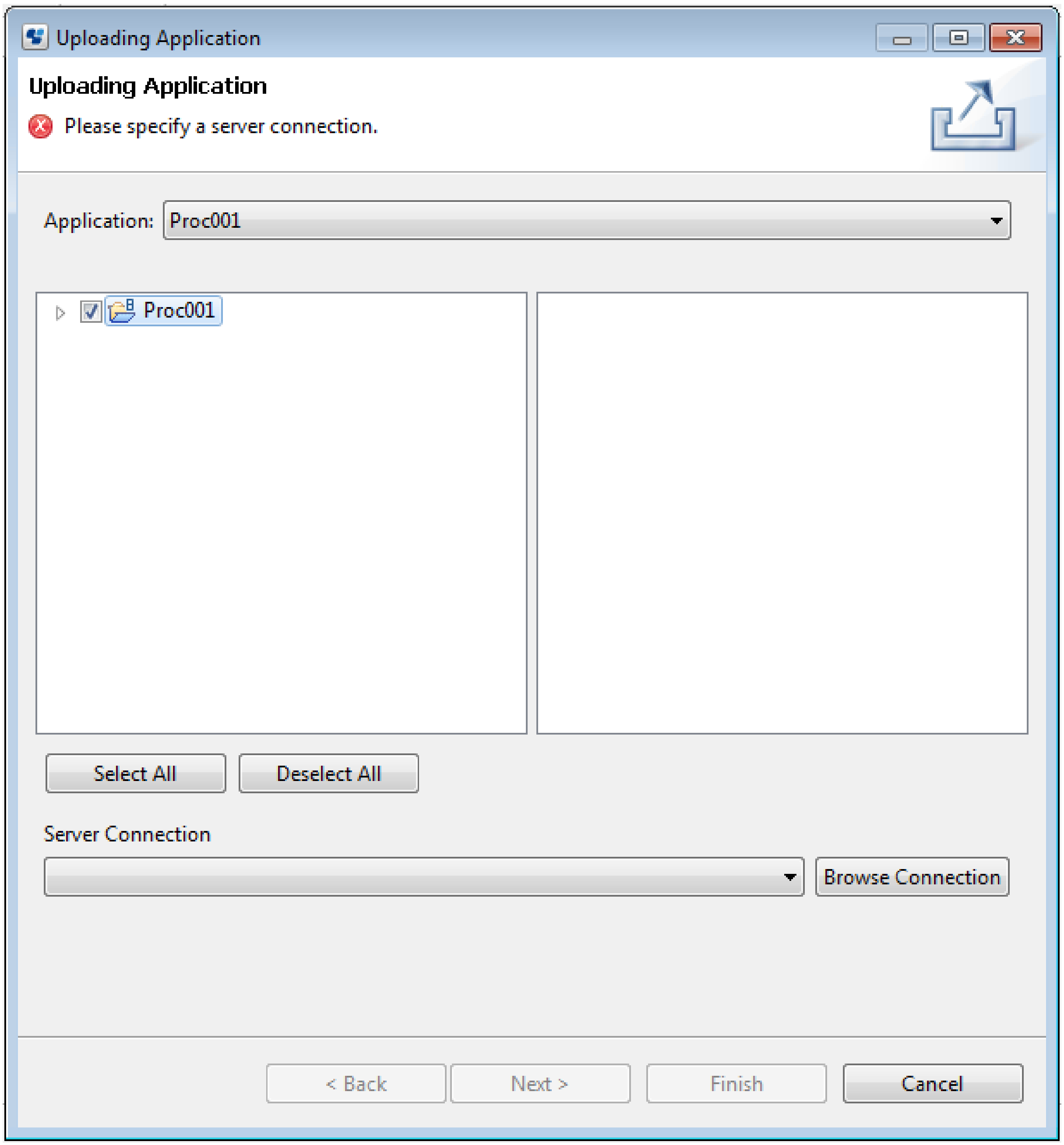
You need to have Administrator rights on the remote server to be able to perform this task.

To upload a Workflow Application project to a remote server:

1. In the **Navigator** view, right-click the project name. Select **Upload Application** from the pop-up menu.

   The **Uploading Application** dialog is displayed. As long as no server connection is specified, the dialog shows an error message.

   Figure 3.20 Displaying the Uploading application dialog

   Uploading Application

   Please specify a server connection.

   Application: Proc001

   ☑ Proc001

   Server Connection

2. To specify the project to be uploaded:

   a. Select a project from the **Application** drop-down list.

      The default application is the name of the project you have selected in the **Navigator** view.

   b. Use the check boxes to select or deselect components to be uploaded. You can upload an entire project or individual project components. To do this, expand the project folder, clicking the **+** sign in front of the project name. The possibility to select project components prevents you from uploading identical projects. Besides, you do not have to deploy files that you don't want to deploy. In addition, you can select or deselect all components at once, using the **Select All** or **Deselect All** tabs. By default, all project components are selected.

3. Select a remote server.

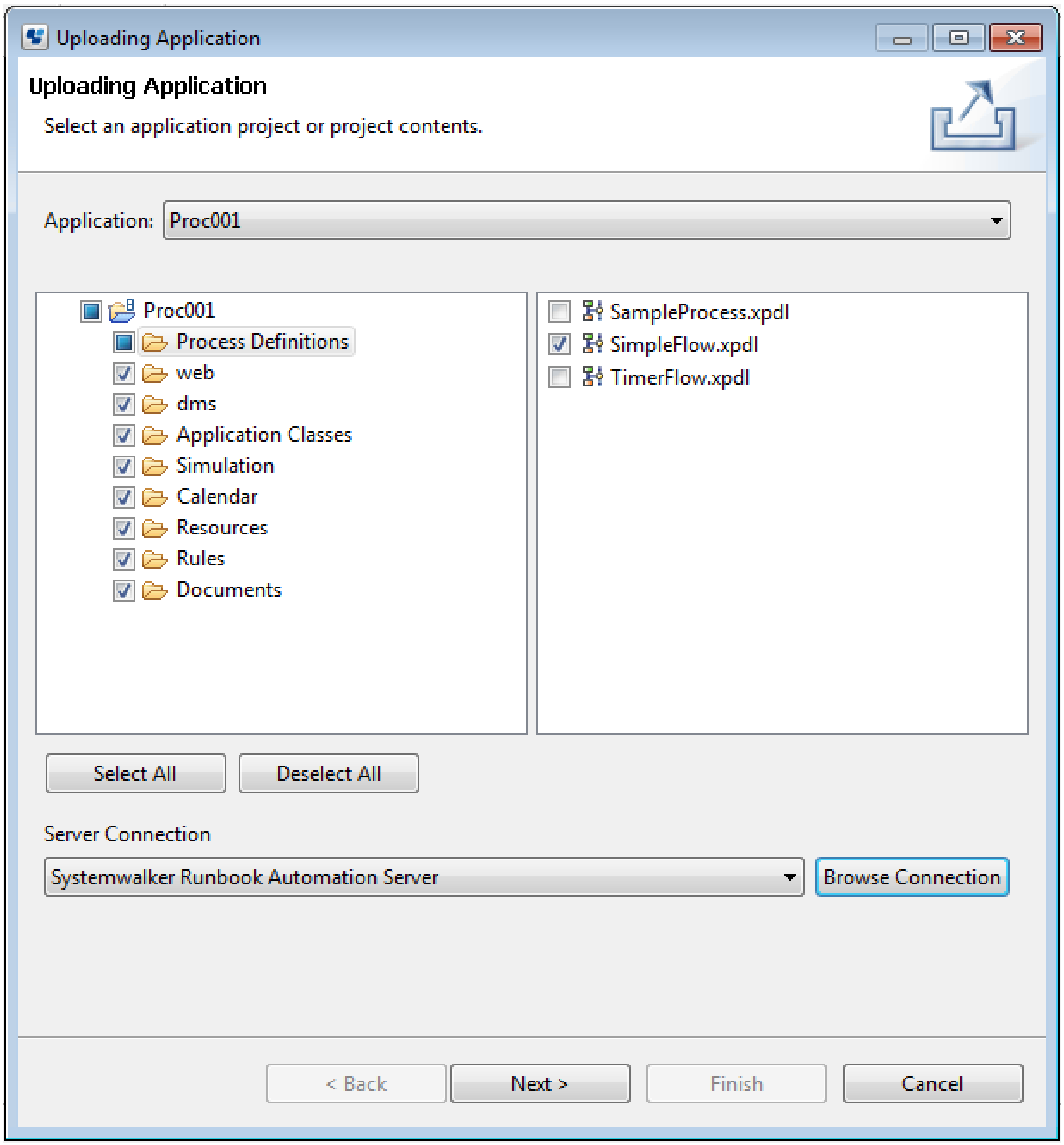
   You can select a server from a drop-down list or by browsing available server connections.

Select a remote server from the **Server Connection** drop-down list. This list automatically displays the last server you have selected. If you have not yet selected any server, the list is empty. In that case, or if you want to select a different server:

1. In the **Uploading Application** dialog, click the **Browse Connection** button, and select a server. This dialog displays all available server connections.
2. Click **OK** to confirm your selection.
3. If no server is available, in the **Select Server Connection** dialog, click **New** to specify a new server connection. **Server Connection Setting** dialog is displayed.
4. To create a new server connection, enter all the parameters in **Server Connection Setting** dialog that is displayed after clicking the **New** button. Refer to "2.4.1 Server Connection Settings" in Chapter 2, "Basic Information and Customization for the User Interface" to know more about various parameters.

In the example, you have specified a Management Server. The **Uploading Application** dialog now looks as follows:

Figure 3.21 Selecting project components and server



Uploading Application

Uploading Application

Select an application project or project contents.

Application: Proc001

- ■ Proc001
  - ■ Process Definitions
  - ☑ web
  - ☑ dms
  - ☑ Application Classes
  - ☑ Simulation
  - ☑ Calendar
  - ☑ Resources
  - ☑ Rules
  - ☑ Documents

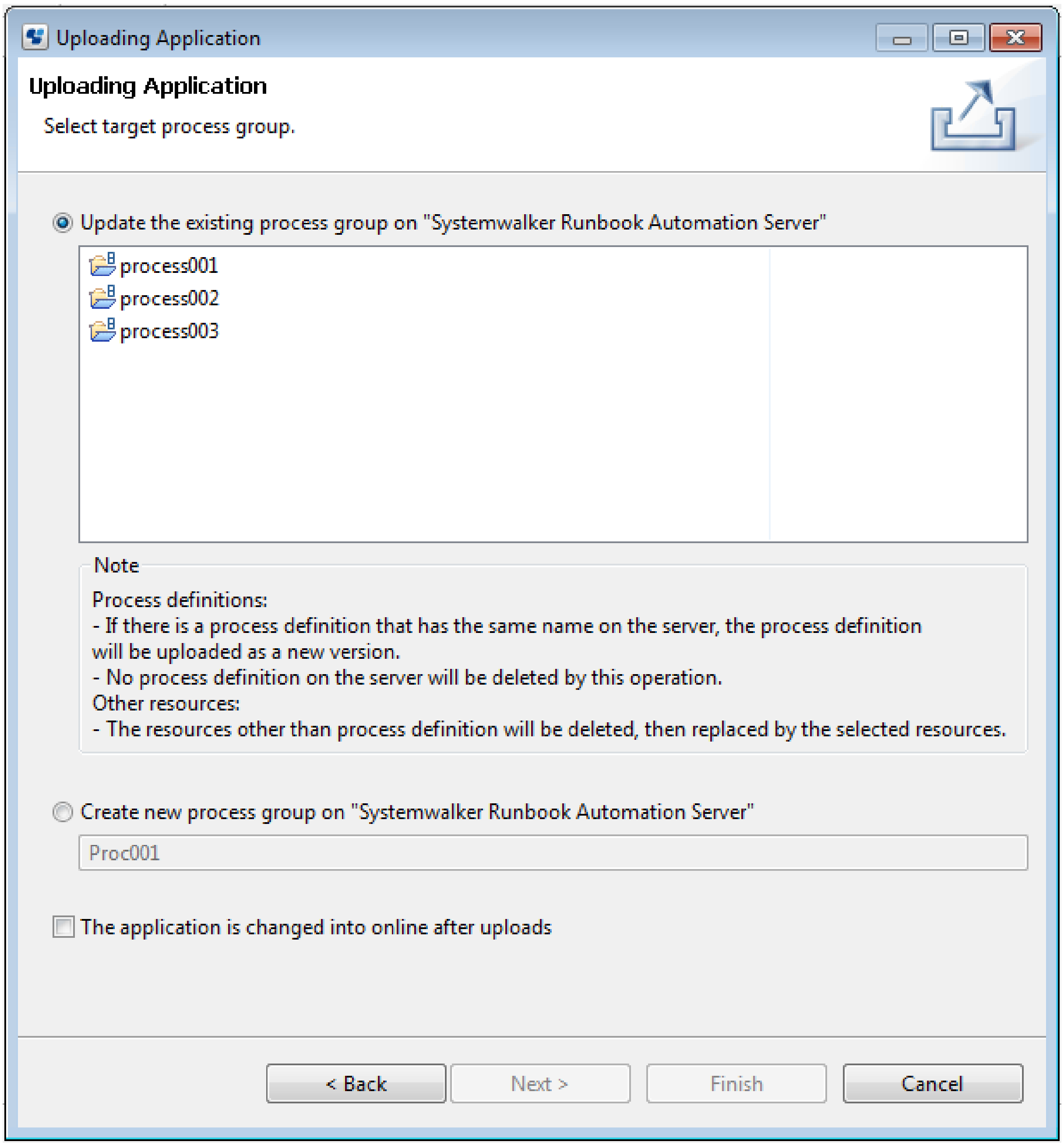
- ☐ SampleProcess.xpdl
- ☑ SimpleFlow.xpdl
- ☐ TimerFlow.xpdl

Server Connection: Systemwalker Runbook Automation Server

4. In the **Uploading Application** dialog, click **Next**.

   You can choose to update a project that already exists on the server, or create a new project:

Figure 3.22 Choosing to update or create an application



Uploading Application

Select target process group.

● Update the existing process group on "Systemwalker Runbook Automation Server"

process001
process002
process003

Note

Process definitions:
- If there is a process definition that has the same name on the server, the process definition will be uploaded as a new version.
- No process definition on the server will be deleted by this operation.

Other resources:
- The resources other than process definition will be deleted, then replaced by the selected resources.

○ Create new process group on "Systemwalker Runbook Automation Server"

Proc001

□ The application is changed into online after uploads

- Select the **Update the existing process group** checkbox to upload your project to another project that already exists on the server.

Note

For Process Definition

- If Process Definition with same name existing on server, Process Definition are added as the latest version.
- Process Definitions are not removed by update procedure.

For the file except of Process Definition

- Replace file on server by uploaded file.

- File on server which not exist among uploaded application.

- Select the **Create new process group to server** checkbox to upload a new project onto the server. If your new project has the same name as another project on the server, it will ask you to overwrite it.

- If you want to set process group where destination of upload to be online, select **The application is changed into online after uploads** checkbox to be on.

Note

For state of process group on server

- When upload, process group on server might to be offline state. If select set process group where destination of upload to be online, set process group where destination of upload to be offline.

- If select **The application is changed into online after uploads** checkbox to be on, process group where destination of upload will be changed into offline automatically, and will be changed into online state after upload procedure.

5. Click **Finish** to upload the new project.

3.1.16 Application Variables

Application Variables are the variables that you can define at workflow application project level. By defining the Application Variables, you can share these variables across all the processes within a specific application project.

Application Variables are dynamic in nature and you can use them for various Java Actions. For example, you can use Application Variables for the Web Service Java Action as a WSDL location. This will allow you to change the WSDL location dynamically without changing the Process Definition Java Action.

You can define Application Variables when you create new application projects.

Along with the process-level variables, Application Variables too are available in the Expression Builder. Refer to *Defining JavaScript Expressions* for more information about Expression Builder.

Application Variables are displayed as **%ApplicationVariable1%** to distinguish them from the process-level variables.

For example, if **Variable1** is an Application Variable then its expression will be

sec.getApplicationVariable("Variable1").

Application Variables are stored in an xml file in the Application folder. The xml file is located at

Workspace >> **Application Name** >> **Appvariable.xml**.

3.1.17 Defining Application Variables

To define Application Variables:

1. Right-click the application project in the **Navigator** view.

2. Click **Properties**.

   The **Properties** dialog for the application project is displayed.

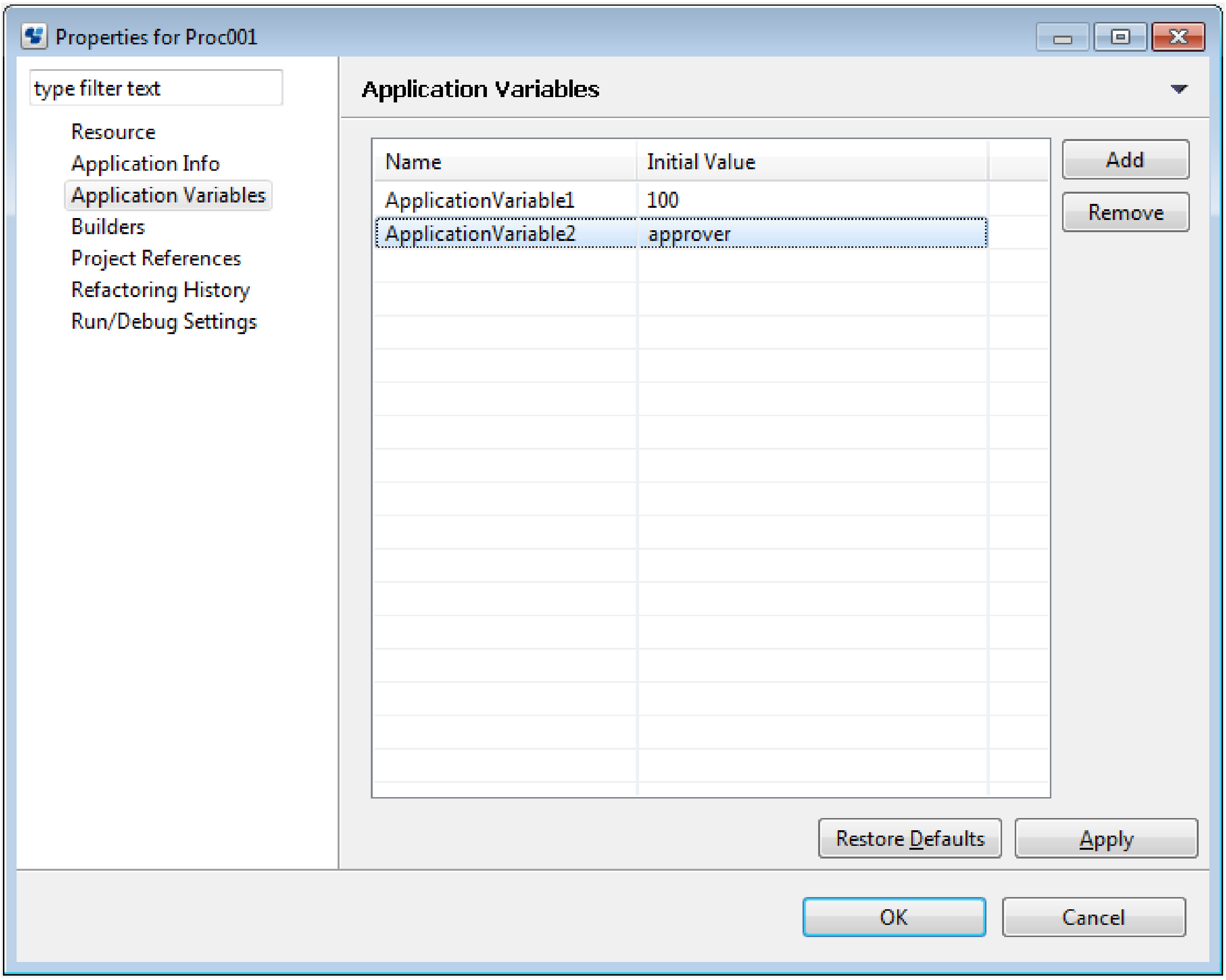
3. Click **Application Variables** in the left pane.

   Application Variables area is displayed in the right pane.

4. Click the **Add** button to add Application Variables.

   An example of defining application variables is shown below:

Figure 3.23 Defining Application Variables



Name	Initial Value
ApplicationVariable1	100
ApplicationVariable2	approver

Note

By default, the Application Variables are added as ApplicationVariable1, ApplicationVariable2 and so on. You can change the names and values.

Note

When Application Variable is defined, and it is set with a certain JavaAction, you can delete this Application Variable. When you delete an Application Variable, it is necessary to check whether this Application Variable is used.

5. Edit the name and the initial value of the Application Variable in the **Name** and **Initial Value** columns respectively.

Note

The maximum allowable length of the variable name is 256 characters and the maximum allowable length of the value is 2000 characters.

6. Click **Apply** and then click **OK** to save the Application Variables.

3.2 Managing Server Projects

This section explains the functions available for managing server projects.

Note

Ensure that you do not edit any process definition that is part of a Workflow Application project of a server project.

3.2.1 Creating Server Projects

This explains how to create a project that accesses a Management Server:

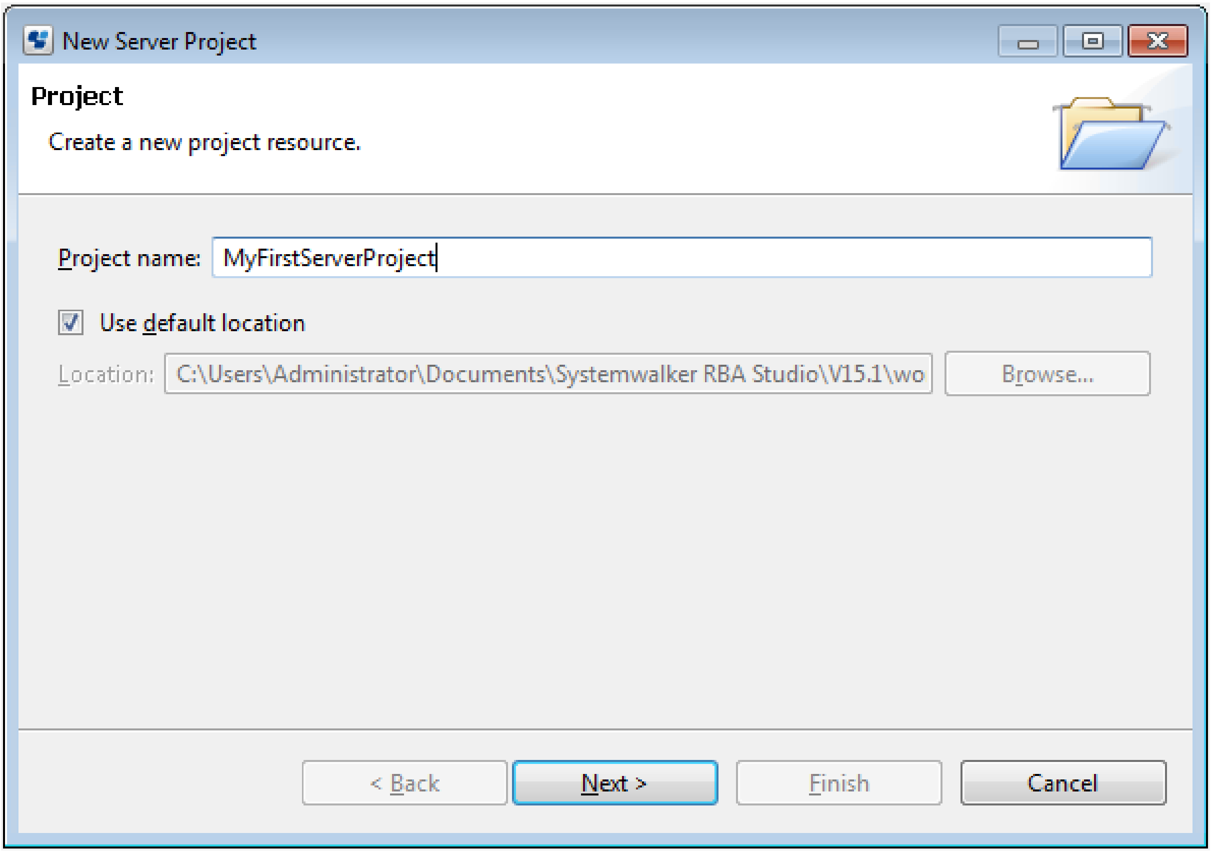
1. Select **File** > **New** > **Project** >**Server**.

   New Server Project dialog is displayed.

2. In **New Server Project**, in the **Project name** field, type in a name for your project.

   The project that you create corresponds to a folder in the file system. As a default, the project is created in your workspace.

   Figure 3.24 Creating a Server Project



New Server Project

Project

Create a new project resource.

Project name: MyFirstServerProject

☑ Use default location

Location: C:\Users\Administrator\Documents\Systemwalker RBA Studio\V15.1\wo

3. To change the location of your project:

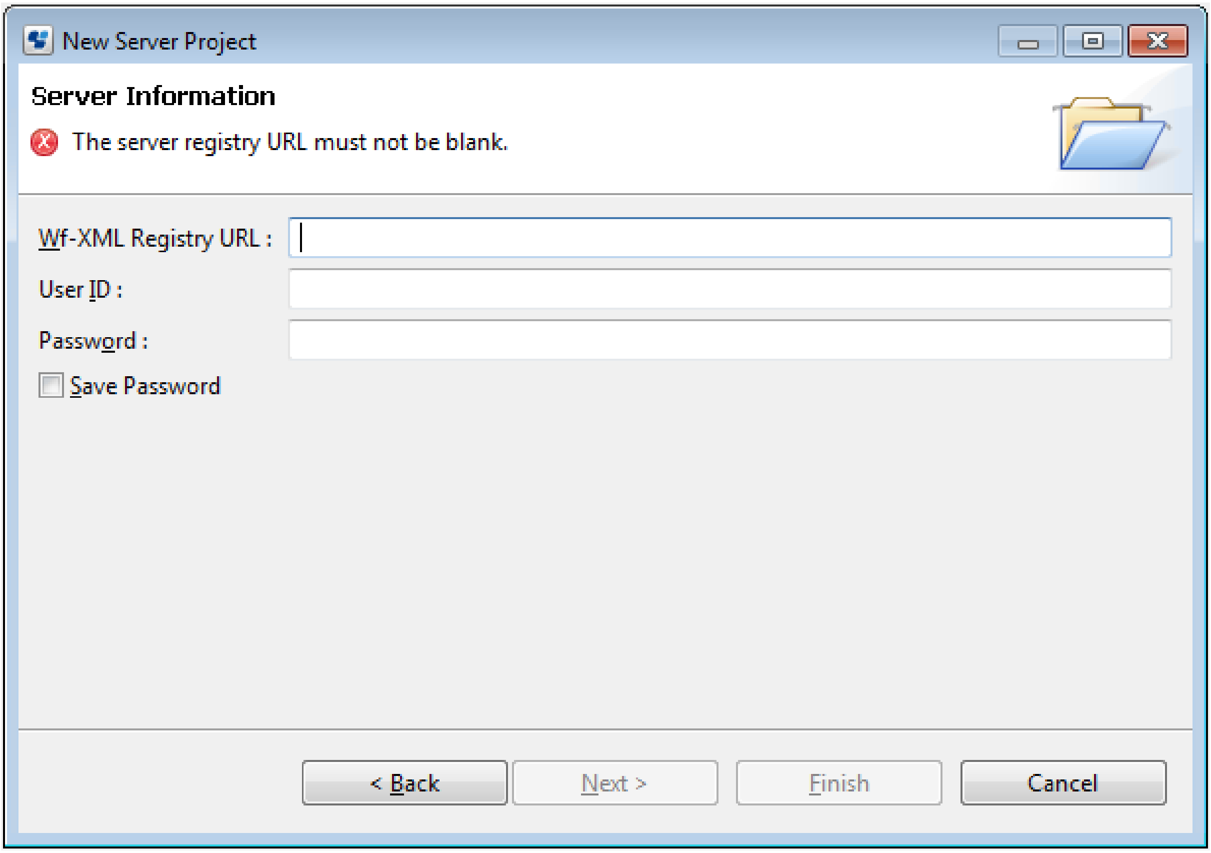
   a. Clear the **Use default location** check box.

   b. Click **Browse**. Select an existing folder, or create a new one. Click **OK**.

4. Click **Next**. The dialog box for typing in information on the location of the Management Server is opened.

   The **New Server Project** dialog for server information, is displayed. Use this dialog to type in the required server information.

   Figure 3.25 Providing server information

   New Server Project

   Server Information

   The server registry URL must not be blank.

   Wf-XML Registry URL :

   User ID :

   Password :

   □ Save Password

5. Enter the **Wf-XML Registry URL**. This URL is composed of the host name, port number. The sample URL is as given below:

   ```
   http://<hostname>:<port>/<context>/_wfxml/Default/service/registry/
   ```

   In the following URL, console is the context root for server connection, rbaserveri is the hostname and 80 is the port.

   ```
   http://rbaserver:80/console/_wfxml/Default/service/registry/
   ```

   Note

   Through a Server Project, you can access only the System application in the Default tenant.

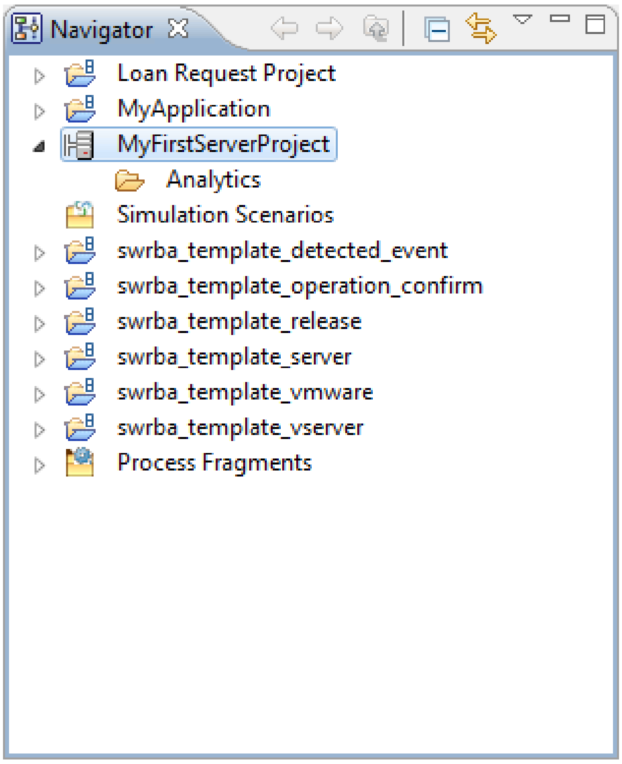
6. Enter the user ID used to authenticate with Systemwalker Runbook Automation in the **User ID** field, and enter the associated password in the **Password** field.

7. Optional: Check the **Save Password** check box if you want to save the specified password so that you do not need to provide it again when you log in to the Management Server.

8. Click **Finish**.

   The new server project is listed in the **Navigator** view (MyFirstServerProject, in the example):

   Figure 3.26 Displaying the new server project

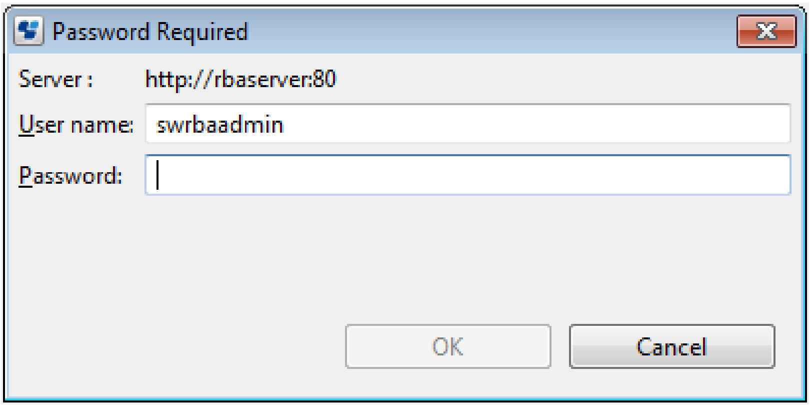
   The default server project contains an empty **Analytics** folder that can be used as an access point for the Analytics application.

   Note

   Systemwalker Runbook Automation Studio does not use the Analytics function.

9. To actually list the process definitions on the server in the **Navigator** view, you need to login to the server: Right-click the new server project in the **Navigator** view and select **Login**. If you saved your password when you created the server project, you are instantly logged in and the process definitions are listed. Otherwise, a dialog is opened in which you must enter the password.

   Figure 3.27 Entering a Server Password

   Server : http://rbaserver:80

   User name: swrbaadmin

   Password:

   To log off from the server again, right-click the server project and select **Logoff**.

Note

If your process definitions reference external files like forms, business calendars, cascading style sheets, Java classes, or rules files, ensure that these files are available on the computer on which Systemwalker Runbook Automation Studio is installed and that any file paths are adjusted. Refer to the following sections for details.

- "Chapter 8 Using Forms"
- "6.22.4 Creating Your Own Business Calendars"
- "11.6 Using Generic Java Actions"
- "Chapter 10 Decision Tables"

Note

If logging in to the Management Server fails, the following error message may be displayed:

Login failed. [Details]: Connection refused:connect

or

Login failed. [Details]: Connection timed out:connect

In this case:

- Ensure that the connection information required to access the Management Server is correct (Host Name, Port Number and Base URL).

  The connection information is displayed on the **Server Info** page in the **Properties for server** dialog. To open this dialog, right-click the server project in the **Navigator** view and select **Properties** from the pop-up menu.

- Check whether the server is started.

3.2.2 Closing Server Projects

You can close server projects in the same manner described for Workflow Application projects.

To close a server project:

- Right click the project in the **Navigator** view. Select **Close Project** from the pop-up menu.

The icon 📁 left to the project name indicates that it is closed.

You can close several projects at once. Hold down the <Shift> key or <Ctrl> key to select all projects to be closed.

3.2.3 Opening Server Projects

You can reopen a server project that is currently closed. The icon left to the project name indicates whether it is closed (📁) or open (📂).

To open a server project:

- Right click the project in the **Navigator** view. Select **Open Project** from the pop-up menu.

3.2.4 Removing Server Projects

You can remove server projects with all of their contents from Systemwalker Runbook Automation Studio.

Note

When you remove a server project, this means that the registered Management Server is also removed and the process definitions located on the server are no longer accessible.

To remove a server project:

1. Right click the project in the **Navigator** view. Select **Delete** from the pop-up menu.
2. Confirm the removal of the project by clicking **Yes**.

Chapter 4 Managing Process Definitions

This chapter explains how to create new process definitions and work with them.

4.1 Creating Process Definitions

This section explains the procedure for creating a new process definition.

Prerequisite:

- You have created a project where the process definitions can be stored.

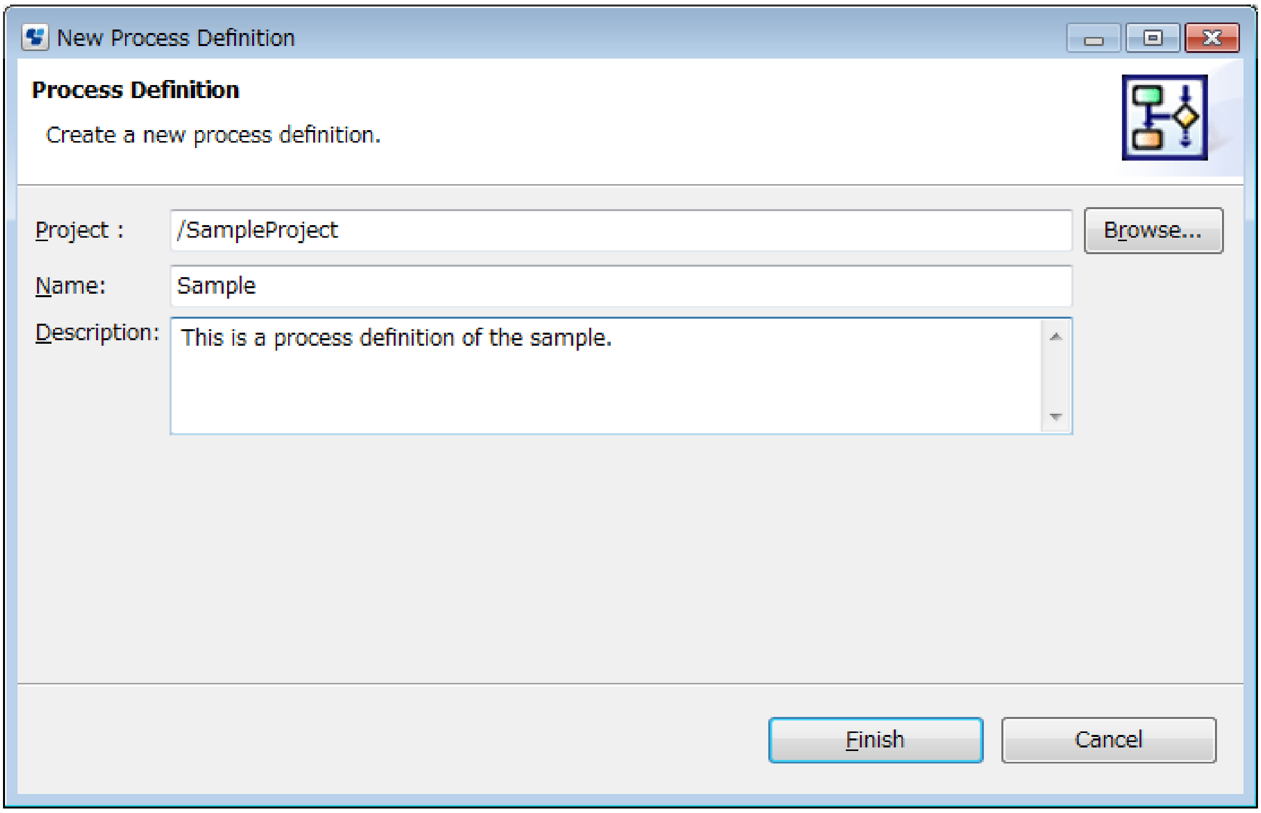
To create a process definition:

1. Select **File** > **New** > **Process Definition**.
2. In the **New Process Definition** dialog, click **Browse**. Select the project where the process definition is to be stored, and click **OK**.

   The project name is displayed in the **Project** field.
3. Type a name for your process definition in the **Name** field.

   If a file with this name already exists, a dialog is displayed asking you for a file name. Type a new file name and click **OK**.
4. Enter a description for the process definition in the **Description** field.

   Figure 4.1 Creating a Process Definition

   New Process Definition

   Process Definition

   Create a new process definition.

   Project : /SampleProject

   Name: Sample

   Description: This is a process definition of the sample.

5. Click **Finish**.

The new process definition is listed in the Navigator view. A Process Definition editor is opened and a Start Node is automatically added.

You can now define a new name and change the description for the process definition and start modeling your process.

Note

When creating a process definition in a server project, its application ID is set to 'System'.

4.2 Validating Process Definitions

Validating a process definition helps identify errors in the process definition, and the arrows, nodes and timers comprising it.

Prerequisite:

- You have created a project where the process definitions can be stored.

Note

Java actions and triggers are not validated by this process.

Note

Process definitions that are part of a server project cannot be validated.

To validate a process definition:

1. Do one of the following:
    - Click the Process Definition editor to make it active and click Validate on the toolbar.
    - Right click the empty space in the Process Definition editor and select **Validate** from the pop-up menu.

Any errors in the process definition will be displayed in the **Problems** view. For more information about **Problems** view refer to section 2.2.10 Problems View.

4.3 Saving Process Definitions

You can save process definitions that have been modified. If there are no changes, the save functions are not active.

To save process definitions:

1. Do one of the following:
    - To save the process definition that is currently displayed in the Process Definition editor, select **File** > **Save**.
    - To save all process definitions that have been modified, select **File** > **Save All**.
2. If a process definition is not valid, a message is displayed telling you so. You can display detailed information on the errors that have been found. You have the following options:
    - You can save the process definition anyway by clicking **Yes**.
    - You can return to Systemwalker Runbook Automation Studio without saving the definition by clicking **No**. You can then fix the errors and try saving the process definition again.

4.4 Saving a Process Definition to a Different Name or Project

To save a process definition to a different name or project:

1. Click the Process Definition editor that displays the process definition to be saved.
2. Select **File** > **Save As**.

3. If the process definition is not valid, a message is displayed telling you so. You can display detailed information on the errors that have been found. You have the following options:

   - You can save the process definition anyway by clicking **Yes**.

   - You can return to Systemwalker Runbook Automation Studio without saving the definition by clicking **No**. You can then fix the errors and try saving the process definition again.

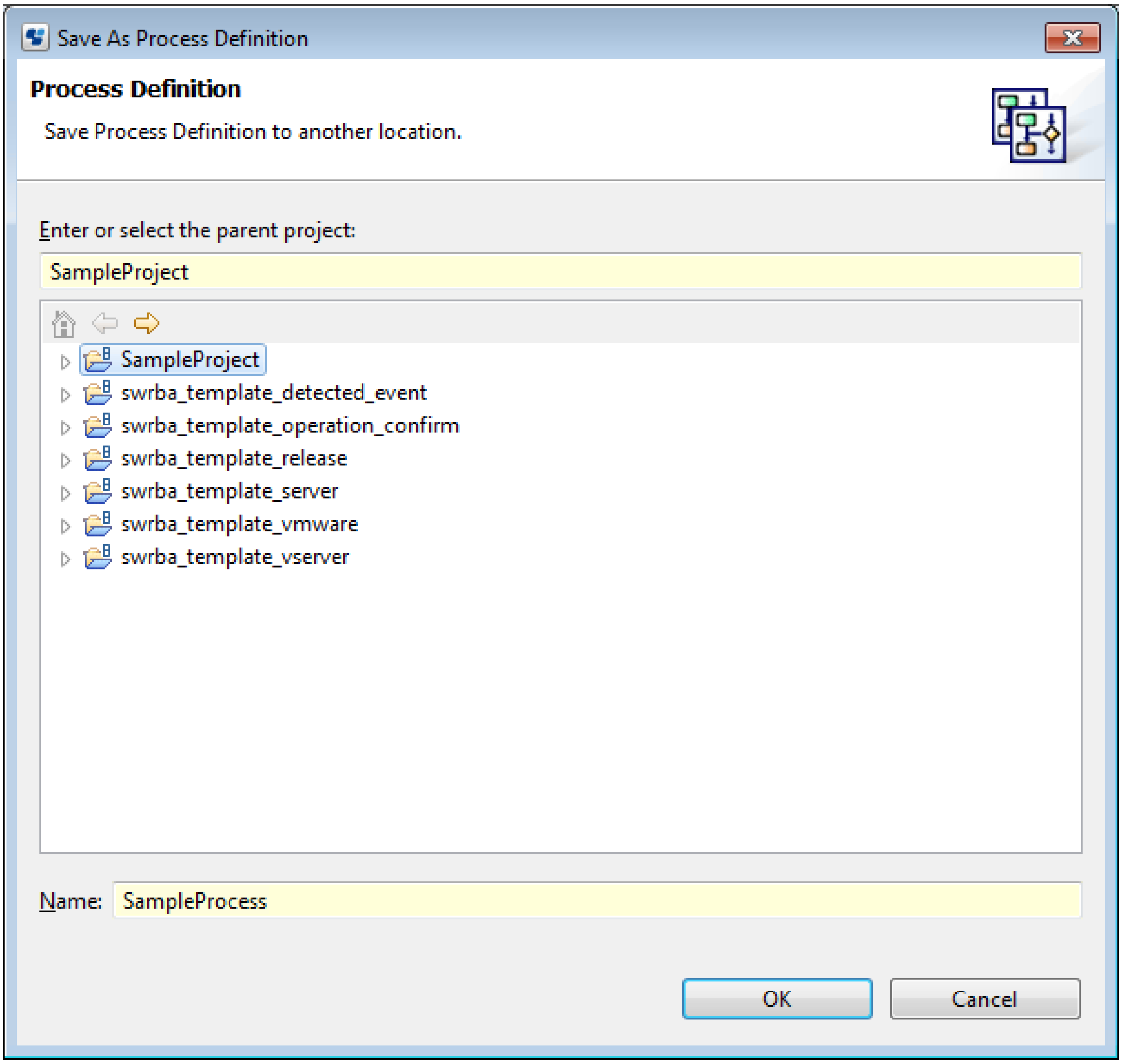
4. In the **Save As Process Definition** dialog, select the project where you want to save the process definition. You can save it to any Workflow Application or other local project, or to a server project to which you are currently logged in. You cannot save the process definition to the "Simulation Scenarios" project.

   Note

   When saving a process definition to a server project, its application ID is set to 'System'.

5. If you want to save the process definition to a different name, type the new name in the **Name** field.

   Figure 4.2 Saving a Process Definition to a Different Name or Project

   Save As Process Definition

   Process Definition

   Save Process Definition to another location.

   Enter or select the parent project: SampleProject

   - SampleProject
   - swrba_template_detected_event
   - swrba_template_operation_confirm
   - swrba_template_release
   - swrba_template_server
   - swrba_template_vmware
   - swrba_template_vserver

   Name: SampleProcess

6. Click **OK**.

4.5 Changing the Process Definition Name and Description

The name of a process definition is used to identify Automated Operation Process on Management Server, you need special attention to manage process definition. You can use a description to provide additional description information on the process.

Note

The name of process definitions contained in a server project cannot be changed.

To change a name and description for a process definition:

1. In the Navigator view, double click the process definition.
2. In the **General** tab of the Properties view, type the process definition's name.
3. If you want to provide more information about the process, type a description in the **General** tab of the Properties view.

4.6 Setting the Process Definition Priority

You can set priority for a process definition. By default, a process definition is given a medium priority of 8. However, its priority can be changed to an integer greater than or equal to 0.

To set the process definition priority:

1. Click the empty space in the Process Definition editor to display the Properties view for the process definition.
2. Select the **General** tab in the Properties view.
3. Enter the priority value in the **Priority** field. This will set the priority for the process definition.

4.7 Using Same Versions of Subprocess Definitions

For a parent process with subprocesses associated with it, there could be several versions of the parent process definition and the subprocess definitions. Depending on your design requirements, you may need to use specific versions of the subprocess definitions. In Interstage BPM Studio, while designing the processes, you can choose to use the versions of subprocesses same as that of the parent process definition.

If you do not update any of the process definitions and still choose to use the same versions of the subprocess definitions then the parent process definition will search for the subprocess definitions of the same version as its own. If they are available then the process instance will start or an error will be thrown.

For example, consider a parent process definition A of version V1 with two subprocess definitions namely B of version V3 and C of version V2 in a certain application APP01.

If you choose to use the same versions of the subprocess definitions and then update APP01, then versions of A, B, and C will become V4. This version is calculated as: **New Version = Maximum version number of the process definitions that will be updated together + 1**. In this scenario, the parent process definition A (having version V4) will search for the subprocess definitions having version V4. B and C will also have version V4 and the process instance will start.

While updating an application, if you want to use the same versions of the subprocess definitions, you need to update the parent process definition as well as the child process definitions.

Note

Ensure that for a version of a parent process definition, subprocess definitions having that same version exist.

Note

You can choose to use the same versions of only subprocesses and chained-processes.

Note

This function is effective when you select the parent process definition and the subprocess definition and execute the **Upload Application** command.

To use the same versions of subprocesses:

1. Click the **General** tab in the **Properties** view of the parent process definition.
2. Select the **Use the same subprocess definition version** checkbox.

4.8 Using Process Definition Reports

You can create reports of the Process Definitions in HTML, PDF and MS-PowerPoint formats. This feature is useful for the users who need an overview of their systems. They do not need to understand the technical details of the Interstage BPM. Their need is to understand the business-specific details about the systems that have been designed using Interstage BPM. For example, a manager of a bank will be interested in understanding the details about the bank's credit card and loan systems. S/he may not be interested in understanding the Interstage BPM-specific details pertaining to his/her systems.

You can create reports for the following:

- Process Definition reports can be generated from Process Definition Editor as well as Process Outline Editor.

The report for any of the above contains the following details:

- Process Definition Name
- Process Definition Description
- Process Definition Creation Date
- Snapshot of the Process Definition (only in Process Definition Editor)
- Description of the following nodes in the Process Definition: Activity, Voting Activity, Compound Activity, Subprocess

Details of a node that are included in the report are:

- Node Name
- Node Description
- User who has been assigned the activity
- Due Date of the activity

An activity in Interstage BPM Studio is referred as a task in the report.

The PDF and HTML reports contain the details in tabular form. Details of each activity and compound activity are listed in separate tables.

The MS-PowerPoint report generates separate slides for each major activity (node) and compound activity in the process.
Details about the child nodes of a compound activity are included in one slide of the MS-PowerPoint.

Note

The reports also contain **Appendix** which includes the day and timer codes for the due dates of the tasks. Refer 6.22.3 Time and Day Codes for Advanced Due Dates and Timers to know about day and timer codes.

Note

The reports do not contain information about triggers, timers, Java Actions, agents, and UDAs of the process.

Note

The reports contain details about only those tasks which need human intervention. These are Activity Nodes and Voting Activity Nodes.

The reports that are generated are saved in the **documents** folder of the application. By default, report is saved with the name that you enter in the **Title** field of the **Generate Report** dialog. You can rename the reports later.

You can perform the following operations on this folder:

- Copy
- Paste
- Rename
- Delete
- Import

Note

You can not create PDF document.

4.8.1 Creating Process Definition Reports

This section explains the procedure for creating a process definition report.

Prerequisites:

- **Generate Process Documentation** in **File** menu is enabled when the Process Definition Editor or the Process Outline Editor is active.
- The Process Definition has been saved.

To create a report of a Process Definition:

1. Open the Process Definition for which you want to create a report, from the **Navigator** view.
2. Click **File** menu and select **Generate Process Documentation**.

   Generate Report dialog is displayed.
3. Enter the name of the report in the **Title** field.
4. Select the appropriate checkbox to select the format of the report. The format options are **HTML**, **PDF** and **MS-PowerPoint**. You can select more than one option to generate reports in multiple formats.
5. Click **Save**.

   The report is generated and saved in the **Documents** folder in the **Navigator** view. A message is displayed stating that the report has successfully been saved in the **Documents** folder. If saving the report fails, an error message is displayed.

Note

If the Process Definition contains compound activity then the snapshot of the child process will not be generated in the report.

Note

When the Process Definition of Server Project is opened, **Generate Process Documentation** cannot be used.

Note

When **Generate Process Documentation** is executed by the Process Definition Editor and HTML report is created, a snapshot of the Process Definition is included in the report of the process definition. The file name of this snapshot is generated by adding **.jpg** as a file extension at the end of the name of the report file.

Note

When the HTML report is created by **Generate Process Documentation**, the _IBPM_STUDIO_Stype.css file is created. This file is the style sheet file of HTML report. You must not rename this file. If you decide to move the HTML report file to another folder, ensure you move this style sheet file as well.

4.9 Opening Process Definitions

To open a process definition, do one of the following:

- In the Navigator view, double click the process definition.
- In the Navigator view, right click the process definition and select **Open** from the pop-up menu.

For process definitions stored in a project located on a Management Server, you have the additional option of opening a specific version of it:

1. In the Navigator view, right click the process definition and select **Open With Version** from the pop-up menu.
2. Select the desired version in the dialog and click **OK**.

4.10 Importing Process Definitions

Process definitions can be imported to Systemwalker Runbook Automation Studio using XPDL format.

Prerequisite:

- You have created a project where the process definition can be imported.

Note

When importing a process definition into a server project, its application ID is set to 'System'.

To import a process definition:

1. In the **Navigator** view, right-click the **Process Definitions** folder of the Workflow Application project where you want to import the process definition. Select **Import** from the pop-up menu.
2. Navigate to the location where the process definition is stored.
3. Select the process definition and click **Open**.
4. If a process definition with the same file name already exists in the project, a dialog is displayed asking you for a file name. Type a new file name and click **OK**.

Note

- If the process definition references external files like forms, cascading style sheets, Java classes, or rules files, ensure that these files are available on the machine where you import it. Refer to the following sections for details.
- "Chapter 8 Using Forms"
- "6.22.4 Creating Your Own Business Calendars"

- "11.6 Using Generic Java Actions"
- "Chapter 10 Decision Tables"

4.11 Exporting Process Definitions

You can export process definitions from Systemwalker Runbook Automation Studio to the file system. You can use the exported files, for example, to import them into other systems. For the export format, you can choose between XPDL 1.0, XPDL 2.0, and XPDL 2.1.

Note

You can export entire Workflow Application projects including all process definitions, forms, attachments, etc. in one step, and later deploy the application on a Management Server.

Refer to section 3.1.13 Exporting Workflow Application Projects for details.

To export a process definition:

1. In the Navigator view, right click the process definition. Select **Export** and then the desired XPDL format.
2. Navigate to the location where the exported process definition is to be stored.
3. Click **Save**.

You can import process definitions into a Management Server by creating a server project (refer *Creating Server Projects* to know how you can create server projects) and transferring them to the server project, or by using the Web Console.

Note

- If the process definition references external files like forms, business calendars, cascading style sheets, Java classes, or rules files, ensure that these files are available on the machine to which you export it. Refer to the following sections for details.
- "Chapter 8 Using Forms"
- "6.22.4 Creating Your Own Business Calendars"
- "11.6 Using Generic Java Actions"
- "Chapter 10 Decision Tables"

4.12 Sending Process Definitions to a Server

This section explains the procedure for sending process definitions to servers.

Prerequisites:

- You have locally created one or more process definitions in Workflow Application projects.
- One or more Systemwalker Runbook Automation Servers are connected.

  Example) You have defined one or more server projects and are logged in to the respective server(s).

Note

When sending a process definition to a server project, its application ID is set to 'System'.

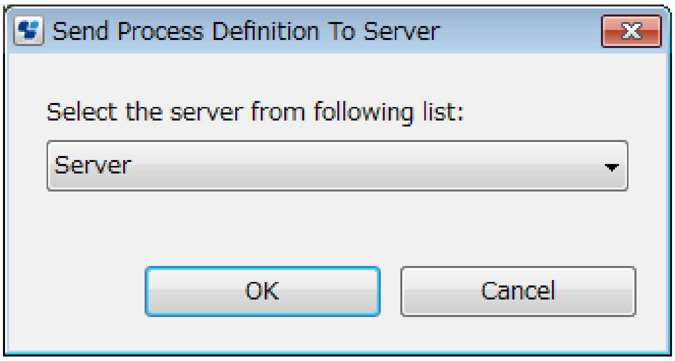
To send a local process definition to the server:

1. In the **Navigator** view, right-click the process definition that is to be uploaded to a Management Server, and select **Send to Server**.

   You can select multiple process definitions in the same project using the <Shift> or <Ctrl> keys.

2. Select the server project from the drop-down list that shows all defined server projects.

   Figure 4.3 Sending Process Definitions to the Server

   Send Process Definition To Server

   Select the server from following list:

   Server

3. If you are currently not logged in to the selected server project, the **Password Required** dialog is displayed. Enter the required password and click **OK**.
   The display of the **Password Required** dialog depends upon whether you selected the **Save Password** checkbox when creating the server project. Refer to section 3.2.1 Creating Server Projects for details.

4. The local process definition is uploaded to the server and listed in the selected server project. If the process definition already exists on the server, a new version is created. You can access a specific version of the process definition using the **Open With Version** function.

If the uploading to the server fails, a red icon is displayed aside the process definition name in the Navigator view. The process definition is saved to the specified server project, but not yet available. You need to retry sending this process definition to the server by selecting **Send to Server** again. If successful, the process definition is instantly transferred to the server you specified when you first tried to send it.

Note

- If your local process definition references external files like forms, business calendars, cascading style sheets, Java classes, or rules files, ensure that these files are available on the Management Server. Refer to the following sections for details.
- "Chapter 8 Using Forms"
- "6.22.4 Creating Your Own Business Calendars"
- "11.6 Using Generic Java Actions"
- "Chapter 10 Decision Tables"

4.13 Copying Process Definitions

You can copy process definitions in the Navigator view.

Note

You can use this function for local process definitions contained in Workflow Application projects only. Process definitions contained in a server project cannot be copied.

To copy a process definition:

1. In the Navigator view, right click the process definition that you want to copy and select **Copy** from the pop-up menu.

2. Right click the project where you want to paste the process definition and select **Paste** from the pop-up menu.

3. If you copy within a project and a process definition using the same file name already exists, a dialog is displayed asking you for a file name. Type a new file name and click **OK**.

4. If you copy between projects and a process definition using the same file name already exists, a dialog is displayed telling you so. Do one of the following:

    - To overwrite the existing process definition, click **Yes**.

    - To keep the existing process definition, click **No**.

    In this case, the process definition is not pasted.

You can copy several process definitions at once. Hold down the <Shift> key or <Ctrl> key while selecting all process definitions to be copied.

Note

When copy process definition, process definition name is same as copied name, so also change process definition name.

If file name are different but process definition name are same, it is treated as a same process definition on Management Server. When register process definition on Management Server, if same process definition exist in target Automated Operation Process Group, existing process definition has gotten to updated.

For detail how to change process definition name, Refer to the ***Changing the Process Definition Name and Description***

4.14 Renaming Process Definitions file

You can specify a new file name for a process definition.

Note

You can use this function for local process definitions contained in Workflow Application projects only, and only if you set your preferences for the Navigator view to **File name**. Process definitions contained in a server project cannot be renamed.

To rename a process definition:

1. Right click the process definition in the Navigator view. Select **Rename** from the pop-up menu.

2. Type the new name for the process definition.

3. Press the <Enter> key.

4. If a process definition using the same file name already exists in that project, a dialog is displayed telling you so. Do one of the following:

    - To overwrite the existing process definition, click **Yes**.

    - To keep the existing process definition, click **No**.

    In this case, the process definition is not renamed.

Note

If you change file name for process definition, process definition name are not changed.

For the detail how to change process definition name, Refer to the ***Changing the Process Definition Name and Description***

4.15 Closing Process Definitions

To close process definitions:

1. Do one of the following:

   - To close a particular process definition, click the Close button of the Process Definition editor.

   - To close all process definitions, select **File** > **Close All**.

2. If there are unsaved changes, a message is displayed telling you so. Do one of the following:

   - To close the process definition and save your changes, click **Yes**.

   - To close the process definition without saving your changes, click **No**.

4.16 Removing Process Definitions

When you remove a process definition, it is removed from Systemwalker Runbook Automation Studio and from the file system as well.

Note

You can use this function for local process definitions contained in Workflow Application projects only. Process definitions contained in a server project cannot be removed.

To remove a process definition:

1. Right click the process definition in the Navigator view. Select **Delete** from the pop-up menu.

2. Click **Yes** to confirm the removal.

Chapter 5 Managing Operation Components

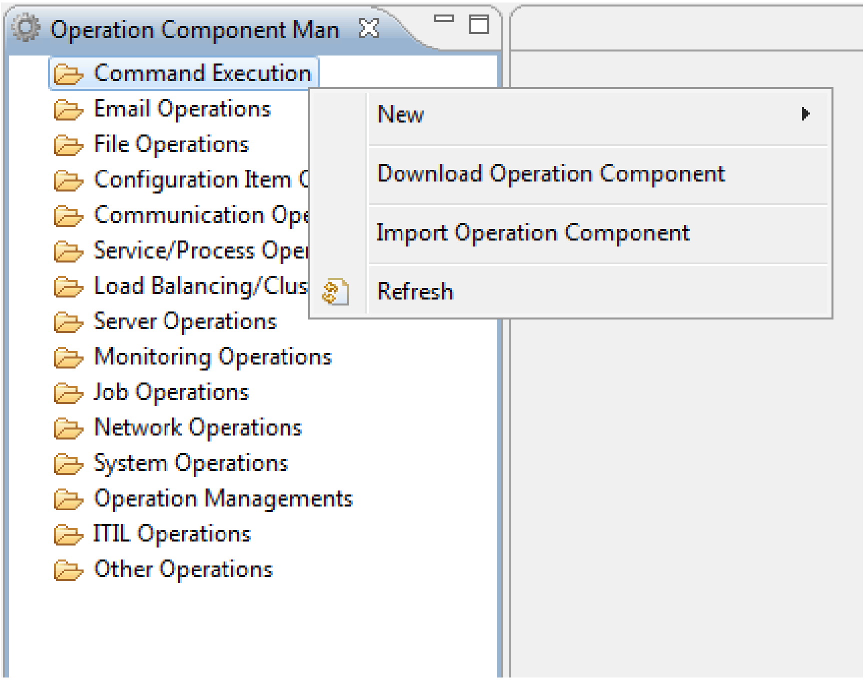
This chapter explains procedures for creating and managing operation components.

5.1 Creating Operation Component Projects

An operation component project is a container for managing the files that make up a single operation component.

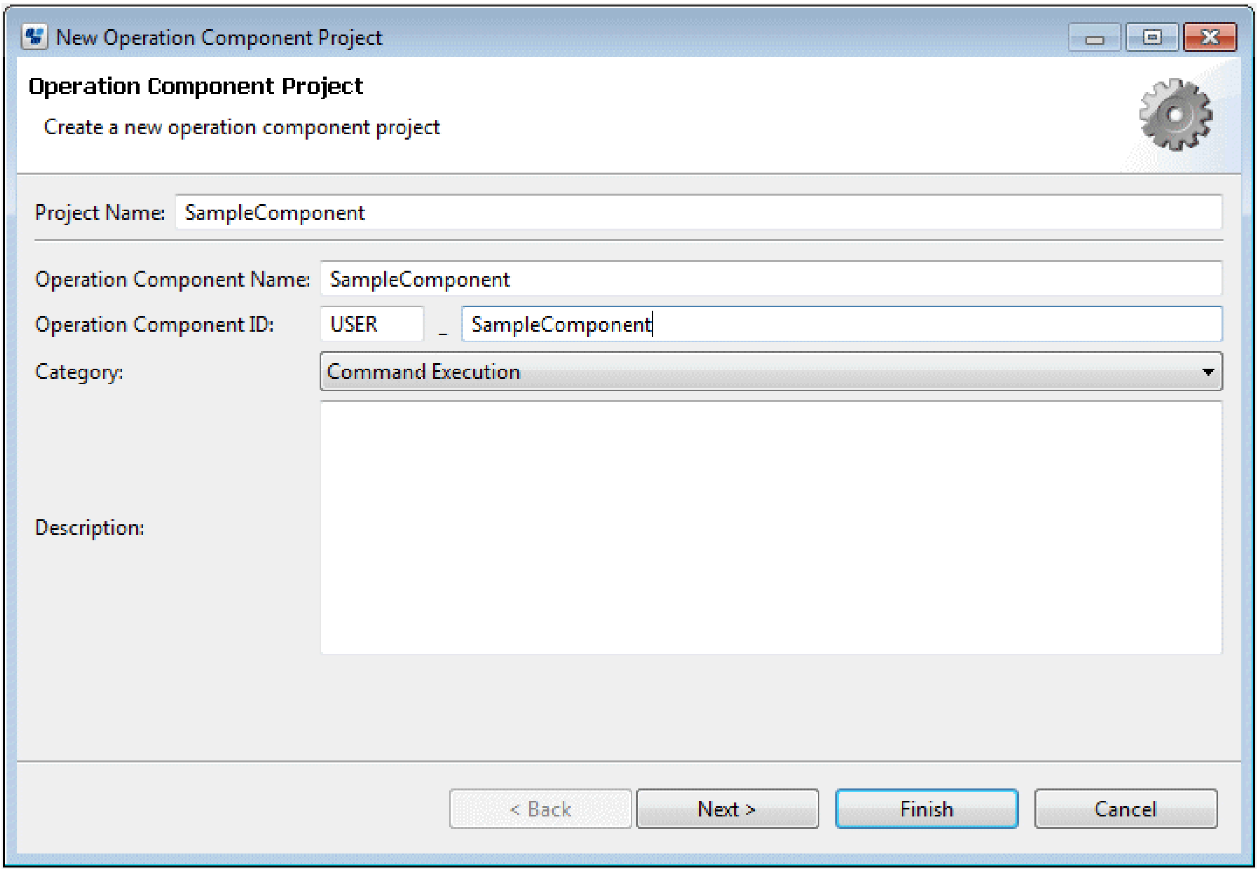
This section explains how to create an operation component project.

1. Right-click on a category in the **Operation Component Management** view.



2. Select **New** and then **Operation Component** from the pop-up menu.

   The **New Operation Component Project** dialog box will open.



New Operation Component Project

Operation Component Project

Create a new operation component project

Project Name: SampleComponent

Operation Component Name: SampleComponent

Operation Component ID: USER _ SampleComponent

Category: Command Execution

Description:

3. Enter a name up to 128 characters for the name of the project in the **Project Name** field.

   The project name must be unique on the workspace - it cannot be the same as another project on the workspace, for example an application project.

4. Enter a name up to 61 characters for the operation component in the **Operation Component Name:** field.

5. In the **Operation Component ID:** field, enter an ID to uniquely identify the operation component. In the field on the left, enter a prefix of up to five letters for the ID. In the field on the right, enter up to 55 characters (alphanumerics and underscores (_)).

Note

- "SWRBA" cannot be used as a prefix for operation component IDs.

- The specified operation component name is used as the default name prefix when the operation component node is placed in the process definition using the Process Definition Editor.

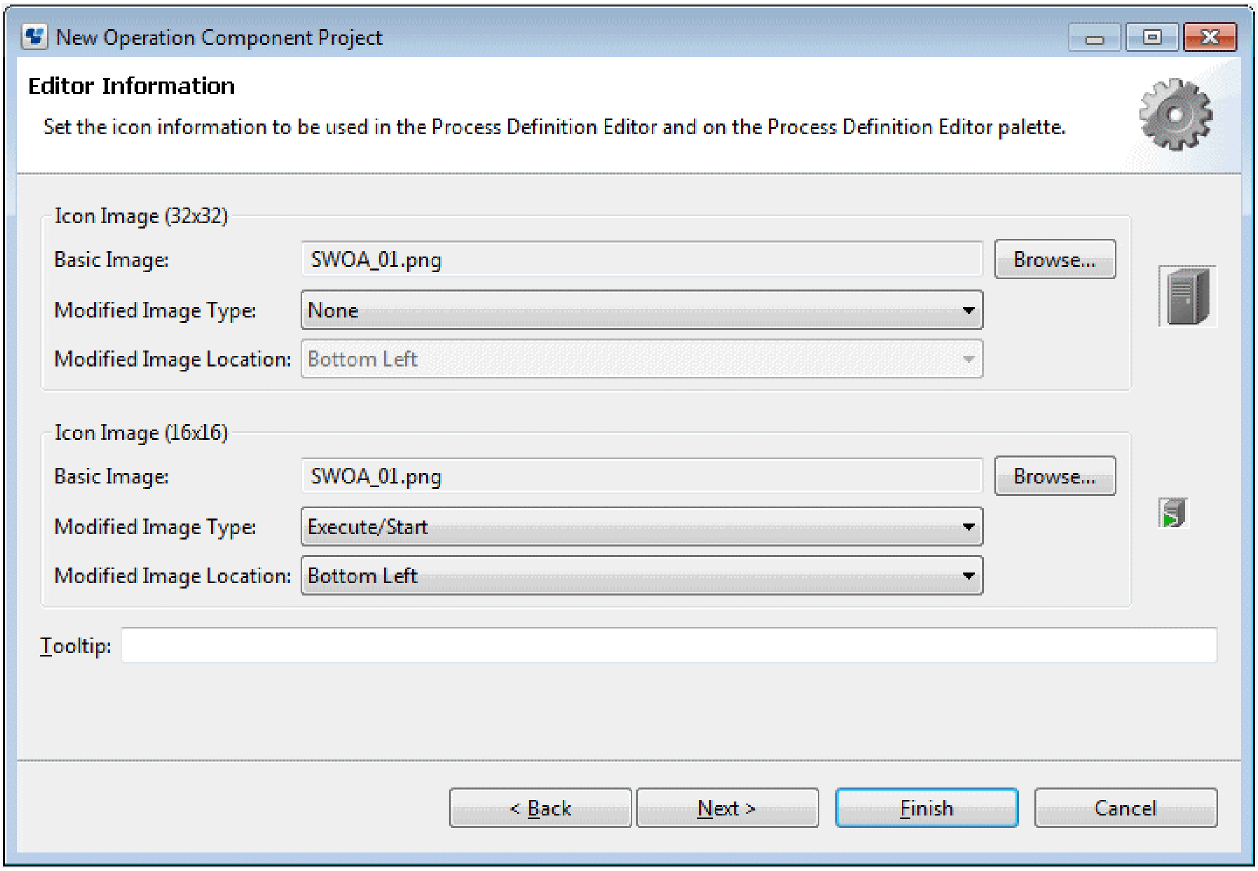
6. Use the **Category** combo-box to select a category for the operation component.

7. Enter a description of up to 4096 characters for the operation component in the **Description** field.

8. Click the **Next** button.

   Note: The Finish button may also be clicked here.

9. Select the image file for the icon to be used in the Process Definition Editor or in the palette of Process Definition Editor.



New Operation Component Project

Editor Information

Set the icon information to be used in the Process Definition Editor and on the Process Definition Editor palette.

Icon Image (32x32)

Basic Image: SWOA_01.png

Modified Image Type: None

Modified Image Location: Bottom Left

Icon Image (16x16)

Basic Image: SWOA_01.png

Modified Image Type: Execute/Start

Modified Image Location: Bottom Left

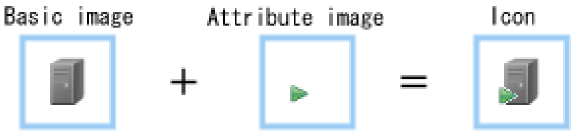
Tooltip:

10. Click the **Browse** button of the icon image (32x32) group and select the icon image file. (Optional)

The following icon is specified by default.

Point

Icons displayed in the Process Definition Editor are generated automatically by overlaying the basic images specified as icon images (32x32) and the other attribute images.



11. From the Modified Image Type combo-box, select the icon image type that will show the status.

The attribute image can be selected from the following 20 types.

Type	Image
Execute, Start	▶
Re-execute	↻
Stop	■
Pause	❚❚

Type	Image
Create, Add	✚
Delete	✕
Confirm, Monitor	✓
Search	🔍
Get, Browse	👁
Set, Modify, Update	✏
Transfer	➡
Convert/Exchange	✖
Connect	⚡
Disconnect	✖⚡
Information/Details/Properties	▤
List/Table	☰
Restore	↶
Communication/Conversation	🌐
Print	🖨
Repeat	🔁

12. From the Modified Image Location combo-box, select the location for the overlay of the attribute image on the basic image.

    The four locations that can be used for overlay for the basic image are the basic image's top right, top left, bottom left,bottom right,. If None is selected for Modified Image Type, the Modified Image Location combo box will be grayed out and cannot be selected.

13. In the same way, from the Icon Image(16x16) group specifyBasic Image, Modified Image Type and Modified Image Location. (Optional)

    The following icon is specified by default.

    ▤

14. In the Tooltip field, enter a simple description about the operation component. (Optional)

    The description will be displayed as tooltip help when the cursor is aligned with the operation component node in the Process Definition Editor Palette.

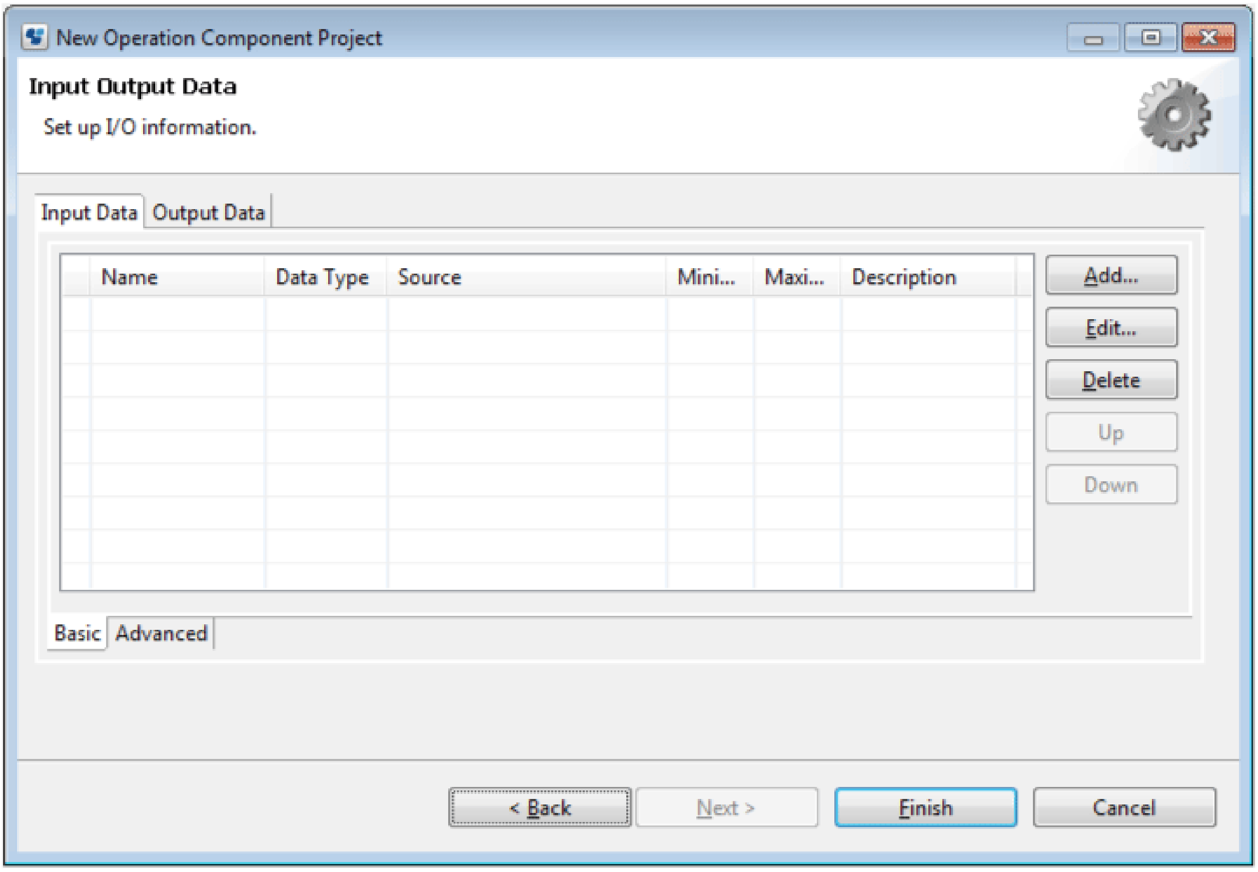
15. Click the **Next** button.

    Note: The Finish button may also be clicked here.

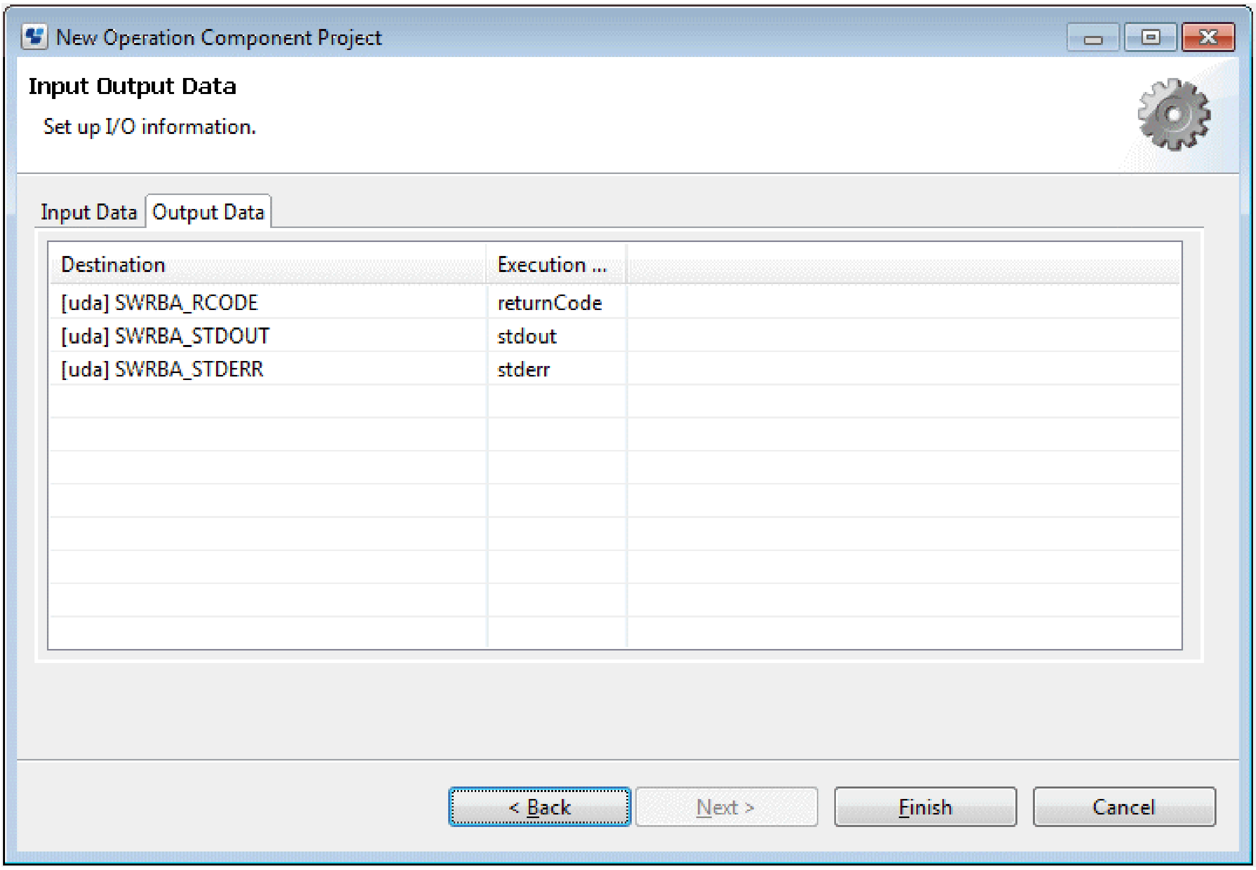
16. In the Input Output Data window, set the I/O information for executing the operation component.

    When the Input Data tab is selected, this will change to the window used for setting the input information. When the Output Data tab is selected, the output information settings can be referenced.

When the Basic tab of Input Data is selected, the basic parameters can be set. When the Advanced tab of Input Data is selected, the extended parameters can be set.

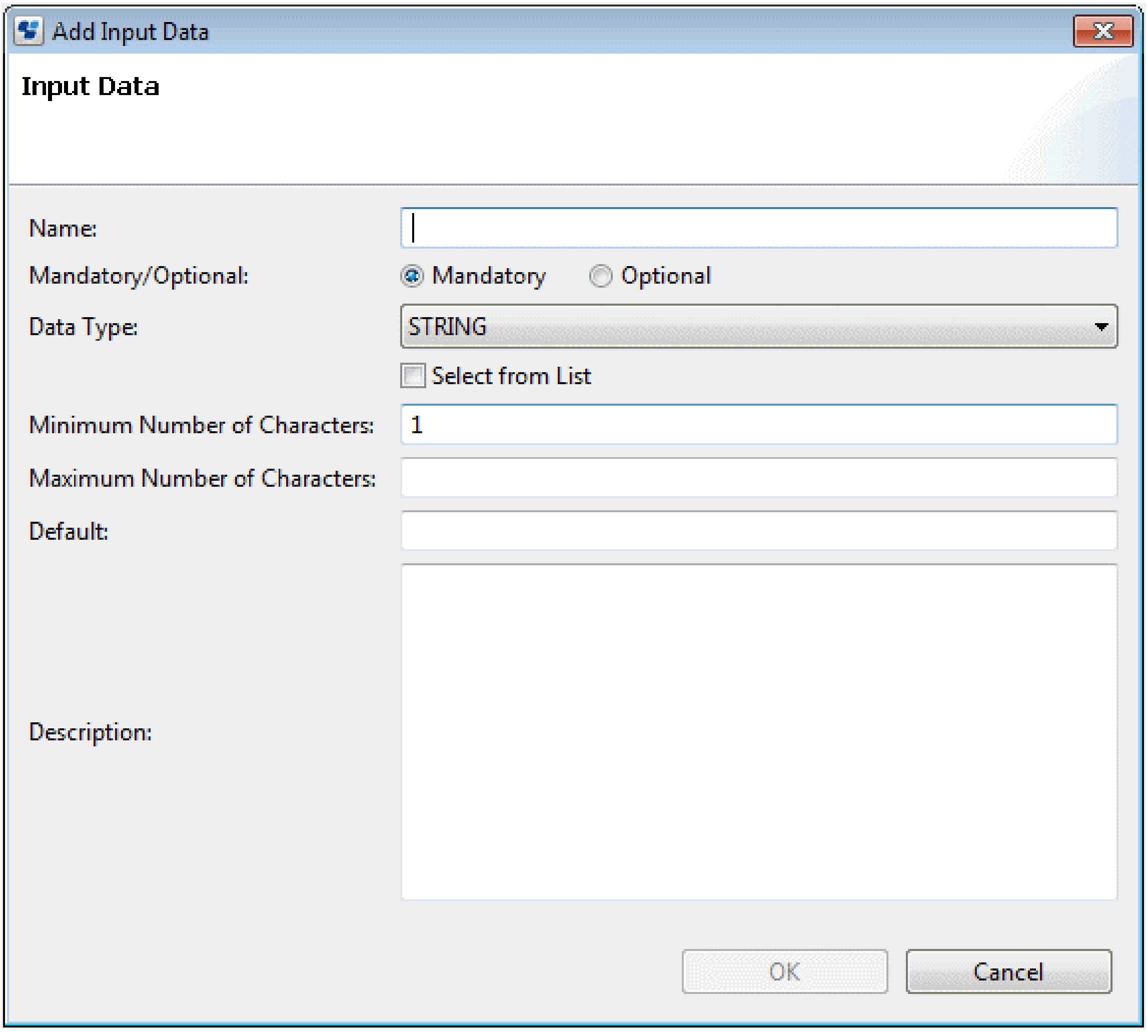


Name	Data Type	Source	Mini...	Maxi...	Description



Destination	Execution ...
[uda] SWRBA_RCODE	returnCode
[uda] SWRBA_STDOUT	stdout
[uda] SWRBA_STDERR	stderr

To add/delete input information, click the **Add** and **Delete** buttons.



Name:

Mandatory/Optional: ● Mandatory ○ Optional

Data Type: STRING

□ Select from List

Minimum Number of Characters: 1

Maximum Number of Characters:

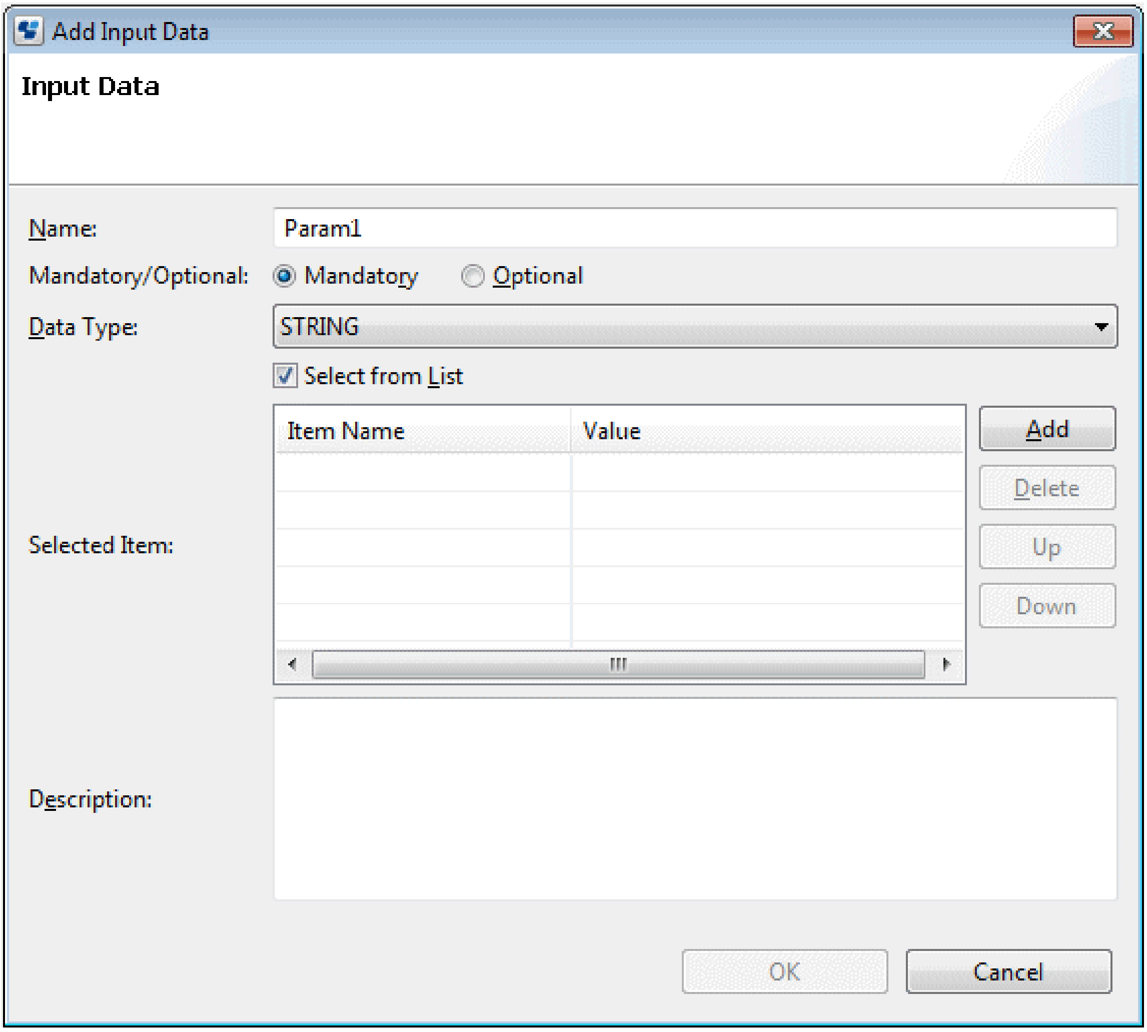
Default:

Description:

When the **Select from List** check box is selected, it will be possible to enter list format data.

When the **Add** button is clicked, the new item will be added to the end of the table control.

Item name: "NewItemText *number*", Value: "NewValue *number*" (If data type is INTEGER, it is "*number*"



Add Input Data

Input Data

Name: Param1

Mandatory/Optional: ● Mandatory ○ Optional

Data Type: STRING

☑ Select from List

Selected Item:

Item Name	Value

Description:

17. Click the **Finish** button.

5.2 Changing an Operation Component

This section explains how to change the information for an operation component.

1. Right-click on an operation component project in the **Operation Component Management** view. Select **Edit Operation Component [Japanese]** from the pop-up menu. To enter a English name or English description, select **Edit Operation Component [English]**.

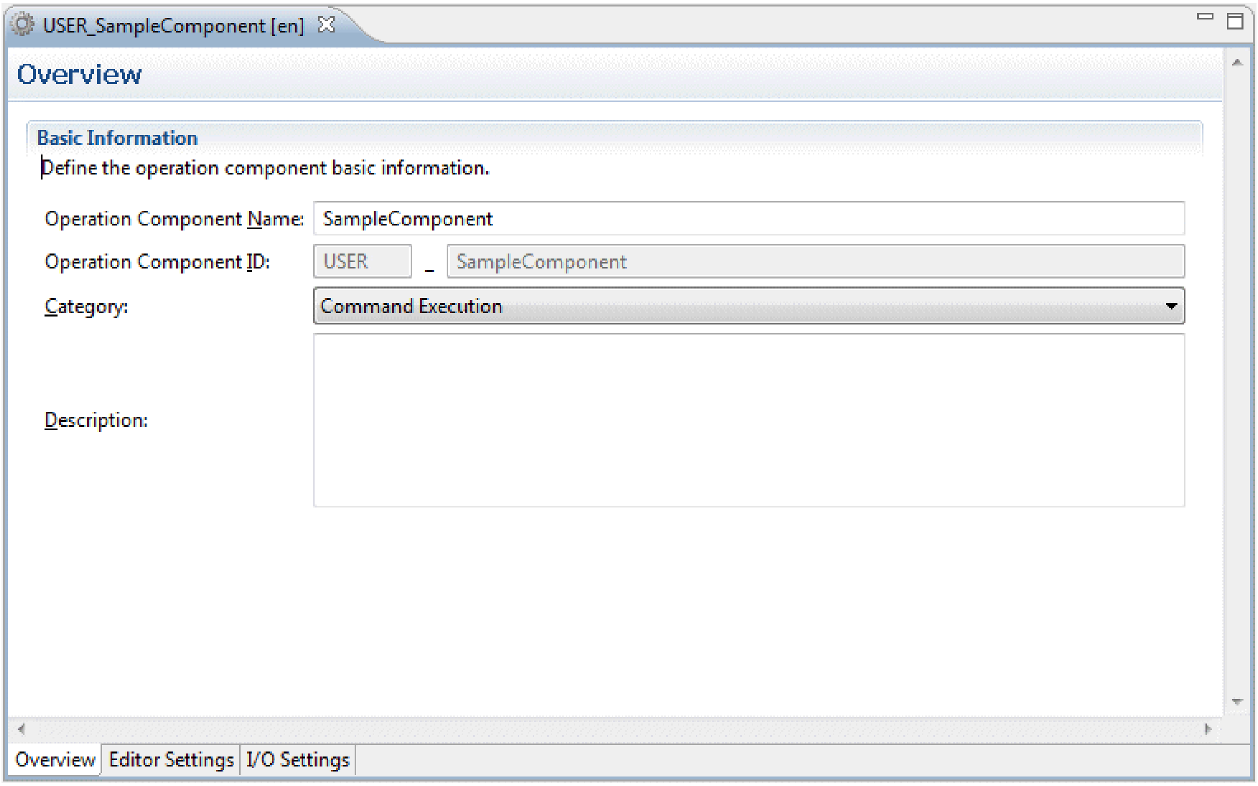
    - If the server where operation component execute in Japanese environment
      The information specified in **Edit Operation Component [Japanese]** will be used.

    - If the server where operation component execute in English environment
      The information specified in **Edit Operation Component [English]** will be used.

    The Editor that will be used to edit the operation component will open.

    On the **Overview** page, set the operation component name or category, and description.

The operation component ID cannot be changed.



Basic Information

Define the operation component basic information.

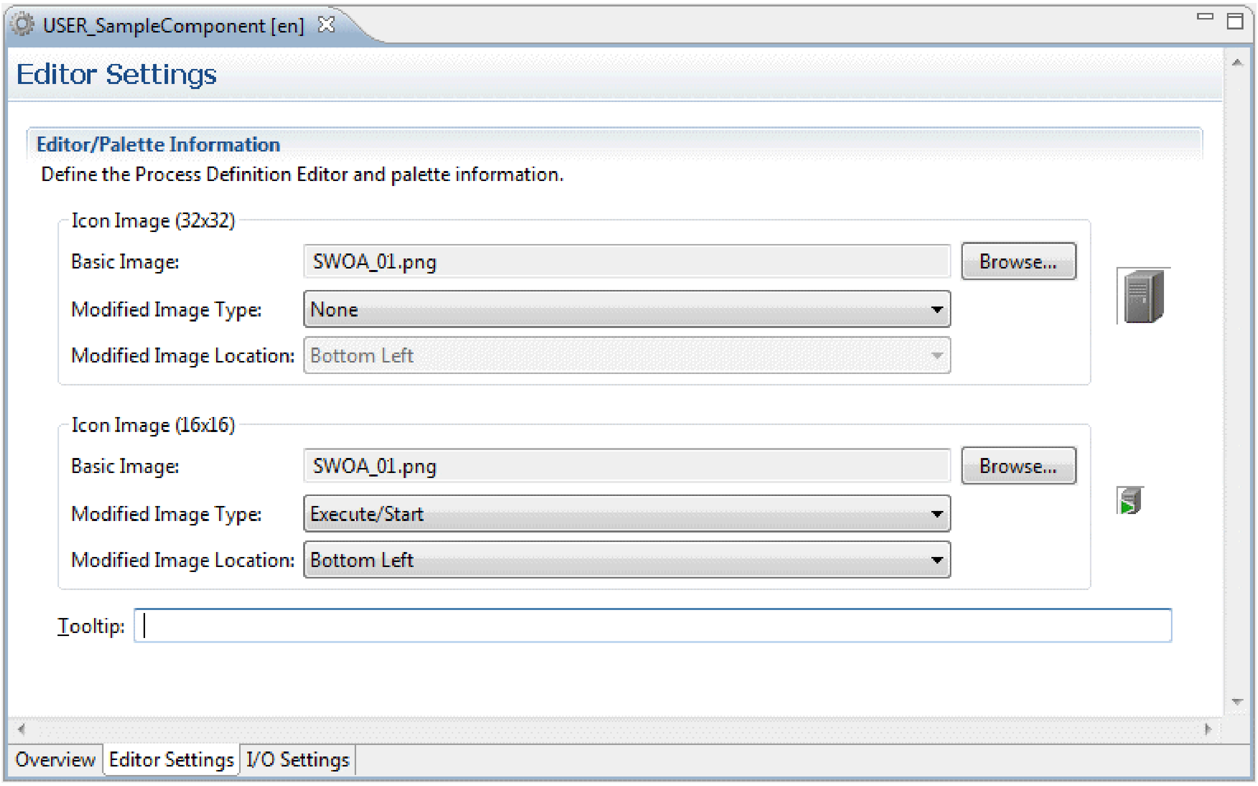
Operation Component Name: SampleComponent

Operation Component ID: USER _ SampleComponent

Category: Command Execution

Description:

2. On the **Editor Settings** page, set the icon information and tooltips that will be displayed in the Process Definition Editor and on the Process Definition Editor palette.



Editor/Palette Information

Define the Process Definition Editor and palette information.

Icon Image (32x32)

Basic Image: SWOA_01.png

Modified Image Type: None

Modified Image Location: Bottom Left

Icon Image (16x16)

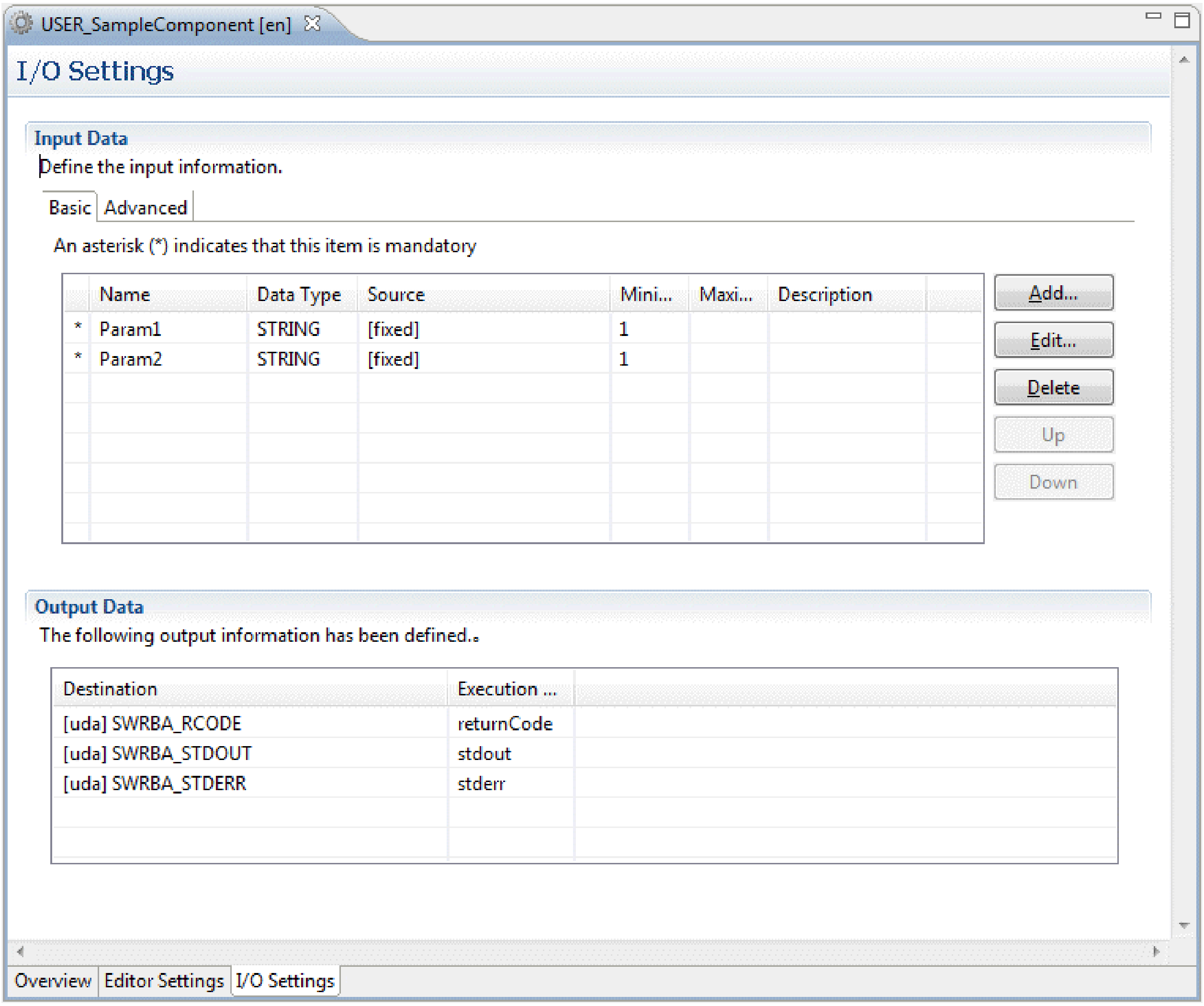
Basic Image: SWOA_01.png

Modified Image Type: Execute/Start

Modified Image Location: Bottom Left

Tooltip:

3. On the I/O Settings page, enter the I/O information for executing the operation component.



Input Data

Define the input information.

An asterisk (*) indicates that this item is mandatory

	Name	Data Type	Source	Mini...	Maxi...	Description
*	Param1	STRING	[fixed]	1		
*	Param2	STRING	[fixed]	1		

Output Data

The following output information has been defined.。

Destination	Execution ...
[uda] SWRBA_RCODE	returnCode
[uda] SWRBA_STDOUT	stdout
[uda] SWRBA_STDERR	stderr

4. Click the **x** button or press the **Files** >> **Save** to save the changes.

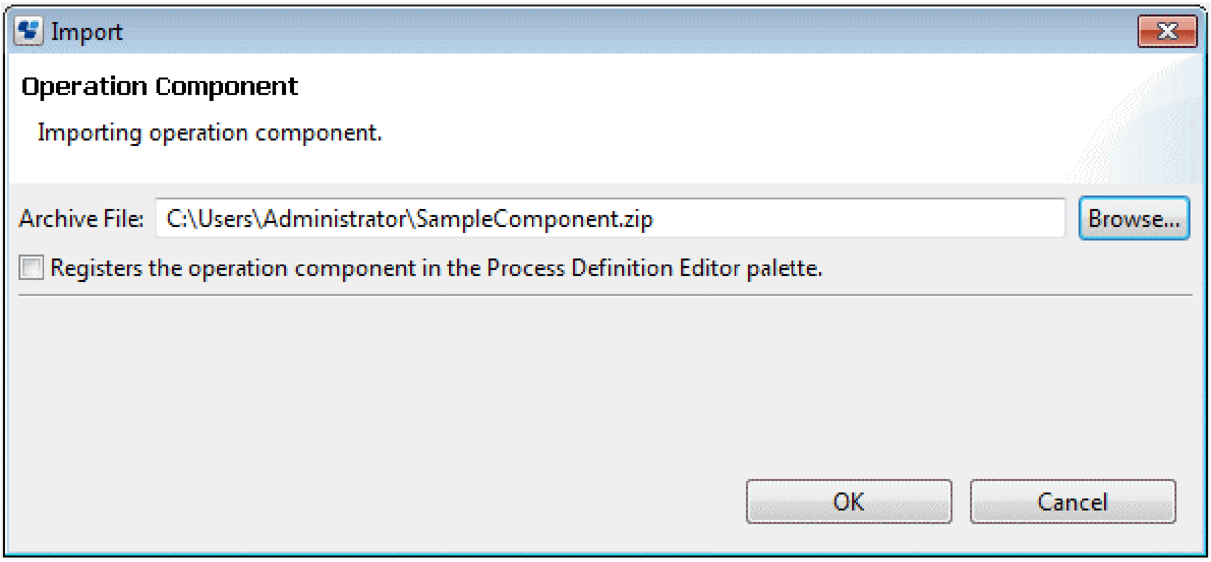
5.3 Importing Operation Component Projects

Packaged operation component projects can be imported into the local workspace.

To import an operation component from the local file system, use the following procedure:

1. In the **Operation Component Management** view, right-click on the category into which the project is to be imported. .Select **Import Operation Component** from the pop-up menu.

   The **Import Operation Component** dialog box will open.



Archive File: C:\Users\Administrator\SampleComponent.zip

□ Registers the operation component in the Process Definition Editor palette.

2. In the **Archive File:** field, enter the full path of the operation component archive file (.zip) to be imported. Alternatively, click the **Browse** button and select the operation component archive file (.zip) to be imported.

3. To register the imported operation components with the palette in the Process Definition Editor, select the **Registers the operation component in the Process Definition Editor palette.** check box.

4. Click the **OK** button.
   In any of the following condition, error messages has been displayed and you can not import.

   - When operation components having same operation component ID exist

   - When operation component project having same operation component name exist

   In the following case, **The name change of project** dialog will display.

   - When operation component project having same name exist

   In **The name change of project** dialog, change name and click **OK**, import will be executed with changing project name. If click **Cancel**, import will not be executed.

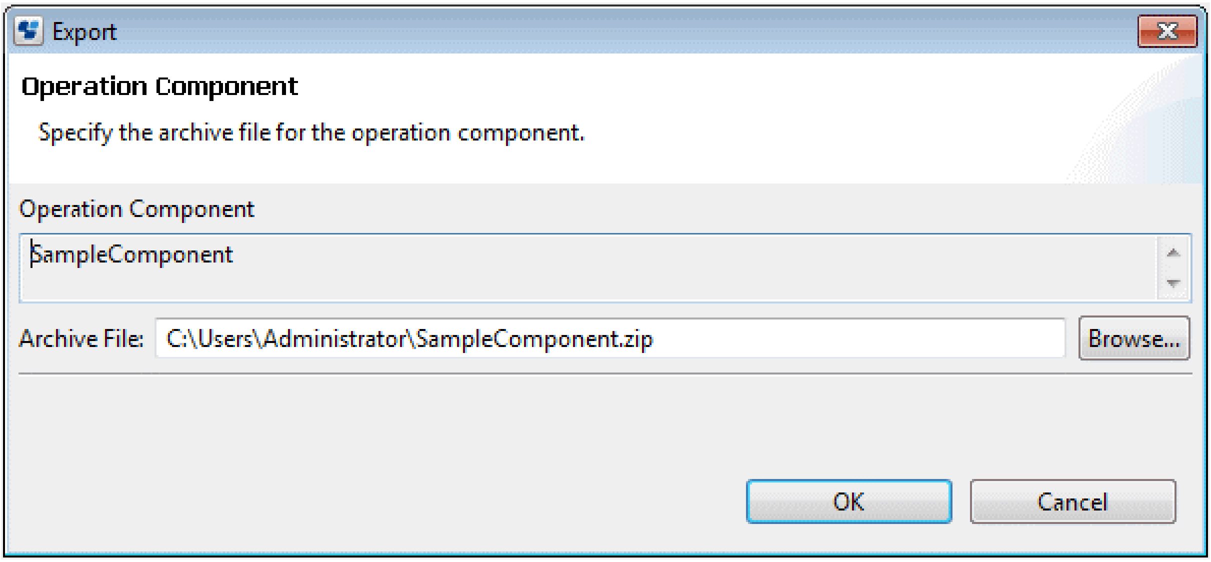
5.4 Exporting Operation Component Projects

It is possible to export operation components from the workspace for Systemwalker Runbook Automation Studio and package all of the files that make up the operation components into a zip file. This zip file can then be registered on the Management Server using a Web Console. They can also be imported into Systemwalker Runbook Automation Studio on another computer.

Use the following procedure to export operation components.

1. Right-click on an operation component project in the **Operation Component Management** view. Select **Export Operation Component** from the pop-up menu. Alternatively, the option can also be selected from the **Files** menu.

   The **Export Operation Component** dialog box will open.



Export

Operation Component

Specify the archive file for the operation component.

Operation Component

SampleComponent

Archive File: C:\Users\Administrator\SampleComponent.zip

2. In the **Archive File:** field, specify the name of the archive file to be exported.

3. Click the **OK** button.

   The operation component project will then be saved as the specified archive file.

5.5 Deleting Operation Component Projects

It is possible to delete projects and all of their contents from Systemwalker Runbook Automation Studio.

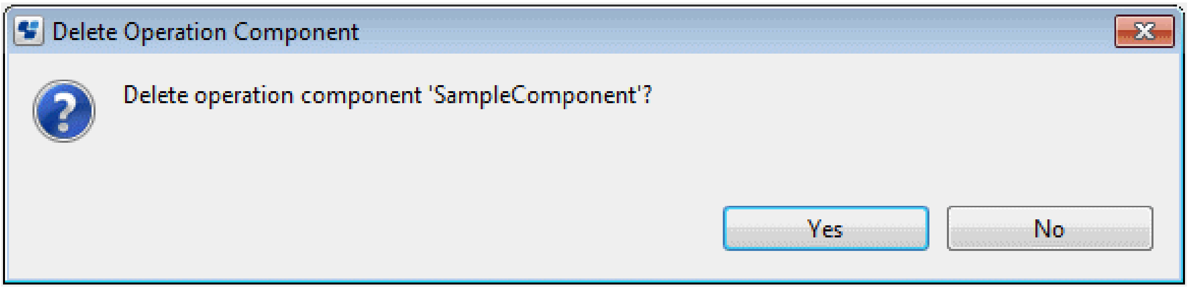
Note

- Once an operation component project is deleted, all of the files, such as a script file, contained in the project are also deleted from Systemwalker Runbook Automation Studio
- The project of operation component to be used in process definition of Systemwalker Runbook Automation Studio cannot be deleted. Before deleting project, it is used to delete from process definition.
- Operation component projects registered in the palette in the Process Definition Editor cannot be deleted. Before deleting the project, delete it from the Process Definition Editor's palette.

This section explains how to delete an operation component project.

1. Right-click on an operation component project in the **Operation Component Management** view. Select **Delete** from the pop-up menu.

   The **Delete Operation Component** dialog box will be displayed.



Delete Operation Component

Delete operation component 'SampleComponent'?

2. Click **Yes** to confirm the deletion.

   If **No** is clicked, the operation component project will not be deleted.

5.6 Uploading to the Operation Component Server

This section explains how to upload the operation component.

1. Right-click on an operation component project in the **Operation Component Management** view. Select **Upload Operation Component** from the pop-up menu. The **Upload Operation Component** dialog box will open. If the server connection information is not specified, an error message will be displayed in the dialog box.

2. Select "Management Server".

   Select the management server from the Server Connection drop-down list. Here, the server that was selected previously will be displayed automatically. If no servers have been selected previously, then the list will be empty. In this case, or to select another management server, use the following procedure:

   a. Click the Browse Connection button. The Select Server Connection dialog box will be displayed.

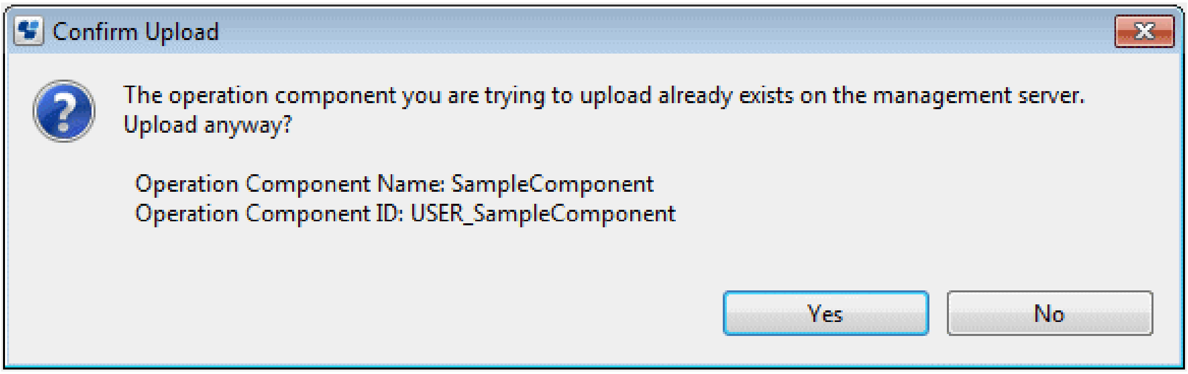
   b. When you want select another Management Server, select the server connection information from list,, then click the **OK** button.

   c. If it is needed to connect Management Server which not exist in **Select Server Connection** dialog, click **New** button,then enter the required items in the Setting Server Connection dialog box. Refer to section 2.4.1 Server Connection Settings for information on the server connection information.

3. To register the uploaded operation components with the palette in the Process Definition Editor, select the **Registers the operation component in the Process Definition Editor palette.** check box.

4. Click **Finish** to upload the operation component.

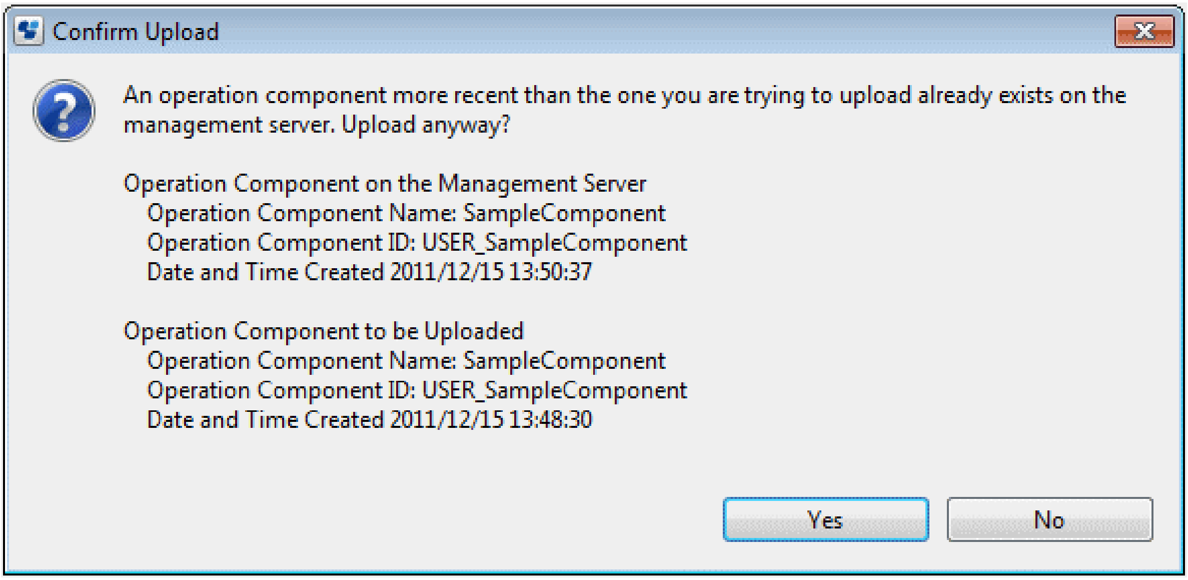
   If an operation component with the same ID exists on the management server, an update upload confirmation message will be displayed. Click **Yes** to update upload the operation component. If **No** is clicked, the operation component will not be uploaded.



The operation component you are trying to upload already exists on the management server.
Upload anyway?

Operation Component Name: SampleComponent
Operation Component ID: USER_SampleComponent

Additionally, if the date and time when the uploaded operation component was created is before that of the operation component on the management server, the following message will be displayed.



An operation component more recent than the one you are trying to upload already exists on the management server. Upload anyway?

Operation Component on the Management Server
Operation Component Name: SampleComponent
Operation Component ID: USER_SampleComponent
Date and Time Created 2011/12/15 13:50:37

Operation Component to be Uploaded
Operation Component Name: SampleComponent
Operation Component ID: USER_SampleComponent
Date and Time Created 2011/12/15 13:48:30

5.7 Downloading from the Operation Component Server

This section explains how to download the operation component from the management server.

1. Right-click on a category or an operation component project in the **Operation Component Management** view. Select **Download Operation Component** from the pop-up menu.

   The **Download Operation Component** dialog box will open. The server connection that had active status previously will be displayed automatically.

2. Select the server from the Server Connection drop-down list.

3. Click the Get List button in the **Download Operation Component** dialog box.

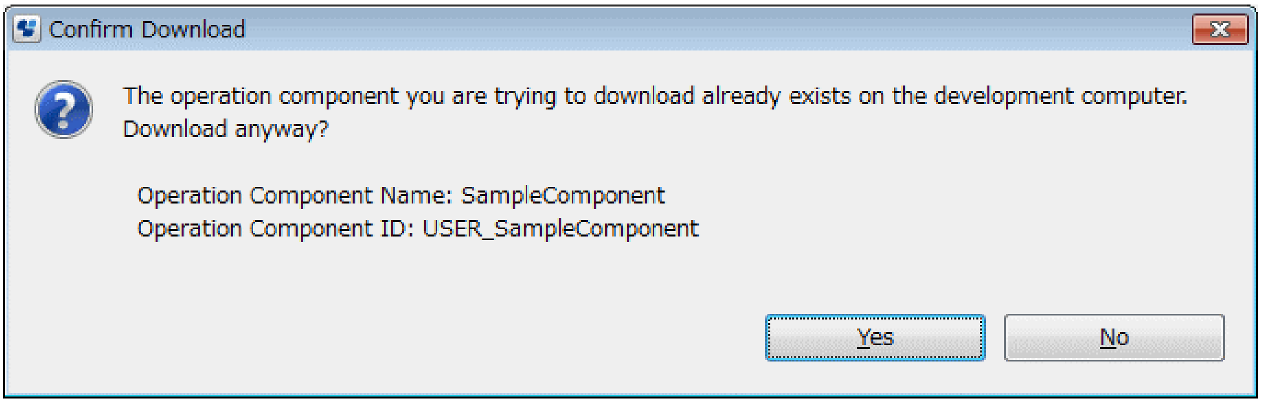
   The list of operation components will be displayed on the tree.

4. Select the operation component to be downloaded.

5. To register the downloaded operation components with the palette in the Process Definition Editor, select the **Registers the operation component in the Process Definition Editor palette.** check box.

6. Click **Finish** to download the operation component from the specified management server.

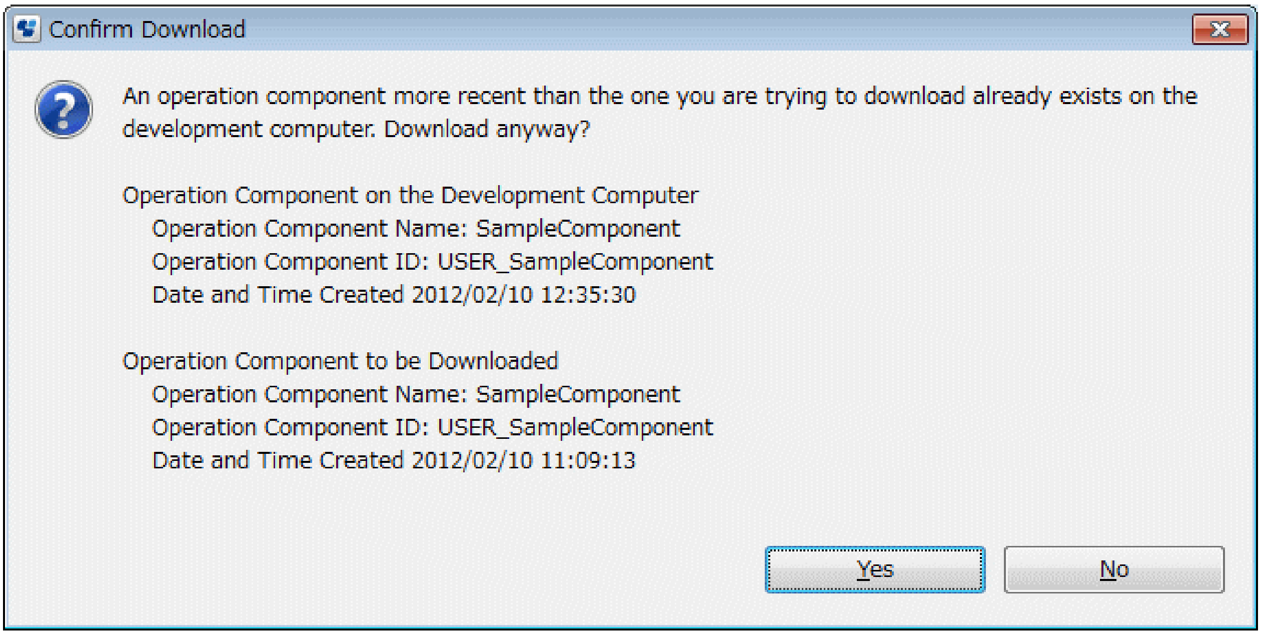
   If an operation component with the same ID exists in Studio, an update download confirmation message will be displayed. Click **Yes** to update download the operation component. If **No** is clicked, the operation component will not be downloaded.



The operation component you are trying to download already exists on the development computer. Download anyway?

Operation Component Name: SampleComponent
Operation Component ID: USER_SampleComponent

Additionally, if the date and time when the downloaded operation component was created is before that of the operation component on the development computer, the following message will be displayed.



An operation component more recent than the one you are trying to download already exists on the development computer. Download anyway?

Operation Component on the Development Computer
Operation Component Name: SampleComponent
Operation Component ID: USER_SampleComponent
Date and Time Created 2012/02/10 12:35:30

Operation Component to be Downloaded
Operation Component Name: SampleComponent
Operation Component ID: USER_SampleComponent
Date and Time Created 2012/02/10 11:09:13

5.8 Importing Script Files

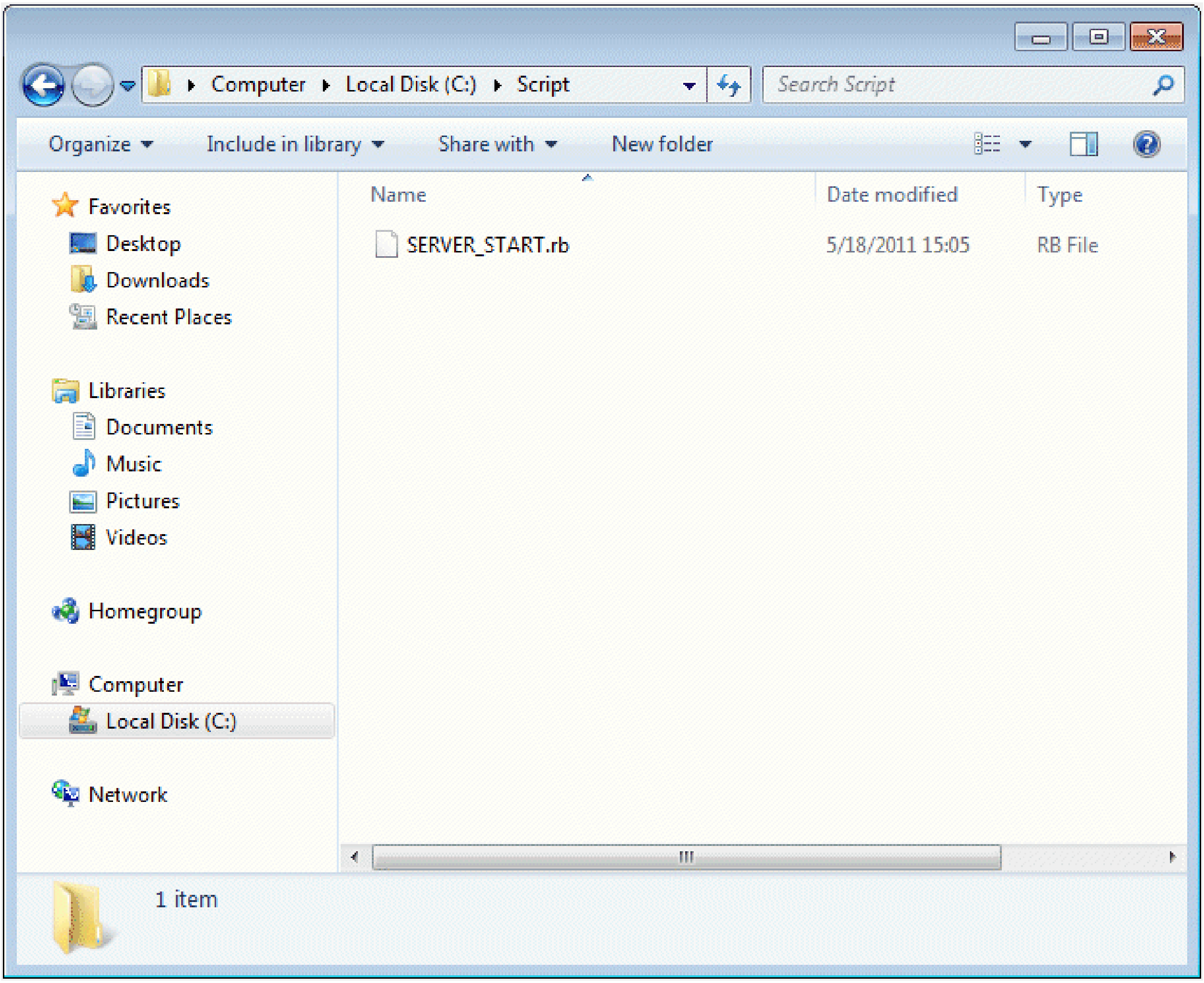
This section explains how to import Ruby or Perl script files.

Prerequisite:

- The operation component project to which the script file will be imported must have already been created.

1. In the **Operation Component Management** view, right-click on the project into which script files are to be imported. Select **Import Script File** from the pop-up menu.

   The **Import Script File** dialog box will be displayed.



2. Move to the location where the script file is stored.

3. Select the script file and then click the **Open** button.

4. If the project already contains a script file with the specified name, a dialog box will be displayed asking if you want to overwrite the file. Also if the project already contains a script file with another name, a dialog box will be displayed asking if you want to replace the file. The file is overwritten or replaced if you click **Yes**. Click **No** to stop the import.

Note

Only files with the extension ".rb" or ".pl" can be imported.

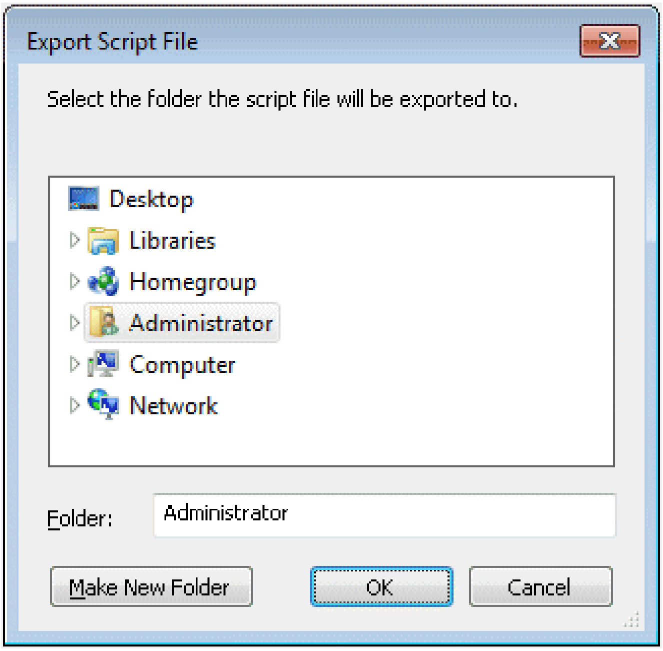
5.9 Exporting Script Files

Ruby or Perl script files can be exported from Systemwalker Runbook Automation Studio to the local file system. For example, the exported files can be used to import script files to Systemwalker Runbook Automation Studio in another environment.

This section explains how to export script files.

1. Right-click on a script file in the Operation Component Management view. Select **Export Script File** from the pop-up menu.

   The **Export Script File** dialog box will be displayed.



Select the folder the script file will be exported to.

Folder: Administrator

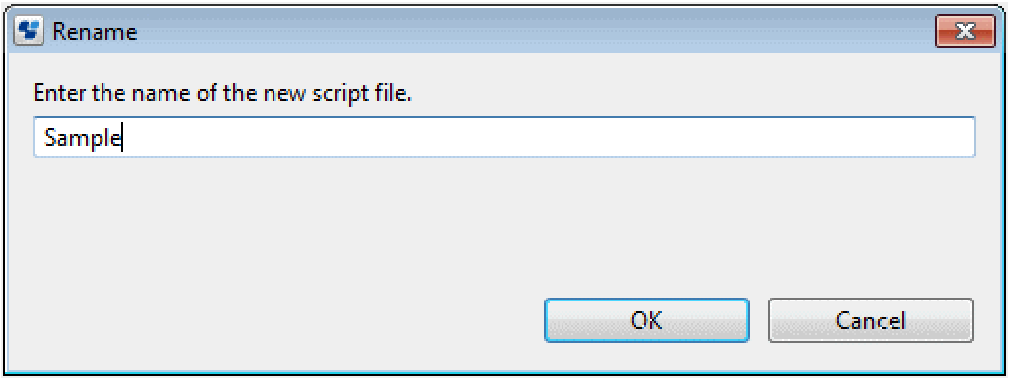
2. Move to the location where the script files to be exported will be stored.
3. Click the **OK** button.
4. If the export destination folder already contains a script file with the specified name, a dialog box will be displayed asking if you want to overwrite the file. Click **Yes** to overwrite the file. Click **No** to stop the export.

5.10 Renaming Script Files

This section explains how to rename a script file.

1. Right-click on a script file in the **Operation Component Management** view. Select **Rename** from the pop-up menu.

   The **Rename** dialog box will be displayed.



Enter the name of the new script file.

Sample

2. Enter the new name for the script file into the text field. Do not include the file extension when entering the name. The previous name appears in the text field. The **OK** button is enabled when another file name is entered.
3. Click the **OK** button to confirm the renaming.

5.11 Opening the Script File

This section explains how to open the script file.

1. Execute one of the following operations.

    - Double-click a Ruby or Perl script file in the **Operation Component Management** view.

    - Right-click on a Ruby or Perl script file in the **Operation Component Management** view. Select Open from the pop-up menu.

A script file is opened by the editor that was started most recently for that file. The text editor opens the script file if the file is being opened for the first time.

5.12 Opening the Script File from an Application

This section explains how to open the script file with a selected editor. .

1. Right-click on a Ruby or Perl script file in the **Operation Component Management** view. Select Open from Application from the pop-up menu.

2. Select the Editor that will open the Ruby or Perl script file. The Editors that can be selected are as follows.

    - Text Editor: The text editor provided in RBA-Studio will start

    - System Editor: The editor associated with the extension in the system (operating system) will start

5.12.1 Content Assistance

Content Assistance is a feature that displays a list of proposals that have been imported into operation component projects in a text editor when editing Ruby scripts. The proposals appear when you press Ctrl+Space on your keyboard or when you enter a key letter. The following shows the list of proposals. Select a proposal from the list to insert the information:

- Variables, constants, methods, classes, and modules defined in the source

- Variables, constants, methods, classes, and modules defined in the libraries standard to Ruby

- Variables, methods, classes, and modules defined in the libraries provided by RBA

- Reserved words

- Syntax templates for control statements

Images representing the information to be inserted and the content of the information are shown in the list of proposals. The following table shows the images used to represent the information displayed in the list:

Image	Information type
C	Class
M	Module
△	Variable, constant
●	Method
▤	Syntax template
Nothing displayed	Reserved words

Usage

The following steps describe how to use input assistance to enter methods and variables into Ruby scripts:

1. Open the Ruby script with a text editor.

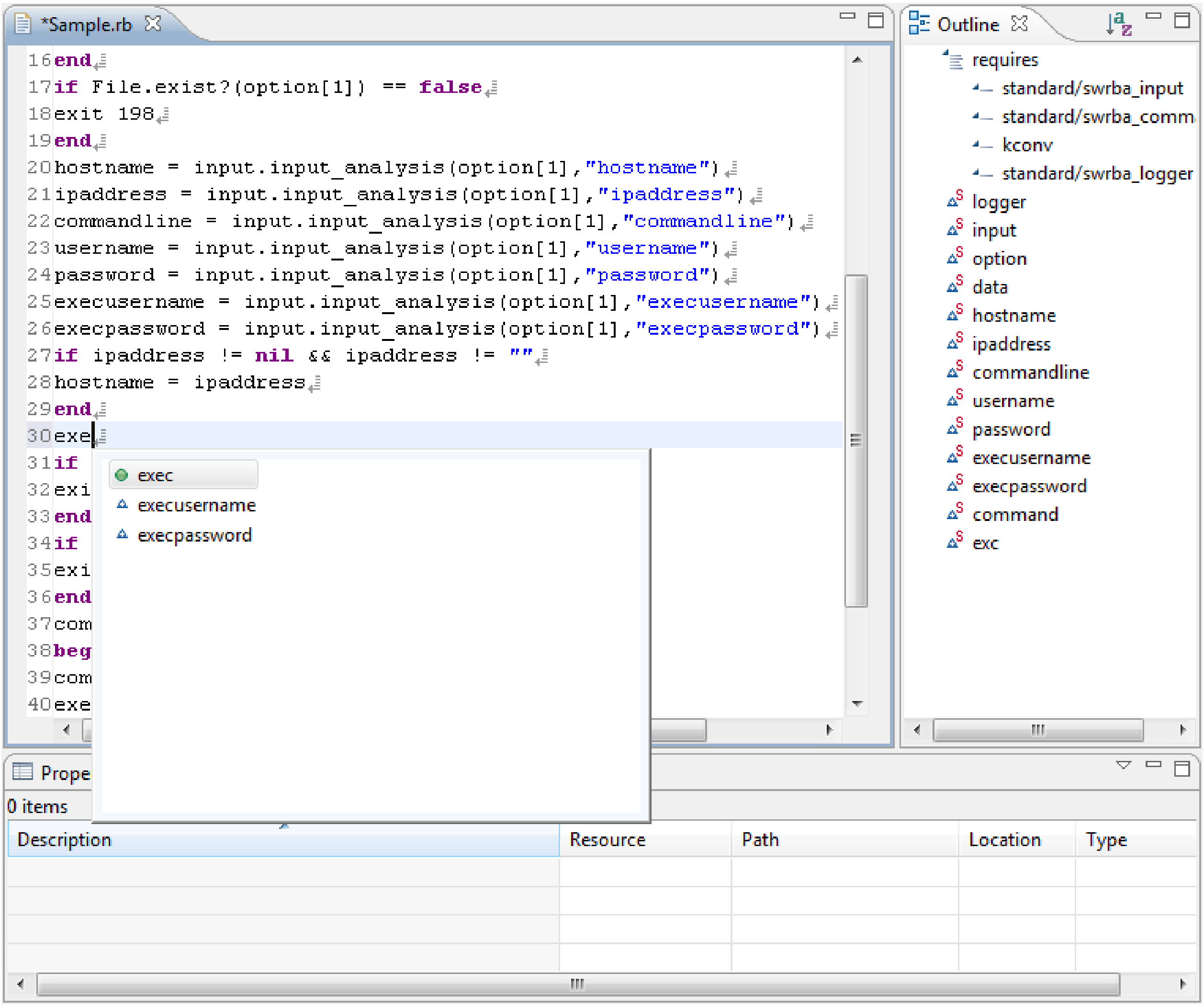
2. In the text editor, a list of proposals appears when you press Ctrl+Space on your keyboard or enter a key letter. The following explains when and what proposals are displayed:

    - Proposals for variables and methods are displayed when you press Ctrl+Space on your keyboard at the separation of words.

- Proposals for classes/packages, modules, reserved words, and syntax templates for control statements that start with the letter or letters you have entered are displayed when you press Ctrl+Space on your keyboard after entering a letter or letters.

- Proposals for variables and methods are displayed when you enter a period (.) after a variable name.

- Proposals for variables and methods are displayed when you enter the second of two successive colons (:).



```
end
if File.exist?(option[1]) == false
exit 198
end
hostname = input.input_analysis(option[1],"hostname")
ipaddress = input.input_analysis(option[1],"ipaddress")
commandline = input.input_analysis(option[1],"commandline")
username = input.input_analysis(option[1],"username")
password = input.input_analysis(option[1],"password")
execusername = input.input_analysis(option[1],"execusername")
execpassword = input.input_analysis(option[1],"execpassword")
if ipaddress != nil && ipaddress != ""
hostname = ipaddress
end
exe
if
exi
end
if
exi
end
com
beg
com
exe
```

Description	Resource	Path	Location	Type

3. Select a proposal from the list.

5.12.2 Displaying the Outline View

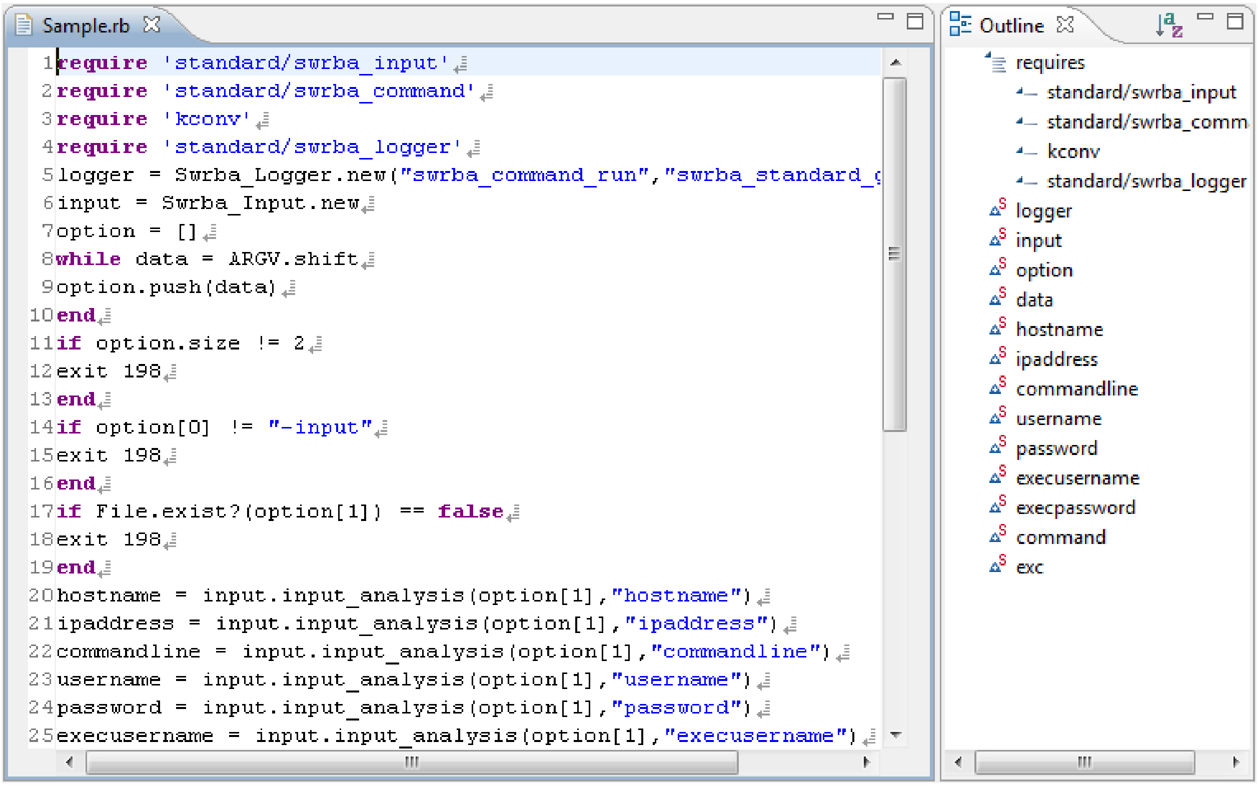
The **Outline** view is a feature that displays a tree that outlines the structure of parent-child relationships amongst Require declarations, classes, modules, variables, constants, and methods in the script. It is displayed when Ruby scripts that have been imported into operation component projects are opened in a text editor. You can also select the node names relevant to the source code (Require declarations, class names, module names, method names, variable names, and constant names) by clicking on the nodes in the tree. The node images displayed in the **Outline** view are as follows:

Image	Node
≡	Require declaration block
—	Require declaration
Ⓒ	Class

Image	Node
Ⓜ	Module
△	Variable, constant
●	Method
C	Constructor
S	Static member

Usage

The following steps describe how to use the features in the **Outline** view with Ruby scripts:



```
require 'standard/swrba_input'
require 'standard/swrba_command'
require 'kconv'
require 'standard/swrba_logger'
logger = Swrba_Logger.new("swrba_command_run","swrba_standard_
input = Swrba_Input.new
option = []
while data = ARGV.shift
option.push(data)
end
if option.size != 2
exit 198
end
if option[0] != "-input"
exit 198
end
if File.exist?(option[1]) == false
exit 198
end
hostname = input.input_analysis(option[1],"hostname")
ipaddress = input.input_analysis(option[1],"ipaddress")
commandline = input.input_analysis(option[1],"commandline")
username = input.input_analysis(option[1],"username")
password = input.input_analysis(option[1],"password")
execusername = input.input_analysis(option[1],"execusername")
```

1. Open the Ruby script with a text editor. The **Outline** view appears on the right side of the text editor by default.

2. Select the node names (Require declarations, class names, module names, method names, variable names, and constant names) in the **Outline** view and click.

3. The string corresponding to the node clicked is selected in the source code. With methods, the arguments are also displayed in the outline, but only the method name part is selected in the source code when you click the node.

Point

- The nodes of the tree displayed in the **Outline** view for the variable names, constant names, and method names can be sorted into alphabetical order for each Require declaration, class or module by clicking the **Sort** icon on the toolbar in the outline. The original order is restored by clicking the **Sort** icon one more time.

- If the source is edited in the text editor, the **Outline** view is updated automatically.

Note

Nothing is displayed in the **Outline** view when Perl scripts are opened in the text editor.

5.12.3 Emphasizing Reserved Words

The following items can be emphasized when Ruby or Perl scripts that have been imported into operation component projects are opened in a text editor. The emphasis can be by color, bolding, or italics.

[Ruby]

- Reserved words
- Strings (strings inside quotation marks)
- Comment
- Method
- Variable

[Perl]

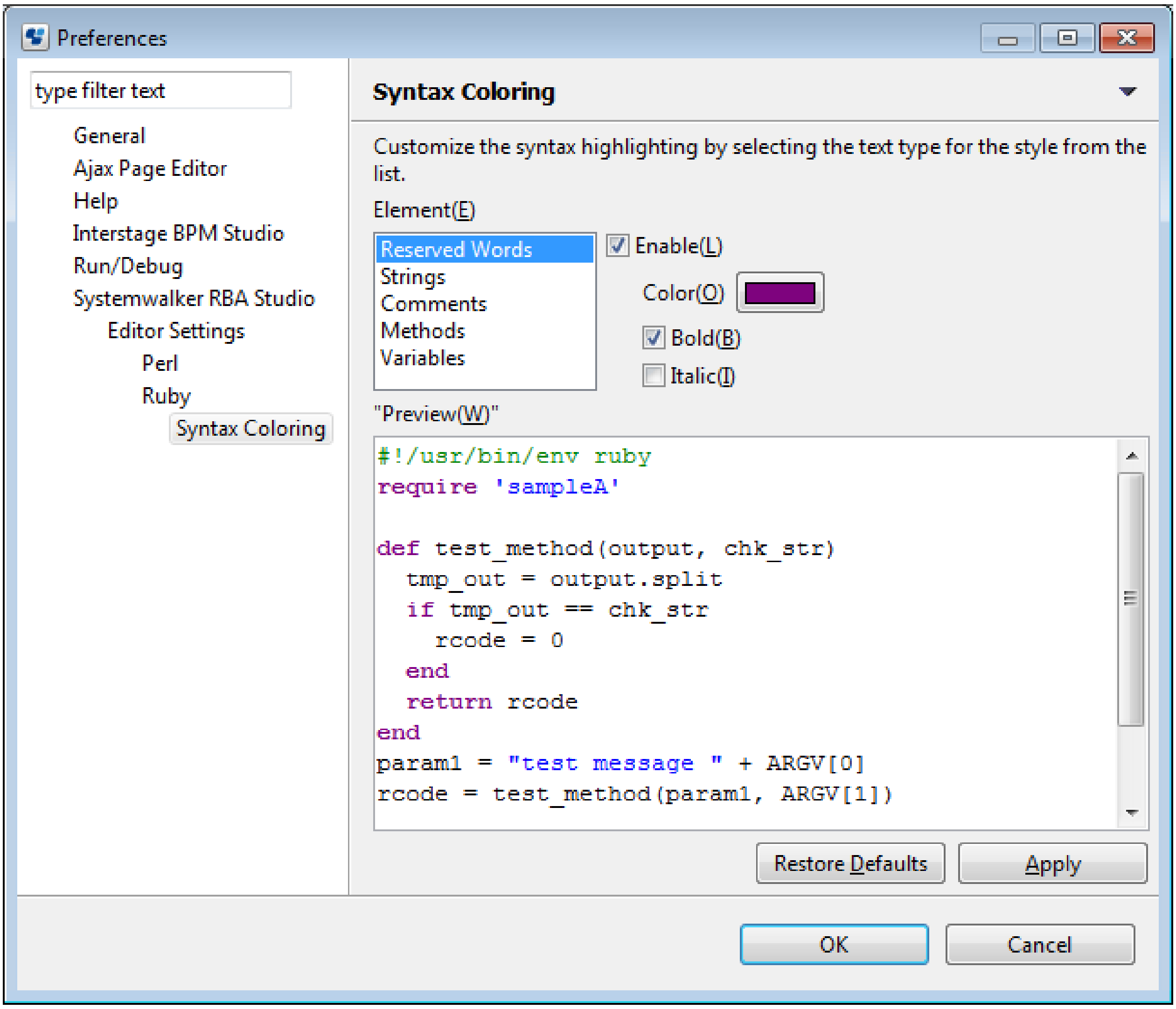
- Reserved words
- String
- Comment

Usage

Open the Ruby script with a text editor. Use the following procedure to change the detail emphasis can be by color of the script. The procedure for Perl scripts is the same, but anywhere "Ruby" is mentioned should be replaced with "Perl".

1. Open the Ruby script with a text editor. Use the following procedure to change the the detail emphasis can be by color of the script.
2. Select **Window** >> **Preferences**.

3. From the tree on the left of the **Preferences** dialog, select **Systemwalker RBA Studio** >> **Editor Settings** >> **Ruby** >> **Syntax Coloring**.



Customize the syntax highlighting by selecting the text type for the style from the list.

Element(E)

Reserved Words
Strings
Comments
Methods
Variables

☑ Enable(L)

Color(O) ■

☑ Bold(B)

□ Italic(I)

"Preview(W)"

```
#!/usr/bin/env ruby
require 'sampleA'

def test_method(output, chk_str)
  tmp_out = output.split
  if tmp_out == chk_str
    rcode = 0
  end
  return rcode
end
param1 = "test message " + ARGV[0]
rcode = test_method(param1, ARGV[1])
```

4. In the **Element** field, select the element to be emphasized.

5. Select or cancel the checkbox for selecting the type of emphasis. To change the color, click the color selection button and select the new color in the **Color** dialog box. Cancelling the checkbox for **Enable** disables all emphasis settings for the element selected in the **Element** field.

6. The emphasis can be previewed in the **Preview** field.

7. Click the **OK** button.

Point

Emphasis settings are reflected immediately in any script files open in the text editor when you click **Apply** or **OK** in the **Preferences** dialog box.

5.13 Script File Syntax Check

To use the Perl script file syntax check function, install the following software.

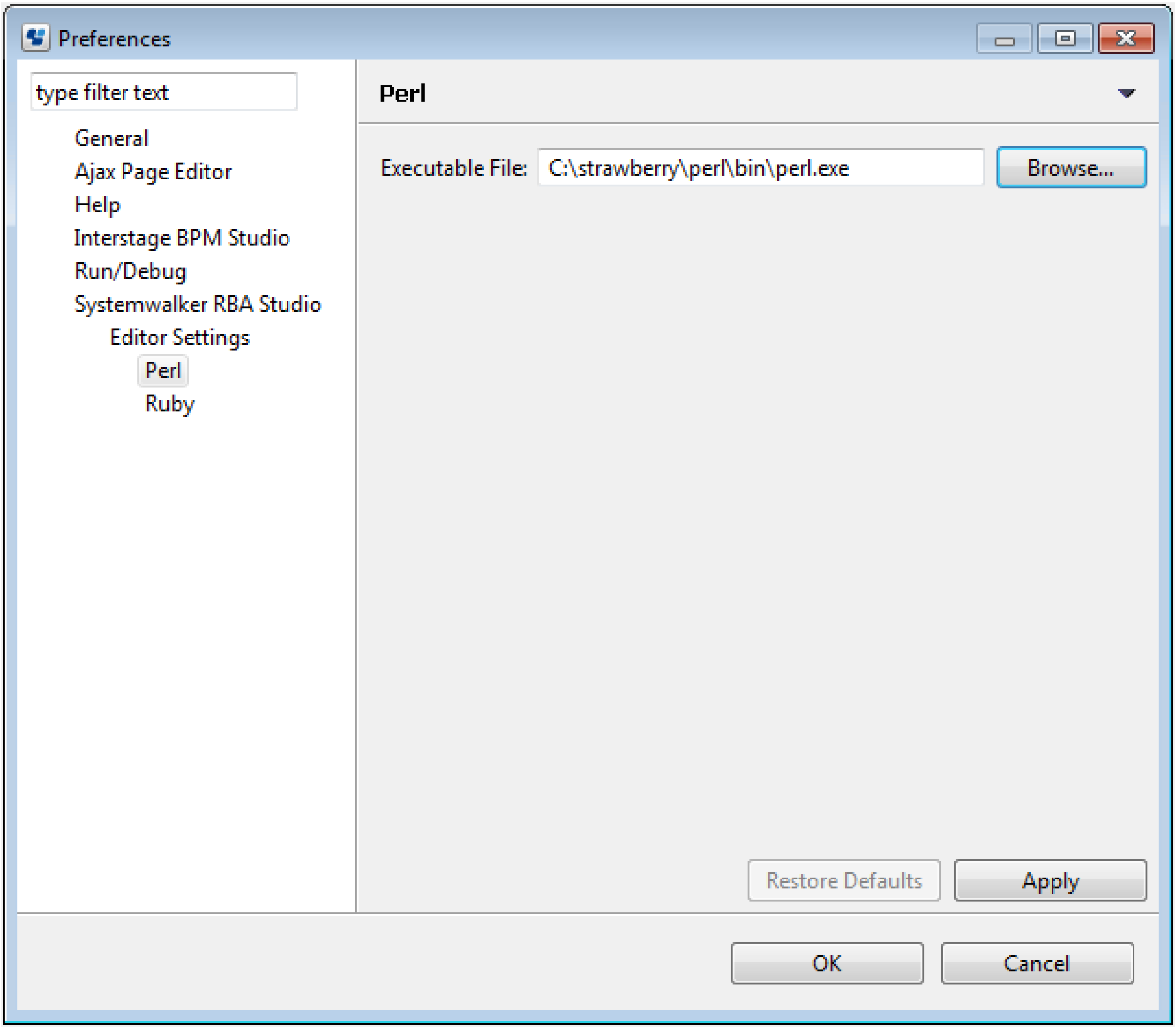
Any directory can be specified as the installation destination.

- strawberry perl 5.12.0

To use the Perl script file syntax check function, the path of the Perl execution file must be set.

1. Select Window >> Settings.

   The Setting window will be displayed.

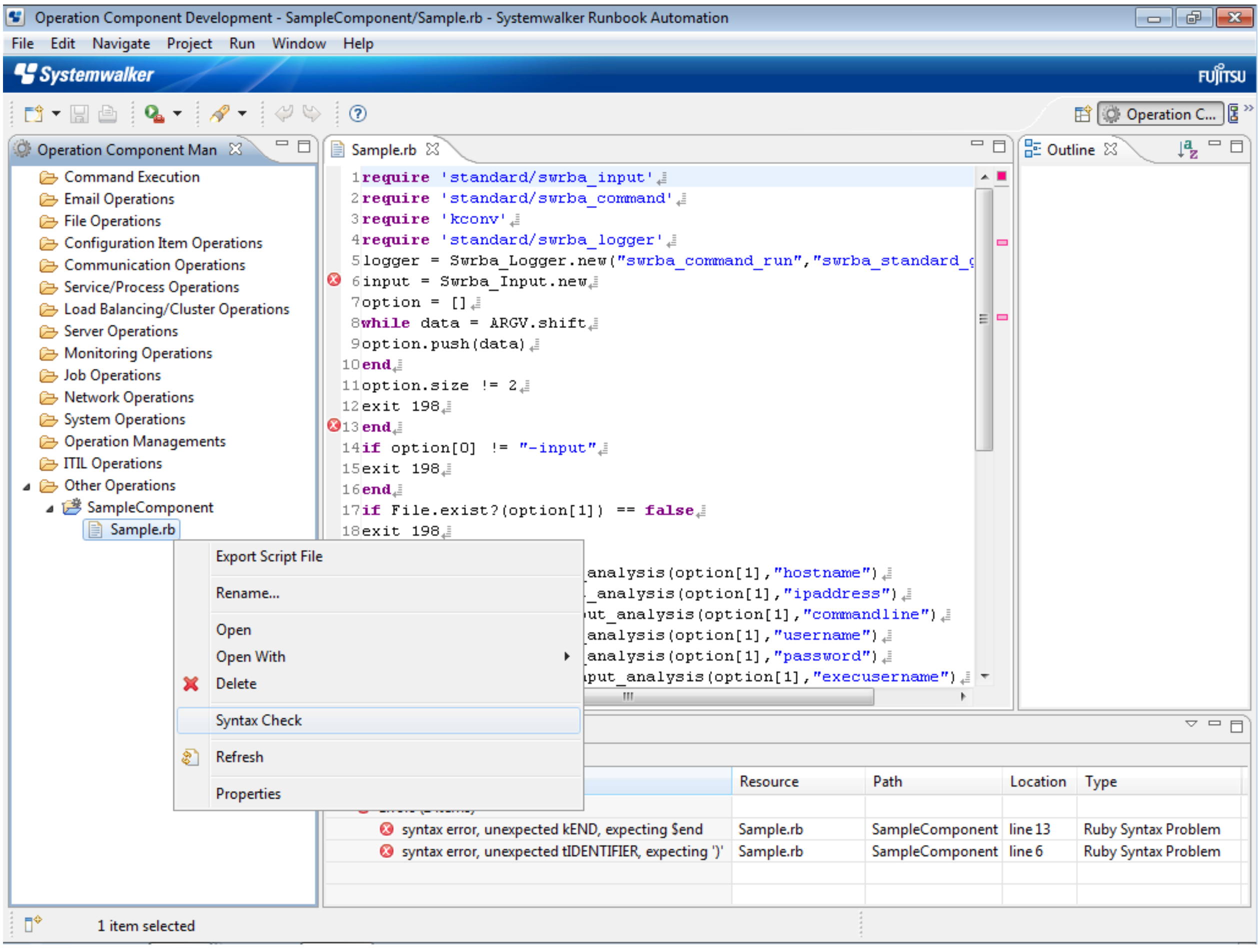


Executable File: C:\strawberry\perl\bin\perl.exe

2. From the left-hand tree in the Settings dialog box, select Systemwalker RBA Studio >> **Editor Settings** >> Perl.
3. Click the **Browse** button then select the Perl execution file.
4. Click the **OK** button.

How to check syntax

This section explains how to execute the Ruby or Perl script file syntax check.



Sample.rb

```
require 'standard/swrba_input'
require 'standard/swrba_command'
require 'kconv'
require 'standard/swrba_logger'
logger = Swrba_Logger.new("swrba_command_run","swrba_standard_c
input = Swrba_Input.new
option = []
while data = ARGV.shift
option.push(data)
end
option.size != 2
exit 198
end
if option[0] != "-input"
exit 198
end
if File.exist?(option[1]) == false
exit 198
analysis(option[1],"hostname")
_analysis(option[1],"ipaddress")
ut_analysis(option[1],"commandline")
analysis(option[1],"username")
analysis(option[1],"password")
put_analysis(option[1],"execusername")
```

	Resource	Path	Location	Type
syntax error, unexpected kEND, expecting $end	Sample.rb	SampleComponent	line 13	Ruby Syntax Problem
syntax error, unexpected tIDENTIFIER, expecting ')'	Sample.rb	SampleComponent	line 6	Ruby Syntax Problem

1. Perform one of the following operations:

   - Right-click on the Ruby or Perl script file displayed in the **Operation Component Management** view. Select **Syntax Check** from the pop-up menu.

   - Open the Ruby or Perl script in the text editor and right click. Select **Syntax Check** from the pop-up menu.

   - Open the Ruby or Perl script in the text editor and then save the file after editing.

2. If there is a problem with the syntax of the script, the results of the verification of the syntax are displayed in the **Problems** view. Markers will also be displayed against the lines with the problems when the script file is open in the text editor.

3. Select the **Problems** view and double-click on the problem displayed or select **Go to** from the context menu to move the cursor to the beginning of the line (the far left) with the problem. The relevant script file opens automatically in the text editor if it is not already open.

Note

- If the system editor was the last application used to open Ruby or Perl scripts, then when you double-click or use the context menu to jump to a problem from the **Problems** view, the system editor will become active rather than the text editor automatically starting. The cursor will not automatically jump to the problem line when this happens.

- If errors are displayed in the **Problems** view and you edit the source in the editor, the markers in the editor, the line numbers in the **Problems** view, and the destination for the jump when you double-click in the **Problems** view are all updated when you save the file.

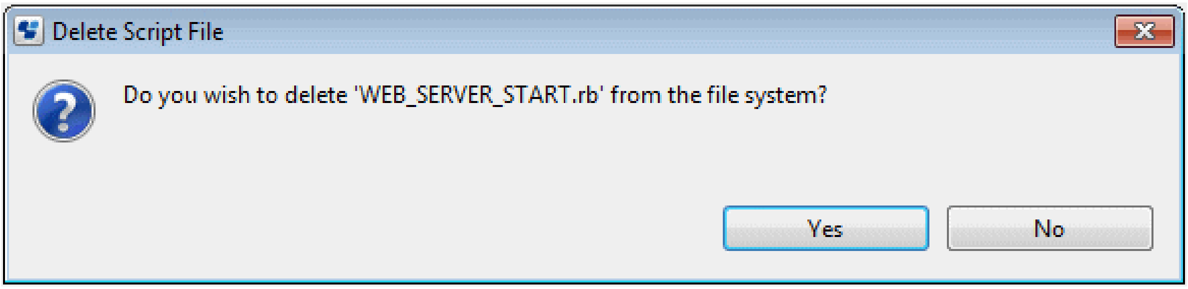
5.14 Deleting Script Files

Deleting a script file removes it from both Systemwalker Runbook Automation Studio and the file system.

This section explains how to delete script files.

1. Right-click on a script file in the **Operation Component Management** view. Select **Delete** from the pop-up menu.

   The **Delete Script File** dialog box will be displayed.



Do you wish to delete 'WEB_SERVER_START.rb' from the file system?

2. Click **Yes** to confirm the deletion. Click **No** to stop the deletion.

5.15 Registering Operation Components on the Palette

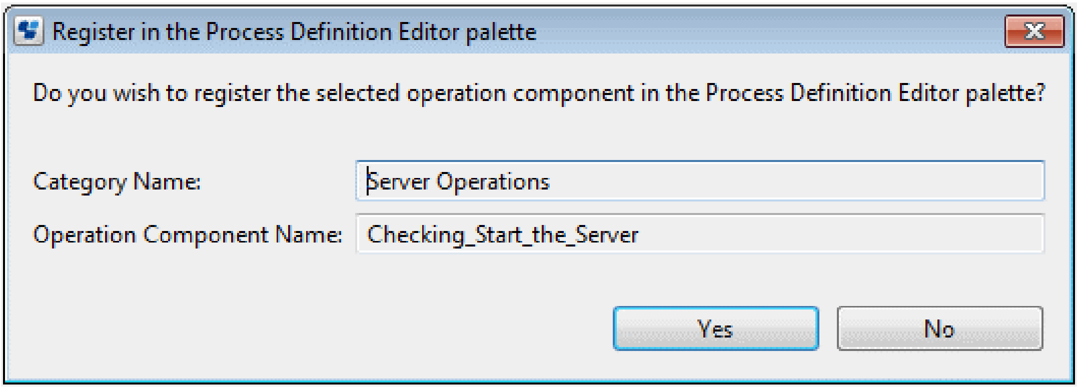
Operation component nodes can be registered with the palette in the Process Definition Editor.

If an operation component already registered on the palette is edited, re-register (overwrite) will be used.

This section explains how to register operation components with the palette in the Process Definition Editor.

1. Right-click on an operation component project in the **Operation Component Management** view. Select **Register in the Process Definition Editor palette** from the pop-up menu.

   The **Register in the Process Definition Editor palette** dialog box will be displayed.



Do you wish to register the selected operation component in the Process Definition Editor palette?

Category Name: Server Operations

Operation Component Name: Checking_Start_the_Server

2. Click the **Yes** button.

Point

- To use the edited operation component in the Process Definition Editor, it must be re-registered (re-register by overwriting) on the Process Definition Editor palette. The pre-edit operation component definition will be used until register to the palette by overwriting is executed.

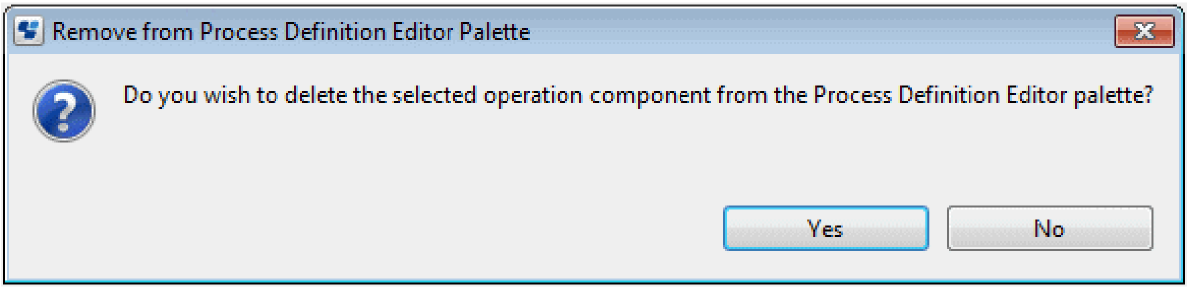
5.16 Removing Operation Components from the Palette

It is possible to remove operation component nodes from the palette in the Process Definition Editor.

This section explains how to remove operation components from the palette in the Process Definition Editor.

1. Right-click on an operation component project in the **Operation Component Management** view. Select **Remove from Process Definition Editor palette** from the pop-up menu.

   The **Remove from Process Definition Editor palette** dialog box will be displayed.



Remove from Process Definition Editor Palette

Do you wish to delete the selected operation component from the Process Definition Editor palette?

2. Click **Yes** to confirm the deletion.

Note

- You cannot delete an operation component to be used in process definition from the Process Definition Editor's palette.

Chapter 6 Modeling Processes

A process definition is complete, when it has a Start Node, at least one Exit Node and all nodes are connected through arrows.

Once you have created a process definition, you are recommended to model your process with the following sequence of steps.

1. Add all nodes that you need in your process definition.

   For details, refer to "6.1 Input Definitions for Operation Component Nodes", "6.2 Output Definitions for Operation Component Nodes", and "6.7 Adding and Editing Nodes" covered later in this chapter.

2. Add all nodes that you need in your process definition.

   For details, refer to section 6.8 Adding and Editing Arrows.

3. Optional: Add swimlanes to visually group activities performed by the same Role.

   For details, refer to section 6.9 Adding and Editing Swimlanes.

4. Optional: Add groups to visually group activities of the same category.

   For details, refer to section 6.10 Adding and Editing Groups.

5. Optional: If you want to make comments on your process definition, add annotations.

   For details, refer to section 6.11 Adding Annotations.

6. Optional: Define the owner of the process instances created from the process definition.

   For details, refer to section 6.14 Assigning Process Instance Owners.

7. Assign all activities to Roles.

   For details, refer to section 6.16 Assigning Activities to Roles.

8. Define the information that process participants need to access, modify, or add.

   For details, refer to section 6.18 Specifying User Defined Attributes.

9. Create forms and associate them with activities.

   For details, refer to Chapter 8 Using Forms.

10. If you have added Voting Activity Nodes, define the voting rules.

    For details, refer to section 6.23 Defining Voting Rules.

11. If activities are due to be completed at a particular time, define a due date or timer for these activities.

    For details, refer to section 6.22 Using Due Dates and Timers.

12. If the process is to start at a particular date, define a timer for the process definition.

    For details, refer to section 6.22.2 Defining Timers.

13. If you have added Delay Nodes, define a timer for them.

    For details, refer to section 6.22.2 Defining Timers.

14. If you have added Conditional Nodes or Complex Conditional Nodes, define the required conditions.

    For details, refer to sections 6.24 Defining Conditions and 6.25 Defining Complex Conditions.

Point

You can also use pre-defined process fragments to add palette elements to your Process Definition. Refer to Chapter 7 Using Process Fragments for more information.

6.1 Input Definitions for Operation Component Nodes

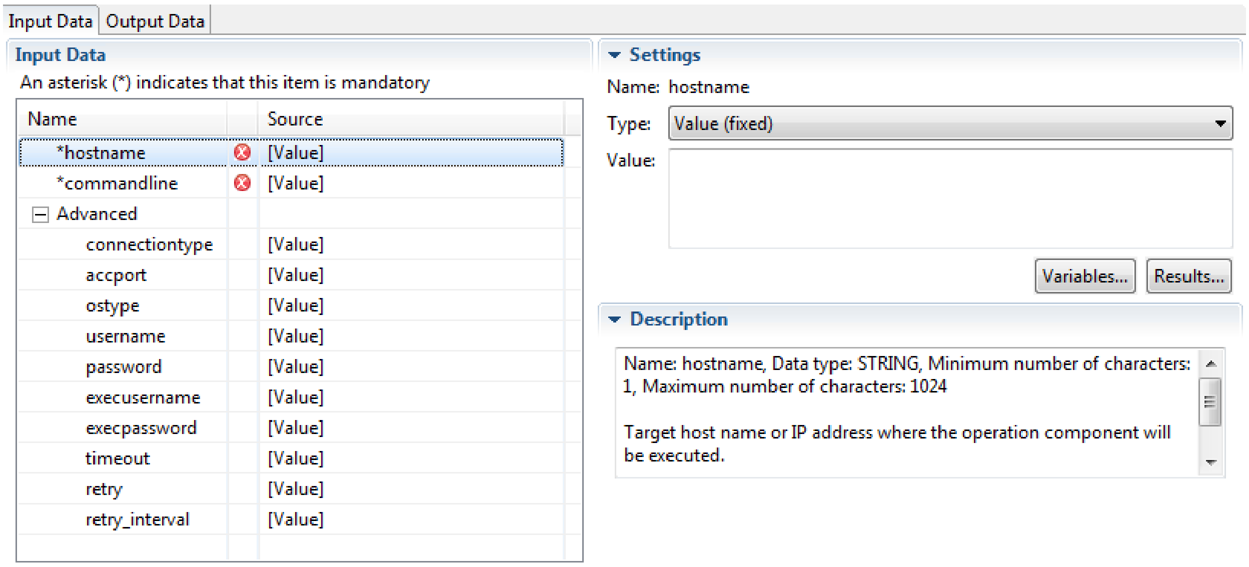
It is possible to define input and output information for operation component nodes used when operation components are called. The following explains input information definitions.

Refer to "6.7 Adding and Editing Nodes" for information on how to perform common operations for all nodes including operation component nodes.

6.1.1 Defining Input Information

This section explains input information used when operation components are called.

Input Data tab



Input Data

An asterisk (*) indicates that this item is mandatory

Name	Source
*hostname	[Value]
*commandline	[Value]
Advanced	
connectiontype	[Value]
accport	[Value]
ostype	[Value]
username	[Value]
password	[Value]
execusername	[Value]
execpassword	[Value]
timeout	[Value]
retry	[Value]
retry_interval	[Value]

Settings

Name: hostname

Type: Value (fixed)

Value:

Description

Name: hostname, Data type: STRING, Minimum number of characters: 1, Maximum number of characters: 1024

Target host name or IP address where the operation component will be executed.

In the **Input Data** tab, define the input information required to execute operation components. The tab has two sections: **Basic**, for setting input information required when the operation component is executed, and **Advanced** , for setting optional input information. In the **Basic** section, values must be set for the input information marked with an asterisk (*) to the left of the name. Set input information in the **Advanced** section only when necessary. When the line to be changed is selected, a **Settings** section will be displayed on the right-hand side of the window so that the input information can be edited.

Note

Regarding the password information that can be set in the **Basic** or **Advanced** section of the **Input Data** tab, it is recommended that you omit setting input information in the operation component node. If the input of a password is omitted, it is safe to use the information set in the Management Server.

The value specified for the input information of the operation component node can easily be accessed from the outside, and so from a security perspective it is not recommended to make password settings in this way. On this product, the setting of the password has been made simple for use in test environments and other situations when security is not an issue.

Items in the Basic and Advanced sections of the Input Data tab

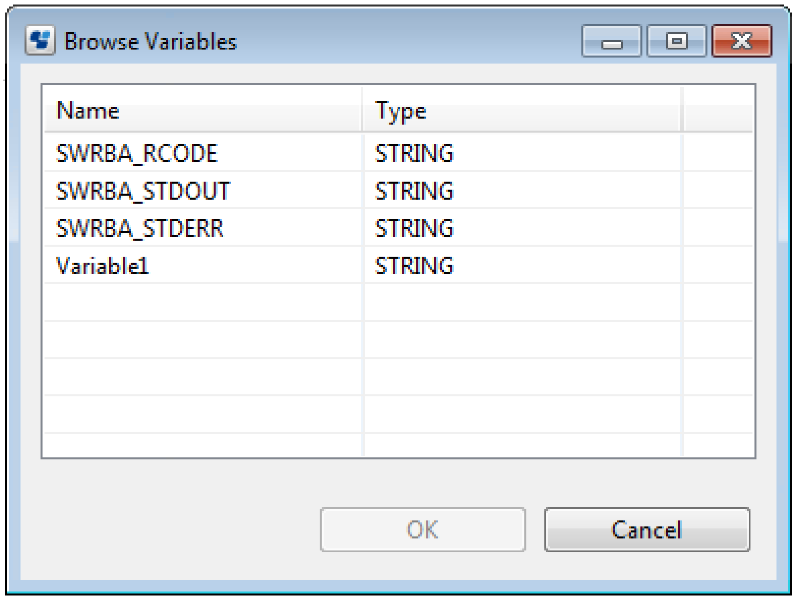
Item name	Description
Basic **(This item name is not displayed)**	Displays a list of input information for operation component to be executed.
Advanced	Displays a list of optional input information.

Item name	Description
Name	Names of input information items. An asterisk (*) is displayed to the left of the name if a value needs to be entered.
Source	Type and value of the input source of information to be passed to the operation component. Display example: "[fixed] server1" Display example: "[uda] hostname"

Items in the Settings section of the Input Data tab

Item name	Description
Name	
Type	Select the input information type: - Value (fixed) - Variable (uda) - Execution result (result)
Value	Enter the value or select it from a list. Multiple variable parameters can be passed when an operation component is executed by putting "@{uda:UDA name}" in the value field. Press Ctrl+Space in the value field to activate variable parameter input assistance. * The label changes to **Value** if **fixed** is selected for type.
Variable	Select the name of the user defined attribute from the list. * The label changes to **Variable** if **Variable (uda)** is selected for type.
Node Name	Select the node name that references the execution results from the combo box. * The label changes to **Node Name** if **Execution Results** is selected for type.
Execution Results	Select the execution results from the list. * A **Execution results** label is not displayed if **Execution Results** is selected for type. It is displayed in the list header.
Browse variables	The **Browse variables** dialog box will open.
Browse Execution Results	The **Browse Execution Results** dialog box will open.
Add	Add an item to the end of the list. (available only when input in the table format is supported)
Batch Edit	Displays the **Batch Edit** dialog box. (available only when input in the table format is supported)
Delete	Delete an item from the list. (available only when input in the table format is supported)
Up	Move a list item up one place. (available only when input in the table format is supported)
Down	Move a list item down one place. (available only when input in the table format is supported)

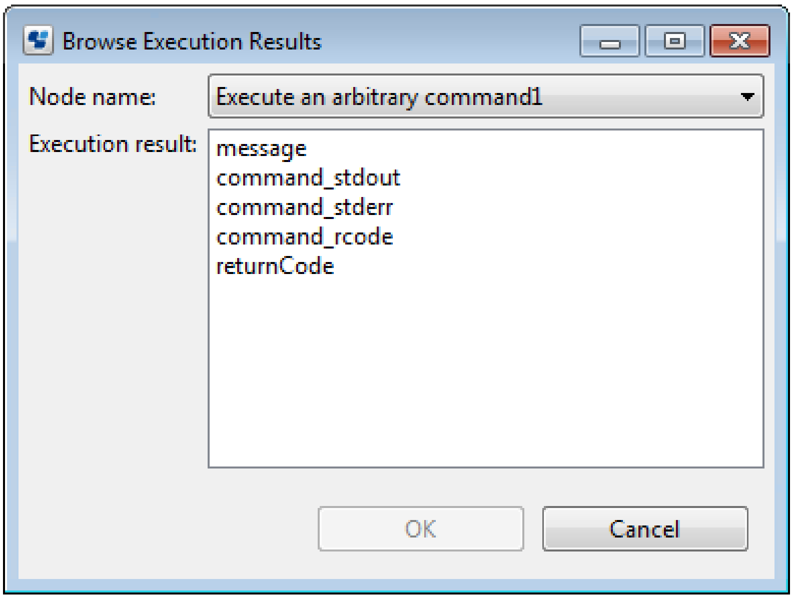
Figure 6.1 Browse Variables dialog box


Browse Variables

Name	Type
SWRBA_RCODE	STRING
SWRBA_STDOUT	STRING
SWRBA_STDERR	STRING
Variable1	STRING

Item name	Description
Name	Displays the names of variables. Click the column to sort by variable name. (the sorting order is shown in the column)(▲: ascending, ▼: descending)
Type	Displays the types of variables. Click the column to sort by type. (the sorting order is shown in the column)(▲: ascending, ▼: descending)
OK	Insert the string "@{uda: variable name}" into the **Value** field of the **Settings** section. It is inserted at the current cursor position.
Cancel	Cancel insertion of the variable and close the dialog box.

Figure 6.2 Browse Execution Results dialog box


Browse Execution Results

Node name: Execute an arbitrary command1

Execution result:

message
command_stdout
command_stderr
command_rcode
returnCode

Item name	Description
Node Name	Displays a list of the operation component node names allocated to the process definition.
Execution results	Displays a list of the execution results of the operation component node.
OK	Insert the string "@{:node name:execution results}" into the **Value** field of the **Settings** section. It is inserted at the current cursor position.
Cancel	Cancel insertion of the execution result and close the dialog box.

Items in the Description section of the Input Data tab

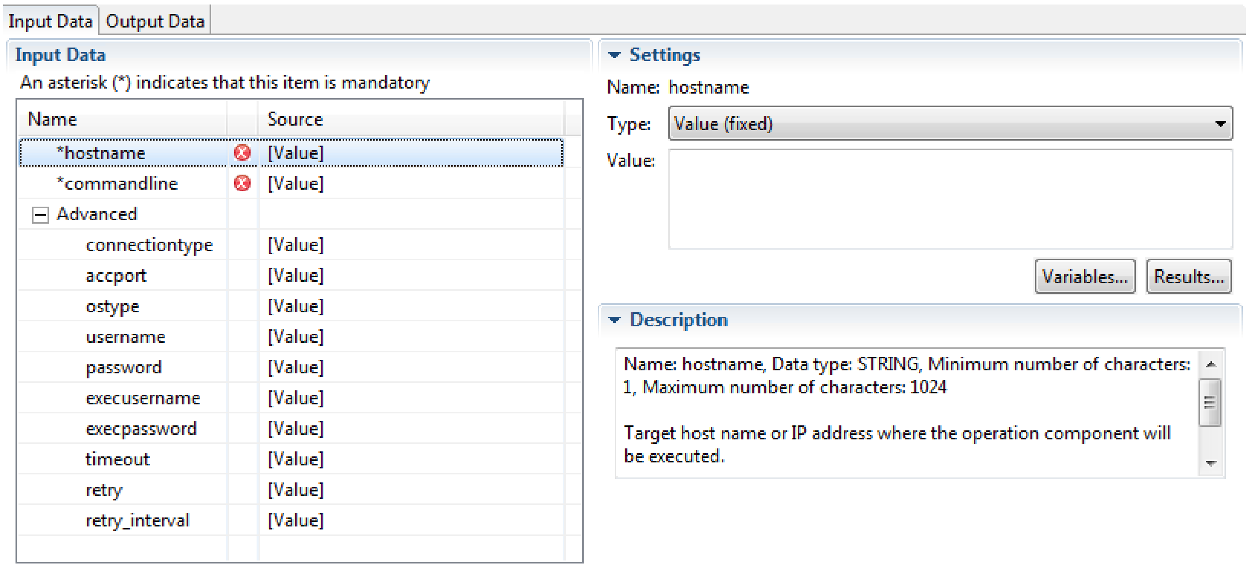
Item name	Description
-	Description for input information (cannot be edited).

Procedure for configuring input information

The following describes some of the more common procedures used to configure input information.

6.1.2 Specifying Fixed Values and Variables

Fixed values and variables can be specified for the input information for operation components.



Name	Source
*hostname	[Value]
*commandline	[Value]
Advanced	
connectiontype	[Value]
accport	[Value]
ostype	[Value]
username	[Value]
password	[Value]
executeusername	[Value]
execpassword	[Value]
timeout	[Value]
retry	[Value]
retry_interval	[Value]

Name: hostname

Type: Value (fixed)

Value:

Name: hostname, Data type: STRING, Minimum number of characters: 1, Maximum number of characters: 1024

Target host name or IP address where the operation component will be executed.

To input fixed values or variables, select "**Value (fixed)**" from the **Type** combo box.

Multiple variable parameters (variables) can be passed when an operation component is executed by putting "@{uda:UDA name}" in the **Value** field. Also, multiple variable parameters (the execution results of previous operation components) can be passed when an operation component is executed by putting "@{:node name:execution results}".

If the data type is STRING, press Ctrl+Space within the **Value** field description to display a list of the variables (UDA) that can be selected.

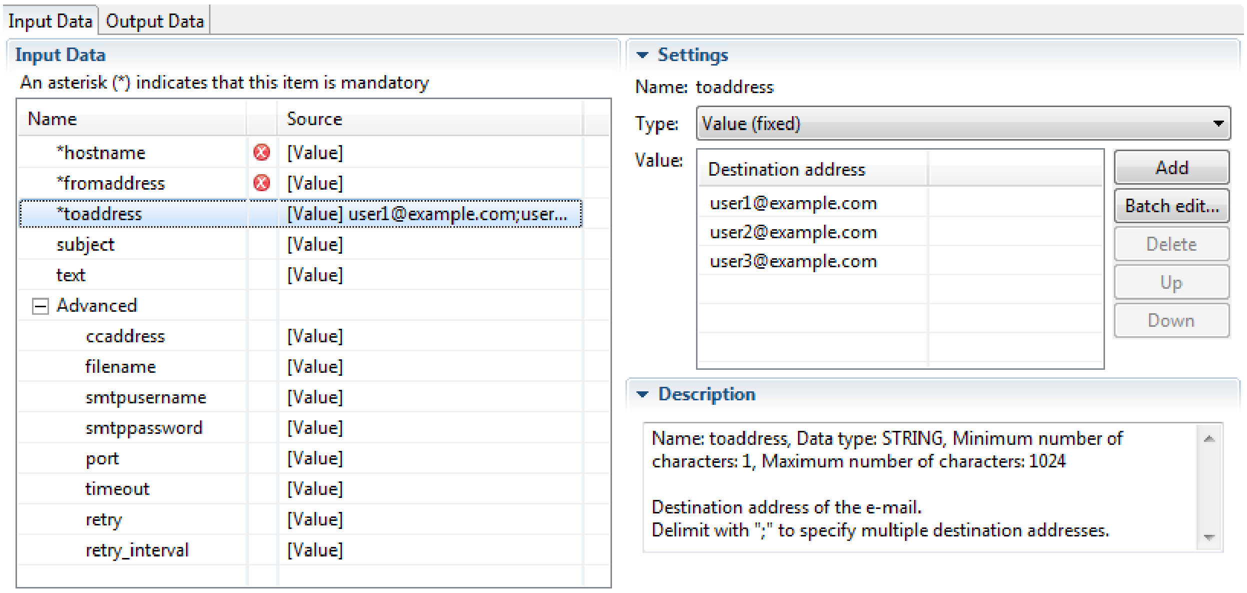
Click the **Browse variables** button to open the **Browse variables** dialog box. Click the **Browse execution results** button to open the **Browse execution results** dialog box. * The **Browse variables** and **Browse execution results** buttons are only displayed if the data type is STRING.

Refer to "Data conversion rules" in "6.1.4 Specify the Values for Variables (UDA)" for information on the rules for converting data when values are passed as the input information for operation components from variables (UDA).

If there is an error in the input value, the error icon will be displayed between the input information name and the input source. An error message is also displayed in the left of the status bar (at the bottom of Studio).

6.1.3 Specify Values in Table Format

You can specify input values in table format by using delimiters. This can be used for input information such as the toaddress input information for the "**Send emails**" component or the param input information for the "**Perform REST-based communications**" component.



Input Data

An asterisk (*) indicates that this item is mandatory

Name	Source
*hostname	⊗ [Value]
*fromaddress	⊗ [Value]
*toaddress	[Value] user1@example.com;user...
subject	[Value]
text	[Value]
⊟ Advanced	
ccaddress	[Value]
filename	[Value]
smtpusername	[Value]
smtppassword	[Value]
port	[Value]
timeout	[Value]
retry	[Value]
retry_interval	[Value]

Name: toaddress

Type: Value (fixed)

Value:

Destination address
user1@example.com
user2@example.com
user3@example.com

Name: toaddress, Data type: STRING, Minimum number of characters: 1, Maximum number of characters: 1024

Destination address of the e-mail.
Delimit with ";" to specify multiple destination addresses.

The input field for tables is displayed when "**Value (fixed)**" is selected from the **Type** combo box.

If the input information does not support the table format, the regular text field (a single row or multiple rows) is displayed.

Click the **Add** button to add a new item to the list. A new item with the default value ("user@example.com" in this example) is added to the end of the list. Click the **Delete** button to delete an item from the list. Click the **Up** button to move an item one position higher in the list. Click the **Down** button to move an item one position lower in the list.

Click the **Batch Edit** button to input data in a batch. Mail addresses can be entered in the **Batch Edit** dialog box, separated by semi-colons as is normally the case. Values can also be loaded in bulk from a CSV file.

To edit a value, click the cell for the list item. Alternatively, select the list item and press the F2 key. (The cell on the left of the selected item becomes editable).

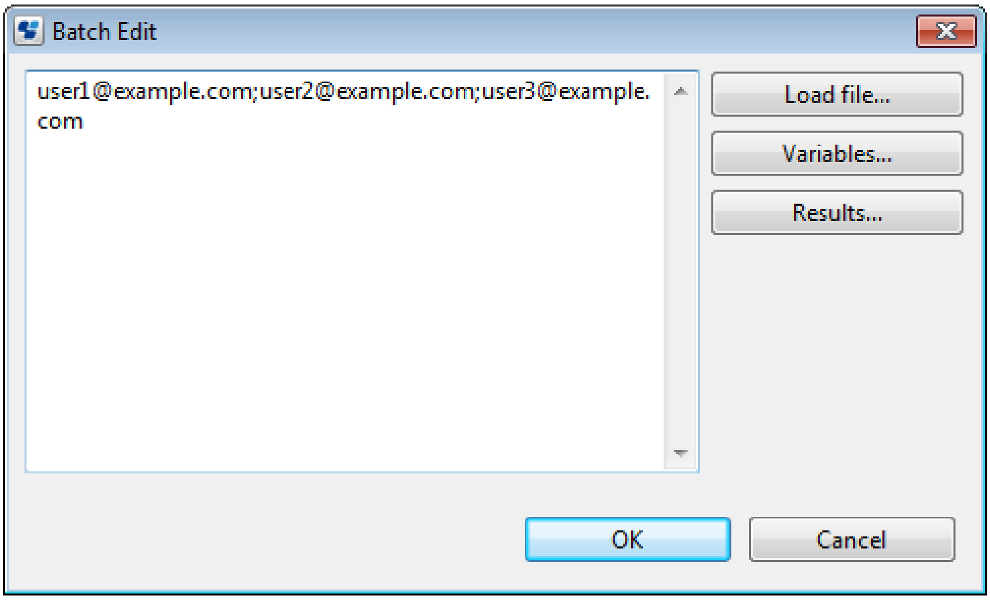
Press the Enter key to confirm the entry. Press the ESC key to cancel the entry.

Press the Tab key to make the cell to right of the list item editable. If you press the Tab key while in the cell on the far right, the cell on the far left then becomes editable.

Press the ↑ (up arrow) key to make the cell above editable. Press the ↓ (down arrow) key to make the cell below editable.

If the item exists on a list that includes null values, a warning icon will be displayed at the head of the item. An error message is also displayed in the left of the status bar (at the bottom of Studio).

Figure 6.3 Batch Edit dialog box



user1@example.com;user2@example.com;user3@example.com

Click the **Load file** button to display the **Select File** dialog box.

Values can be loaded in bulk from a CSV file selected in the **Select File** dialog box.

The content set in the **Value** field is overwritten when the file is loaded.

Point

The CSV files that can be loaded are shown below:

- Separate each item with a linefeed character.
- If working with parameters that allow two or more data to be input into one item, separate them with commas (,).
- If the data contains commas (,) and double-quotes ("), then enclose all the data that is specified in the column in double-quotes ("). Additionally, if the data contains double-quotes ("), put two double-quotes in succession. Doing so represents the appearance of the double-quote in the data.

Click the **Variables** button to open the **Variables** dialog box. Click the **Results** button to open the **Results** dialog box.

Refer to 6.1 Input Definitions for Operation Component Nodesfor information on how to use the **Variables** and **Results** dialog boxes.

Data input using the table format can be used with the following operation component parameters:

Operation component name	Parameter name
Perform REST-based communications	param
Send emails	toaddress, ccaddress, and filename
Get value using SNMP	oid
Set value using SNMP	oid, value, and type
Send SNMP trap	oid, value, and type
Execute Web service	parametername, parametervalue, and returnpropname
Check that service has started	service
Start the server	hostname, ipmiipaddress, ipmiusername, and ipmipassword
Stop the server	hostname, ipmiipaddress, ipmiusername, and ipmipassword
Stop OS	hostname, ostype, username, password, and execusername

Operation component name	Parameter name
Restart OS	hostname, ostype, username, password, and execusername.
Start virtual server	servername
Stop virtual server	servername
Restart virtual server	servername
Build virtual server	network and serveripaddress
Check server running normally	service and port
Disable server monitoring	deterrenceipaddress and deterrencehostname
Get performance counter	perfcounterpath
Check operational status of node	hostname
Check port connection	port
Set the operating system network	dns and wins
Install operating system updates	mod_patchfilename and wsus_patchno

About formatting a file to enable the Batch Edit dialog box to be read

Example 1: If the data input into each item is limited to 1

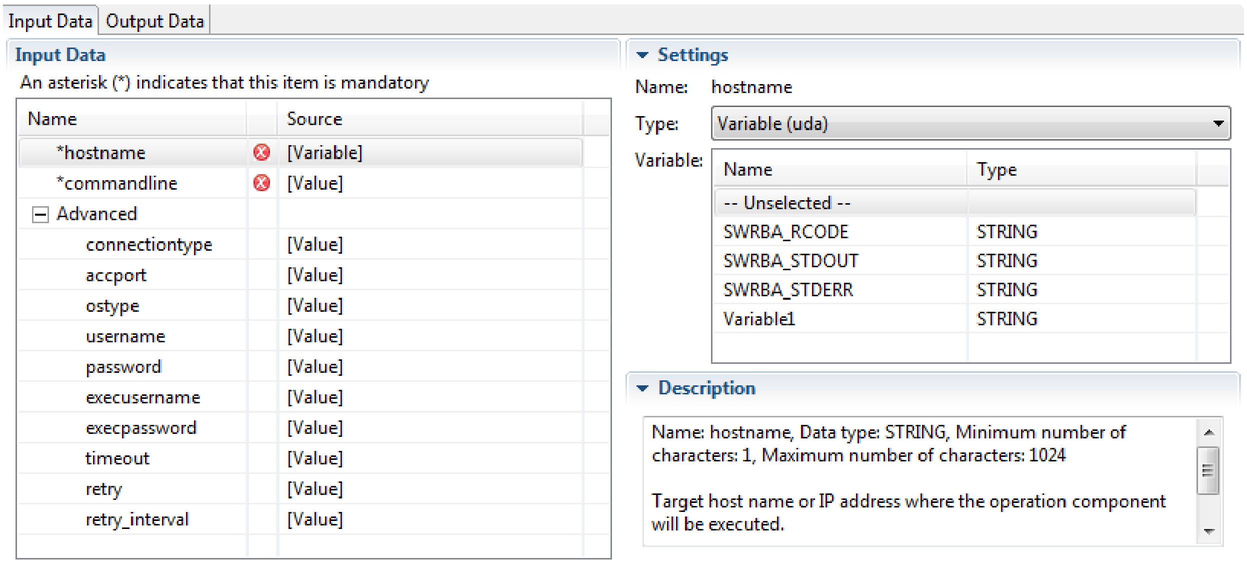
Operation Component Name:	Check that service has started
Parameter name:	service
Input file:	service 1 [linefeed] service 2 [linefeed] service 3 [linefeed] service 4 [linefeed]

Example 2: If the data input into each item is limited to 2

Operation Component Name:	Perform REST-based communications
Parameter name:	param
Input file:	param 1, value 1 [linefeed] param 2, value 2 [linefeed] param 3, value 3 [linefeed] param 4, value 4 [linefeed]

6.1.4 Specify the Values for Variables (UDA)

Variable (UDA) values can be specified for the input information of operation components.



Input Data

An asterisk (*) indicates that this item is mandatory

Name	Source
*hostname	[Variable]
*commandline	[Value]
Advanced	
connectiontype	[Value]
accport	[Value]
ostype	[Value]
username	[Value]
password	[Value]
execusername	[Value]
execpassword	[Value]
timeout	[Value]
retry	[Value]
retry_interval	[Value]

Name: hostname

Type: Variable (uda)

Variable:

Name	Type
-- Unselected --	
SWRBA_RCODE	STRING
SWRBA_STDOUT	STRING
SWRBA_STDERR	STRING
Variable1	STRING

Name: hostname, Data type: STRING, Minimum number of characters: 1, Maximum number of characters: 1024

Target host name or IP address where the operation component will be executed.

1. Select **Variable (uda)** from the **Type** combo box.

   A list of variables (UDA) that can be selected is displayed in the **Variable** list.

   Click the **Name** column to sort the list by the name of the variable (UDA). Click the **Type** column to sort by type. The default is ascending.

   The type of variable (UDA) displayed in the **Variable** list depends on the type of input information (STRING/INTEGER/PASSWORD).

Types of Input Information	Types of variables (UDA) displayed in the **Variable** list
STRING	STRING, INTEGER, BOOLEAN, DATE, XML
INTEGER	STRING, INTEGER
PASSWORD	STRING

Point

Data conversion rules

Variable values are converted as follows and passed to the input information:

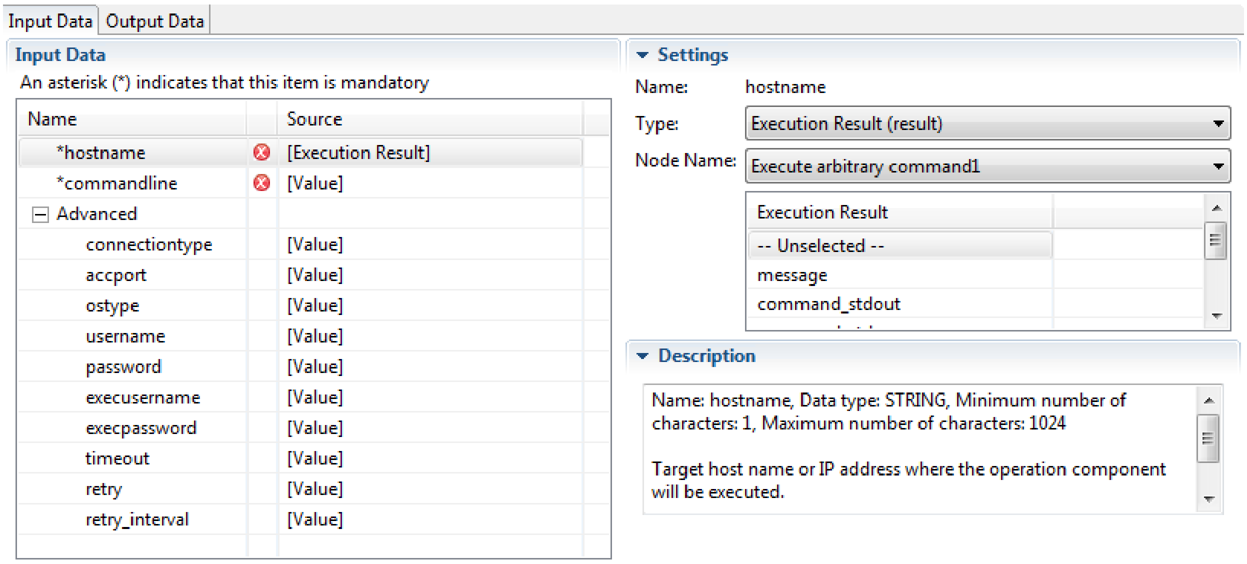
Types of Input Information	Types of Variable	Variable value (example)	Content passed to input information (examples)	Remarks
STRING	STRING	"aaaa"	"aaaa"	No conversion
	INTEGER	100	"100"	Integer strings
	BOOLEAN	true	"true"	Boolean strings
	DATE	2011/08/31 10:49:12	"1314755352734"	Date/Time (in milliseconds) strings
	XML	<root>...</root>	"<root>...</root>"	XML data strings
INTEGER	STRING	"100"	100	Converts the strings to integers. An error occurs if characters other than numbers are included.

Types of Input Information	Types of Variable	Variable value (example)	Content passed to input information (examples)	Remarks
	INTEGER	100	100	No conversion
PASSWORD	STRING	"password"	"password"	No conversion

Select the variable from the **Variable** list.

6.1.5 Specify the Execution Results from the Previous Operation Component

Execution results from the previous operation component can be specified for the input information of operation components.



Input Data

An asterisk (*) indicates that this item is mandatory

Name	Source
*hostname	[Execution Result]
*commandline	[Value]
Advanced	
connectiontype	[Value]
accport	[Value]
ostype	[Value]
username	[Value]
password	[Value]
execusername	[Value]
execpassword	[Value]
timeout	[Value]
retry	[Value]
retry_interval	[Value]

Name: hostname

Type: Execution Result (result)

Node Name: Execute arbitrary command1

Execution Result
-- Unselected --
message
command_stdout

Name: hostname, Data type: STRING, Minimum number of characters: 1, Maximum number of characters: 1024

Target host name or IP address where the operation component will be executed.

1. Select **Execution results (result)** from the **Type** combo box.
2. Select the operation component node from the **Node Name** combo box.
3. Select the execution result to be passed as input information from the **Execution results** list.

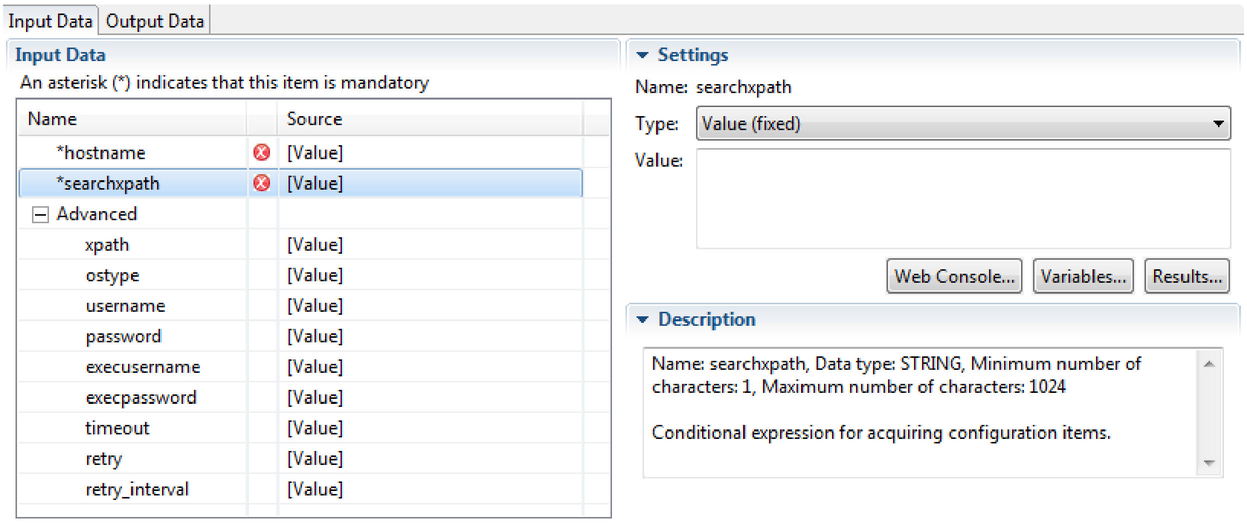
Point

To make it easier to find the required operation component, make sure the names used for the operation components in the process definitions are unique.

A message indicating that the names are the same appears in the **Problems** view when you save the process definition or click the **Validate.**

6.1.6 Generate the XPath Expression required to Obtain Configuration Information

XPath generated based on the entered conditions can be used as input information values:



Input Data

An asterisk (*) indicates that this item is mandatory

Name		Source
*hostname	⊗	[Value]
*searchxpath	⊗	[Value]
⊟ Advanced		
xpath		[Value]
ostype		[Value]
username		[Value]
password		[Value]
execusername		[Value]
execpassword		[Value]
timeout		[Value]
retry		[Value]
retry_interval		[Value]

Settings

Name: searchxpath

Type: Value (fixed)

Value:

Description

Name: searchxpath, Data type: STRING, Minimum number of characters: 1, Maximum number of characters: 1024

Conditional expression for acquiring configuration items.

1. Click the **Web Console** button.

   The **Select Server Connection** dialog box opens.

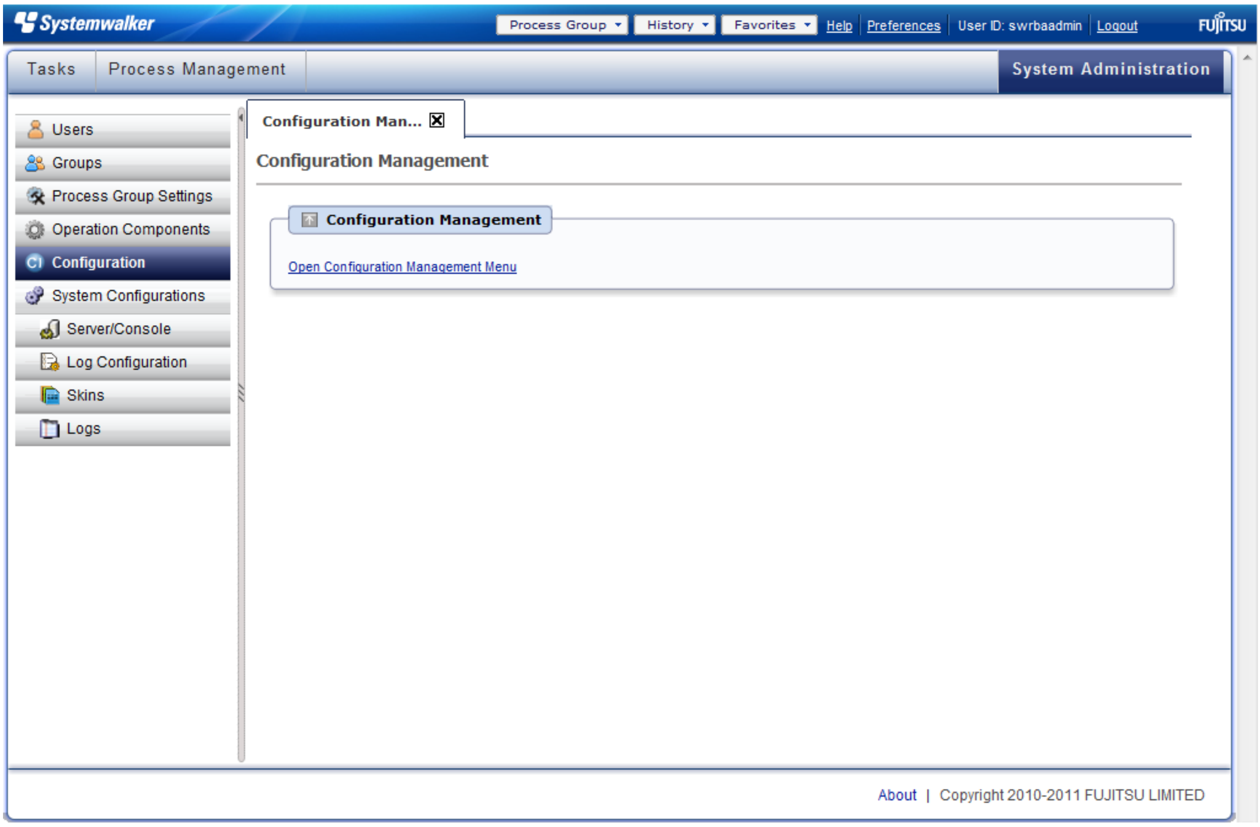
2. Select the server to be connected from the **Select Server Connection** dialog box.

   The browser starts and the login window is displayed.

   The login window does not appear if the Web console is already being displayed.

3. Enter the ID and password, and then click the **OK** button.

   The **Configuration Management** window opens.



4. Click **Open Configuration Management menu**.

   The **Configuration Management Menu** window opens.

5. Select the **CI View** menu.

   The **CI View** window opens.

6. Select the **Search** tab.

7. The XPath expression is displayed at the bottom left of the window as you select or enter the conditions in the window. Copy the XPath expression, and then paste it into the **Value** field in the Studio.

6.2 Output Definitions for Operation Component Nodes

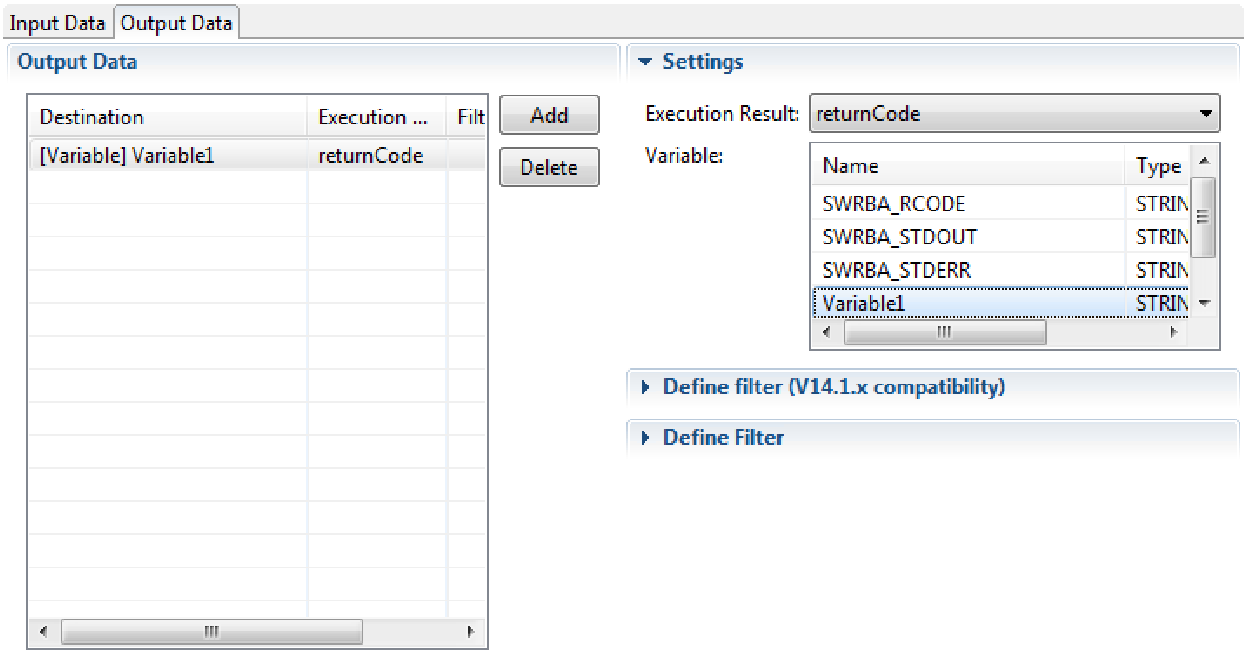
It is possible to define input and output information for operation component nodes used when operation components are called. The following explains output information definitions.

Refer to "6.7 Adding and Editing Nodes" for information on operations in nodes, including in operation component nodes.

6.2.1 Defining Output Information

This section explains output information that is used when operation components are called.

Output Data tab



Destination	Execution ...	Filt
[Variable] Variable1	returnCode	

Execution Result: returnCode

Variable:

Name	Type
SWRBA_RCODE	STRIN
SWRBA_STDOUT	STRIN
SWRBA_STDERR	STRIN
Variable1	STRIN

Output information is not displayed in the **Output Data** section immediately after the operation component node is allocated to the Process Definition Editor. If necessary, click the **Add** button and add output information. Select the output information in the list to display the **Settings** section to the right where output information can be edited.

Note

- The default output information (SWRBA_RCODE/SWRBA_STDOUT/SWRBA_STDERR) in the **Output Data** section of the operation component nodes in Studio V14.1.0A or earlier cannot be edited (deleted).

- The output information in **Output Data** of the operation component nodes in Studio V14.1.0A or earlier does not have a **Define Filter** section displayed. The output information in **Output Data** of the operation component nodes in Studio V15.0 or later displays a **Define filter (V14.1.x compatibility)** section and a **Define Filter** section.

- The **Output Data** of the operation component nodes created in Studio V15.0 or later does not have a **Define Filter** section displayed.

Items in the Output Data section of the Output Data tab

Item name	Description
Execution results	Displays the information to be stored in the output destination. returnCode: End code message : Message indicating the results There are different types of execution results for each type of operation component node not mentioned above. Refer to "Reference for Operation Components" in the *Systemwalker Runbook Automation Reference Guide* for details.
Filter	Displays the number of filters defined. Example: "n and JavaScript filters" Nothing is displayed if a filter has not been defined.
Add	This button is used to add new output information. The following new output information is added to the list. Output destination: **Variable**-- Unselected - Execution results: returnCode (nodes located in Studio earlier than V15.0.0) Execution results: message (nodes located in V15.0 Studio)
Delete	This button is used to delete output information. * The **Delete** button is unavailable if you have selected output information that has been pre-prepared for nodes in Studio V14.1.0A or earlier.

Items in the Settings section of the Output Data tab

Item name	Description
Execution results	Select output information. * The items shown in the list depend on the operation component node type. Refer to "Reference for Operation Components" in the *Systemwalker Runbook Automation Reference Guide* for details.
Variable	Select the name of the variable (user defined attribute) where the output information will be stored. Click the **Name** or **Type** column to sort in ascending or descending order.
Filter definitions (V14.1.x compatible)	
Filter name	Displays the names of filters. (Cannot be edited)
Add	Define filters. The **Define Filter** dialog box is displayed. Refer to "6.5 Defining Operation Component Filters (Using Java Scripts)" for information on how to setup filters using Java scripts. This button is grayed out if a filter has been added.
Edit	Edit a filter. The **Define Filter** dialog box is displayed. This button is grayed out if a filter has not been added.
Delete	Delete filters. This button is grayed out if a filter has not been added.

Item name	Description
Define Filter	
-	Displays a list of the filters that have been set. (Cannot be edited)
Add	Add filters. The **Filter settings** dialog box is displayed. This new filter format enables script-less multi-staged filtering. Refer to Filtering Output in the *Systemwalker Runbook Automation Operation Guide* for information on how to configure filters. Refer to "6.3 Defining Operation Component Filters (Script-less)" for information on script-less filtering. This button is grayed out if a filter has been added.
Edit	Edit a filter. The **Filter settings** dialog box is displayed. This button is grayed out if a filter has not been added.
Delete all	Delete filters. All filters are deleted. To delete individual filters, do so from the **Filter settings** window. This button is grayed out if a filter has not been added.

Note

Filters (V14.1.x compatible) using JavaScript are executed after all script-less filters have been run on the output information.

6.3 Defining Operation Component Filters (Script-less)

Using script-less filter definitions means that is no longer necessary to write Java scripts to process the execution results (output information) of operation components.

Use is explained below.

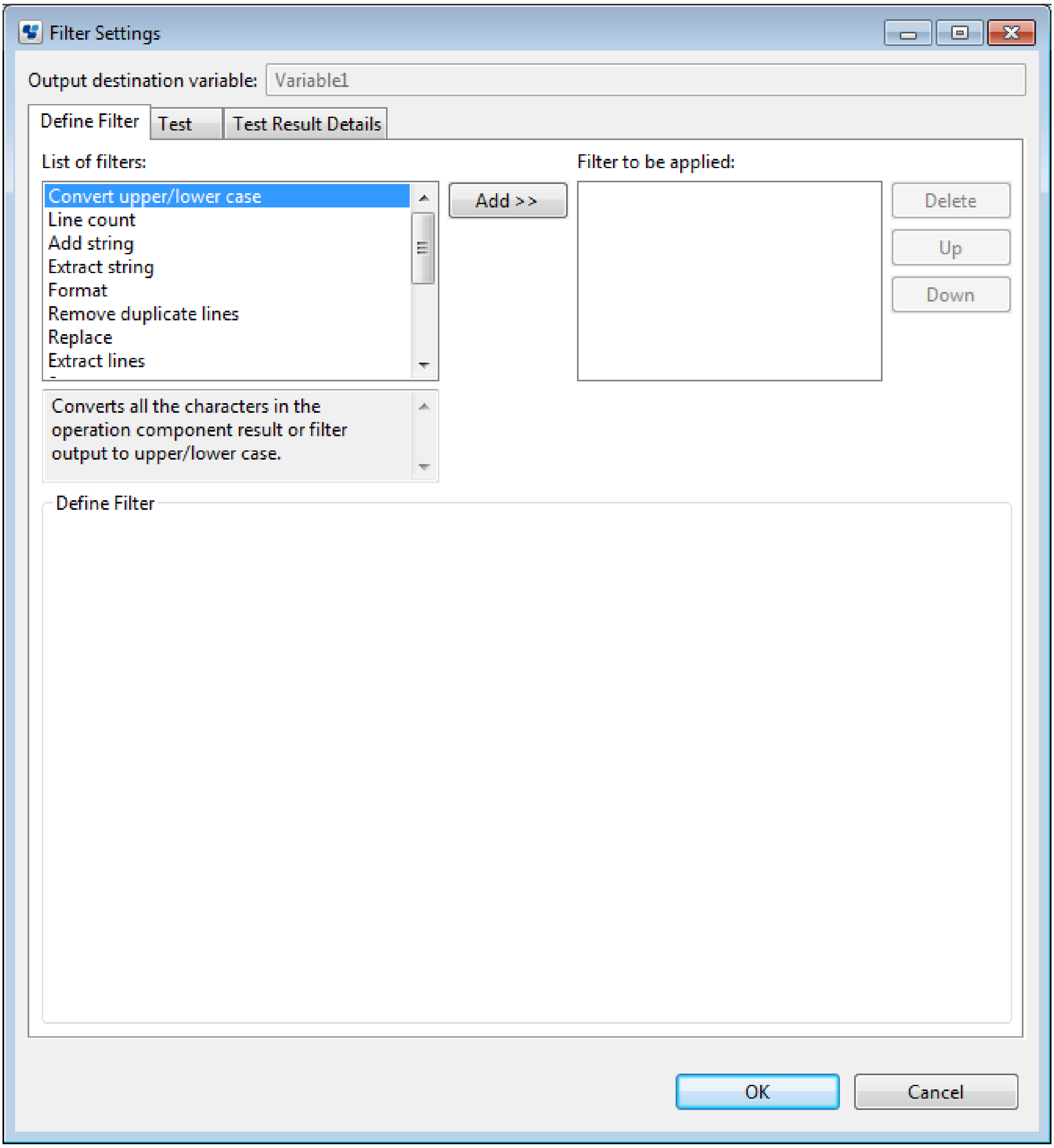
6.3.1 Filter Settings

The following explains how to configure filters for the output information of operation components.

1. Select the output information used to set the filter in the **Output Data** section in **Properties** >> **IO Setting** >> **Output Data** tab.

2. Click the **Settings** >> **Define Filter** >> **Add** button.

   The **Filter settings** dialog box will open.



Output destination variable: Variable1

List of filters:

- Convert upper/lower case
- Line count
- Add string
- Extract string
- Format
- Remove duplicate lines
- Replace
- Extract lines

Converts all the characters in the operation component result or filter output to upper/lower case.

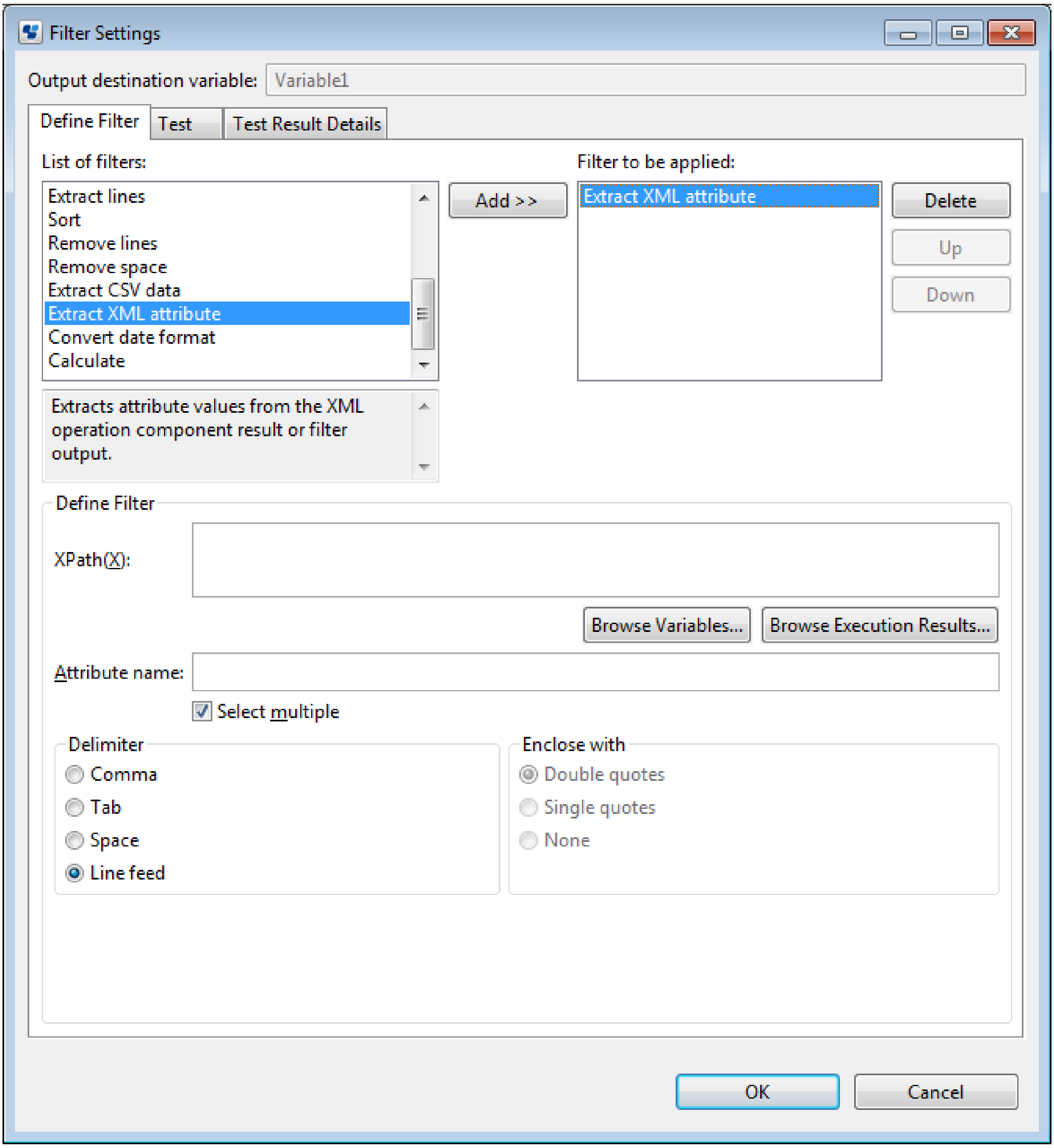
3. Select the filter to apply from the **List of filters** and click the **Add** button.

   The filter is added to the **Filter to be applied** list.

   Up to 10 filters can be configured for one piece of output information.

4. Select the filter that was added to the **Filter to be applied** list.

   The filter settings are displayed in the **Define Filter** area.



Filter Settings

Output destination variable: Variable1

List of filters:

- Extract lines
- Sort
- Remove lines
- Remove space
- Extract CSV data
- Extract XML attribute
- Convert date format
- Calculate

Extracts attribute values from the XML operation component result or filter output.

Filter to be applied: Extract XML attribute

Define Filter

XPath(X):

Attribute name:

☑ Select multiple

Delimiter

○ Comma

○ Tab

○ Space

● Line feed

Enclose with

● Double quotes

○ Single quotes

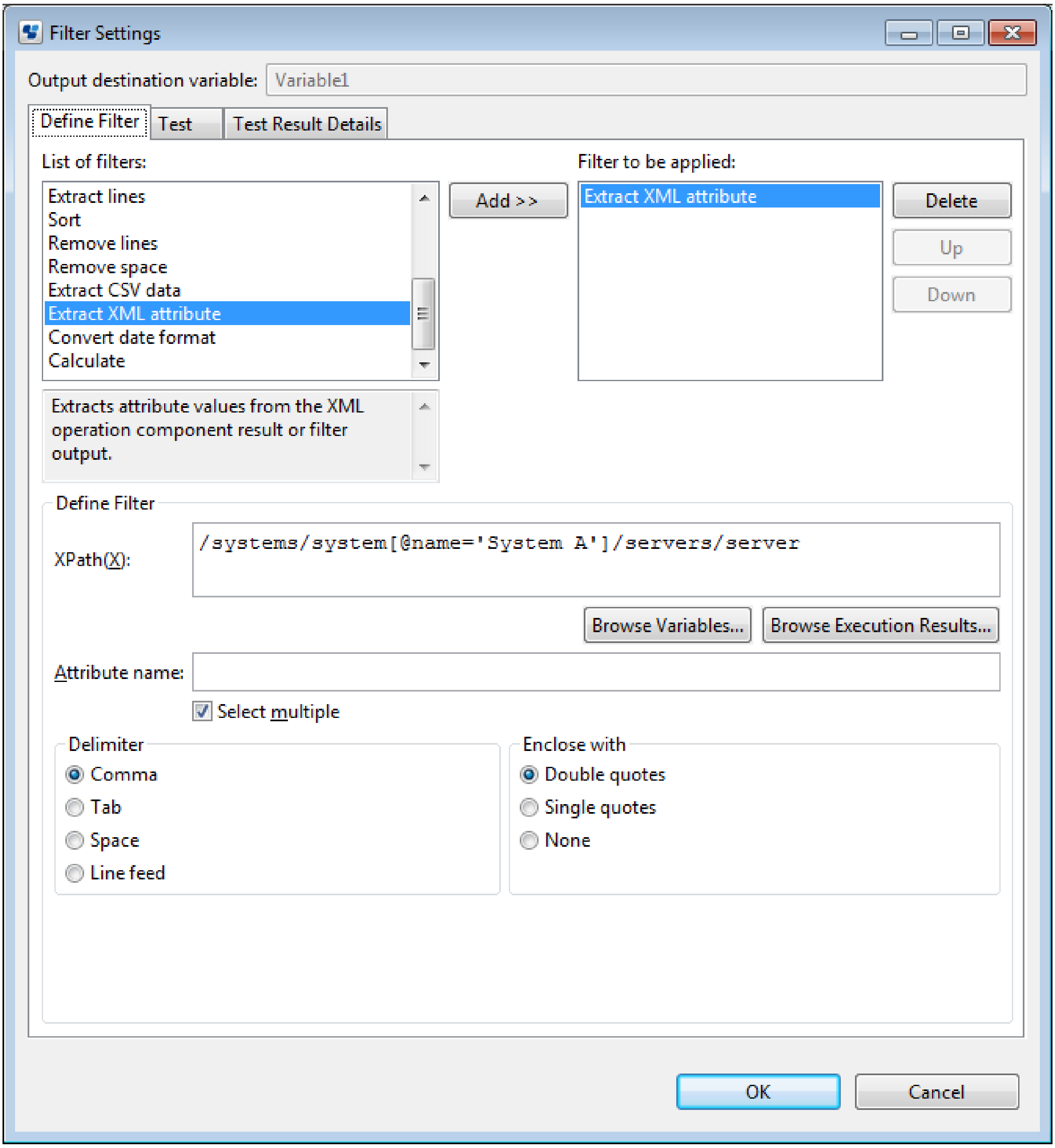
○ None

5. Set the filter items in the **Define Filter** area.
6. Click the **OK** button.

6.3.2 Testing Filters

You can test the filters in the Studio.

1. Add a filter in the **Define Filter** tab and set the items.



Filter Settings

Output destination variable: Variable1

List of filters:

Filter to be applied: Extract XML attribute

Extracts attribute values from the XML operation component result or filter output.

Define Filter

XPath(X): /systems/system[@name='System A']/servers/server

Attribute name:

☑ Select multiple

Delimiter: ● Comma ○ Tab ○ Space ○ Line feed

Enclose with: ● Double quotes ○ Single quotes ○ None

2. Select the **Test** tab.

3. Enter the test data in the **Input** field and click the **Run test** button.

   The filter test is run and the results are displayed in the **Output** field.

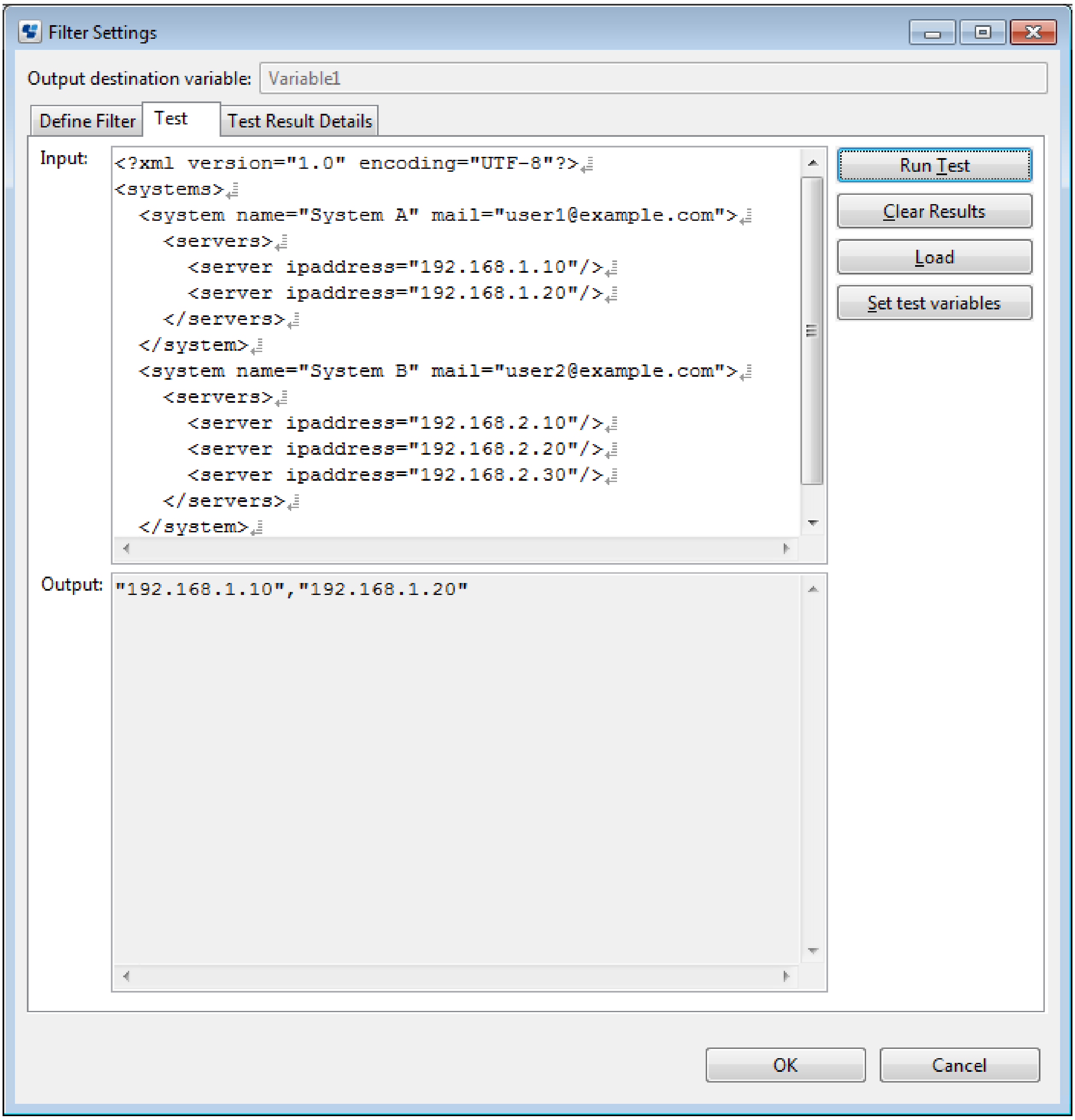
   Click the **Clear results** button to clear **Output** field contents. Click the **Load** button to load test data from a file.

   The content of the error appears in the **Output** field if the filter ends in an error.

   Click the **Set test variables** button to display the **Set test variables**dialog box.

   Press the **OK** button in the **Filter settings** dialog box to set the value of the variable set in the Set test variables dialog box - this value will be maintained until Studio stops.

*The variable value is maintained in a process definition unit.



Output destination variable: Variable1

Input:

```
<?xml version="1.0" encoding="UTF-8"?>
<systems>
  <system name="System A" mail="user1@example.com">
    <servers>
      <server ipaddress="192.168.1.10"/>
      <server ipaddress="192.168.1.20"/>
    </servers>
  </system>
  <system name="System B" mail="user2@example.com">
    <servers>
      <server ipaddress="192.168.2.10"/>
      <server ipaddress="192.168.2.20"/>
      <server ipaddress="192.168.2.30"/>
    </servers>
  </system>
```

Output:

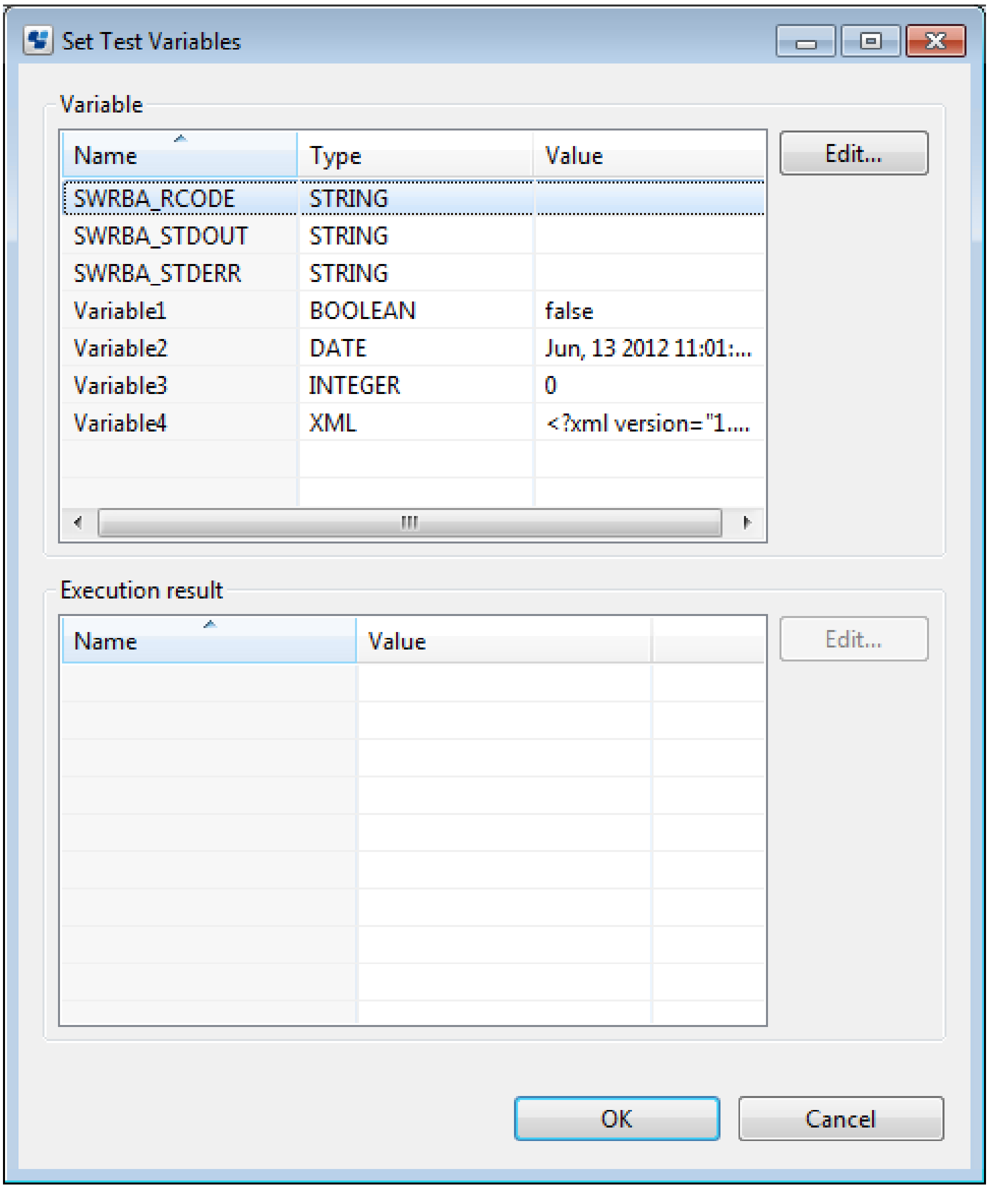
```
"192.168.1.10","192.168.1.20"
```

Set test variables

In the **Set test variables** dialog box, the following list is displayed:

- List of variables (UDA) defined in process definition

- List of execution results used in the Filter Definition of each operation component node



Set Test Variables

Variable

Name	Type	Value
SWRBA_RCODE	STRING	
SWRBA_STDOUT	STRING	
SWRBA_STDERR	STRING	
Variable1	BOOLEAN	false
Variable2	DATE	Jun, 13 2012 11:01:...
Variable3	INTEGER	0
Variable4	XML	<?xml version="1....

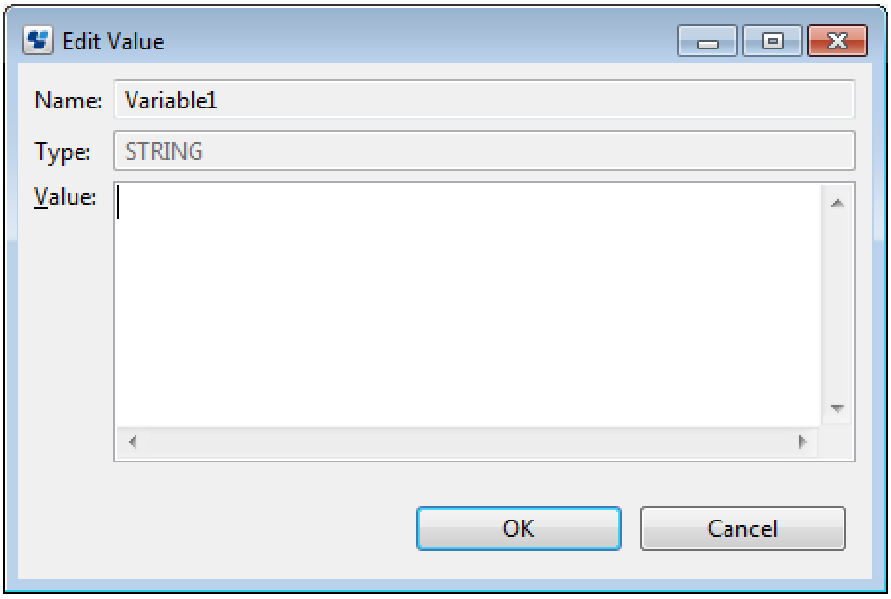
Execution result

Name	Value

The STRING/INTEGER/BOOLEAN/DATE/XML variable types are displayed in the **Variable** list - variables with names starting with two underscores (system defined attributes) are not displayed.

Clicking the **Edit** button displays the **Edit Value** dialog box, where it is possible to set values for the variables and the execution results.

Edit Value dialog box (type: STRING)

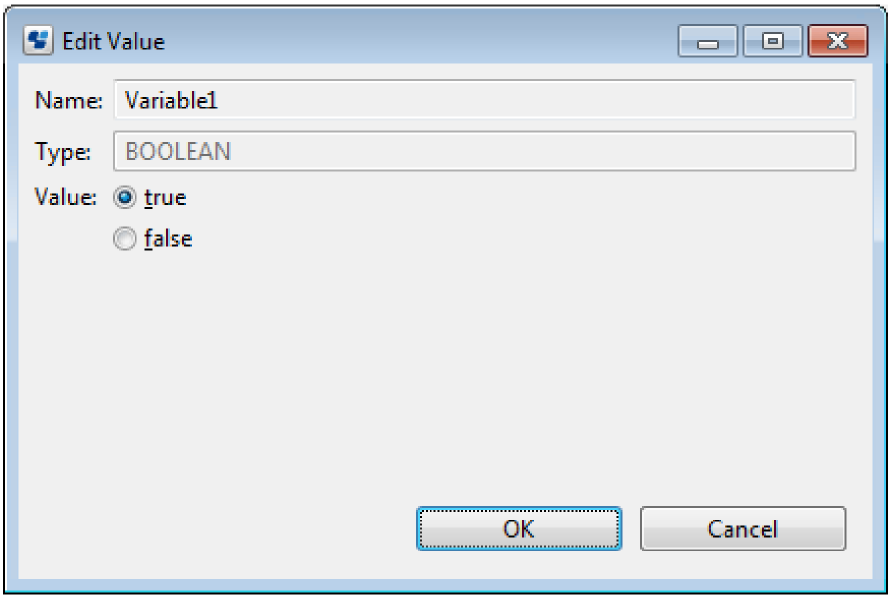


Name: Variable1

Type: STRING

Value:

Edit Value dialog box (type: BOOLEAN)

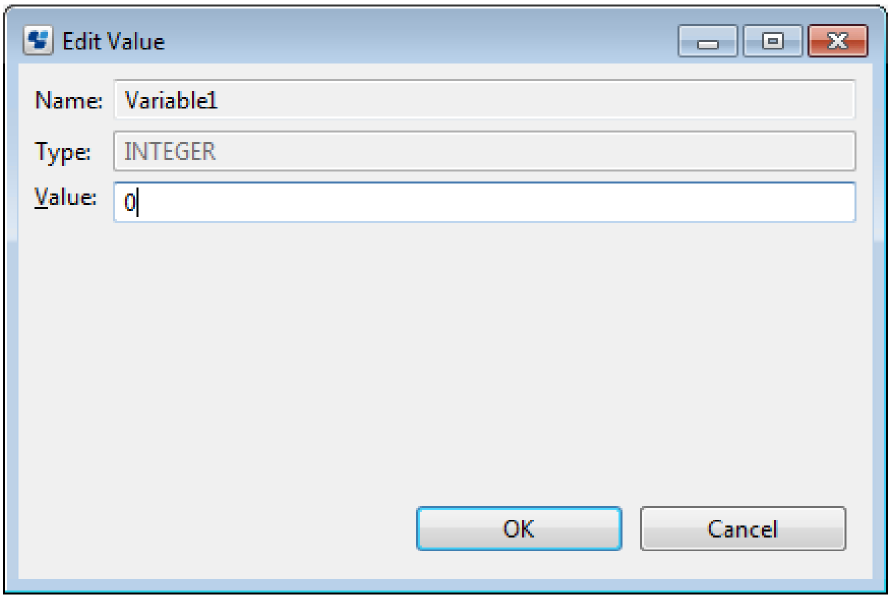


Name: Variable1

Type: BOOLEAN

Value: ● true ○ false

Edit Value dialog box (type: INTEGER)



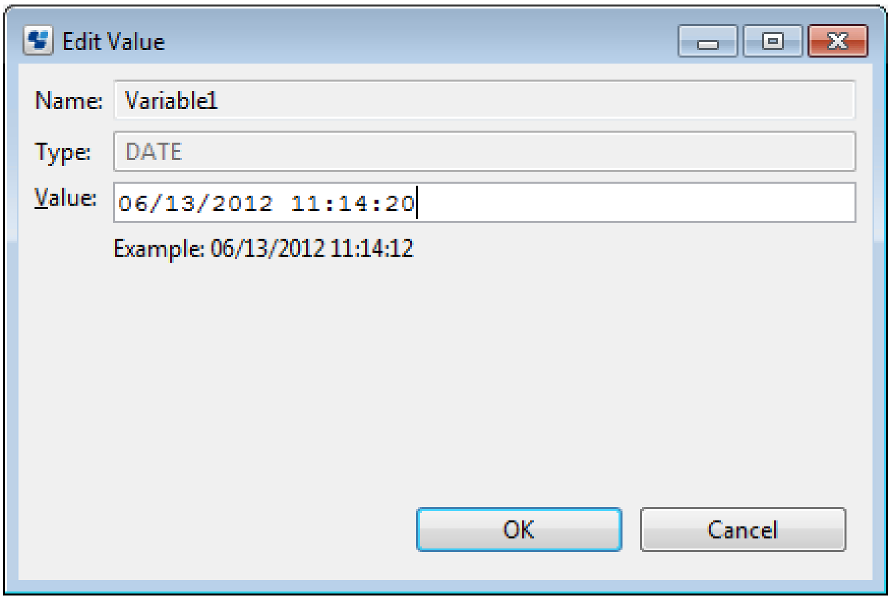
Name: Variable1

Type: INTEGER

Value: 0

Only numerals can be specified in the **Value** field.

Edit Value dialog box (type: DATE)



Name: Variable1

Type: DATE

Value: 06/13/2012 11:14:20

Example: 06/13/2012 11:14:12

Only date and time formats that accord with the OS settings can be specified in the **Value** field. If a wrong format is specified, clicking the **OK** button will display an error message.

When the **Edit Value** dialog box is displayed, the following date and time formats are displayed in the **Value** field, in accordance with the locale:

Japanese

- yyyy/MM/dd HH:mm:ss

English

- MM/dd/yyyy HH:mm:ss

Depending on the locale, it is possible to input date and time in one of the formats below (insert a half width space between date and time)

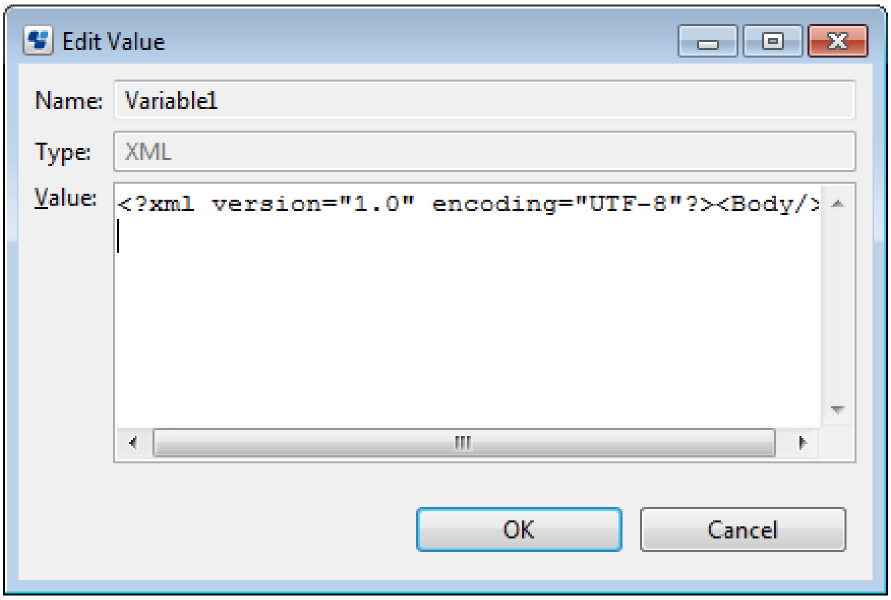
Japanese

- yyyy/MM/dd HH:mm:ss
- yyyy-MM-dd HH:mm:ss

English

- MM/dd/yyyy HH:mm:ss
- yyyy-MM-dd HH:mm:ss

Edit Value dialog box (type: XML)



Name: Variable1

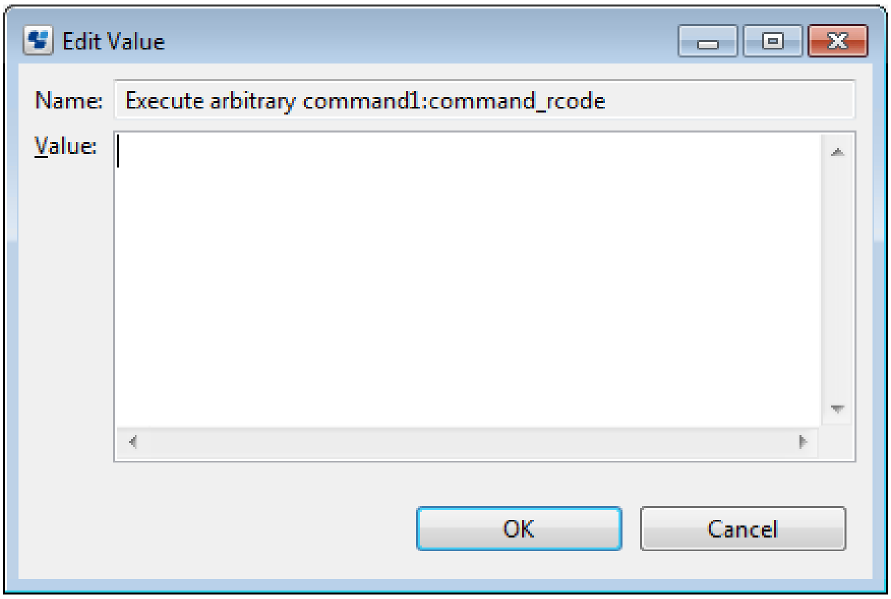
Type: XML

Value:

```
<?xml version="1.0" encoding="UTF-8"?><Body/>
```

If there is an error in the XML data specified in the **Value** field, clicking the **OK** button will display an error message.

Edit Value dialog box (execution results)

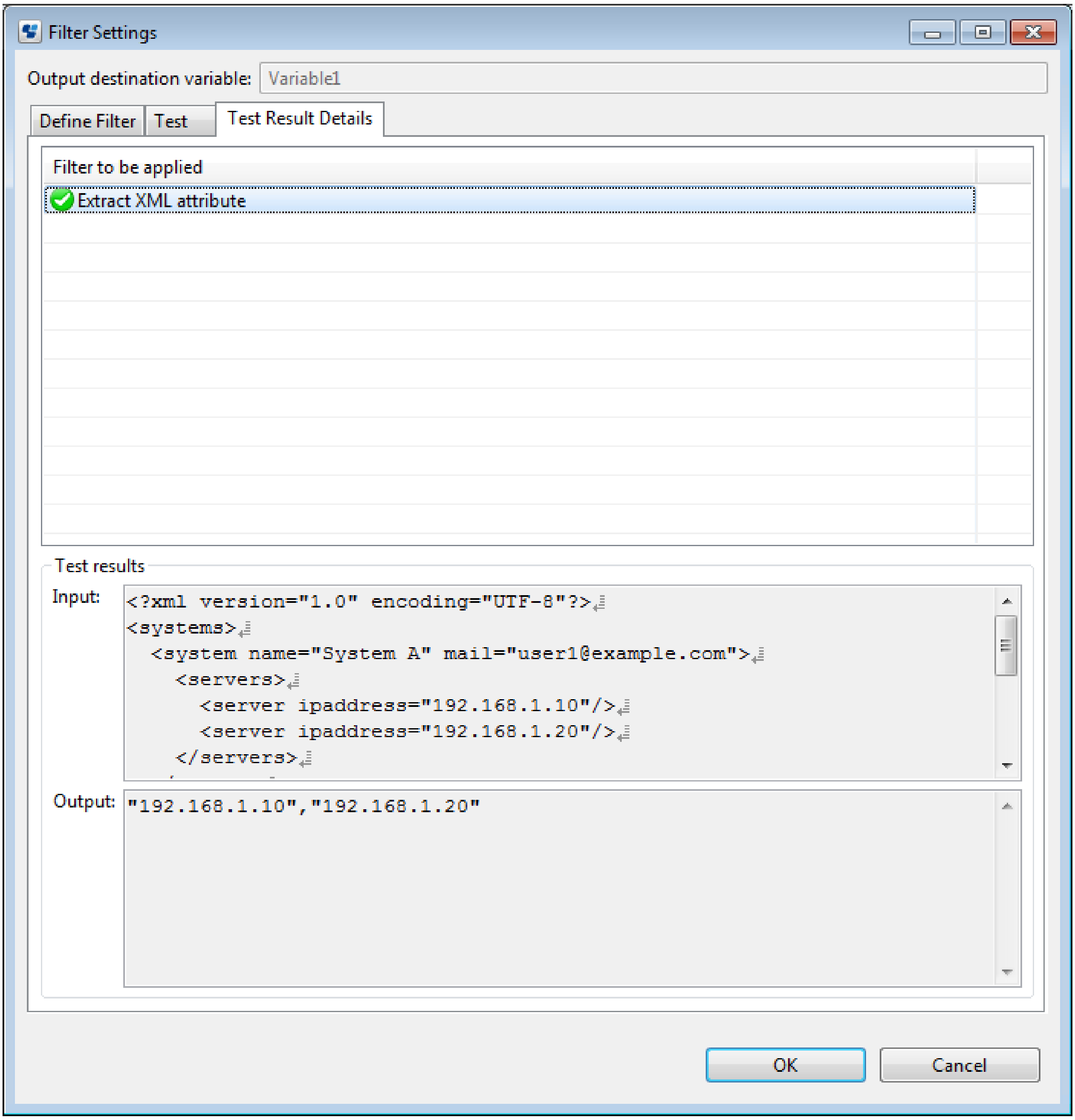


Name: Execute arbitrary command1:command_rcode

Value:

4. Select the **Test result details** tab to view the execution results for each filter.

   The filters appear in the **Filter to be applied** list with icons. (✓: normal, ✗: error). Select the item in the **Filter to be applied** list to show the input and execution results for the filters in the **Test results** area. Select a filter with the ✗ icon to display the content of the error in the **Output** field of the **Test results** area. (The content of errors of XML-related filters is the error messages held by exception objects.)



Output destination variable: Variable1

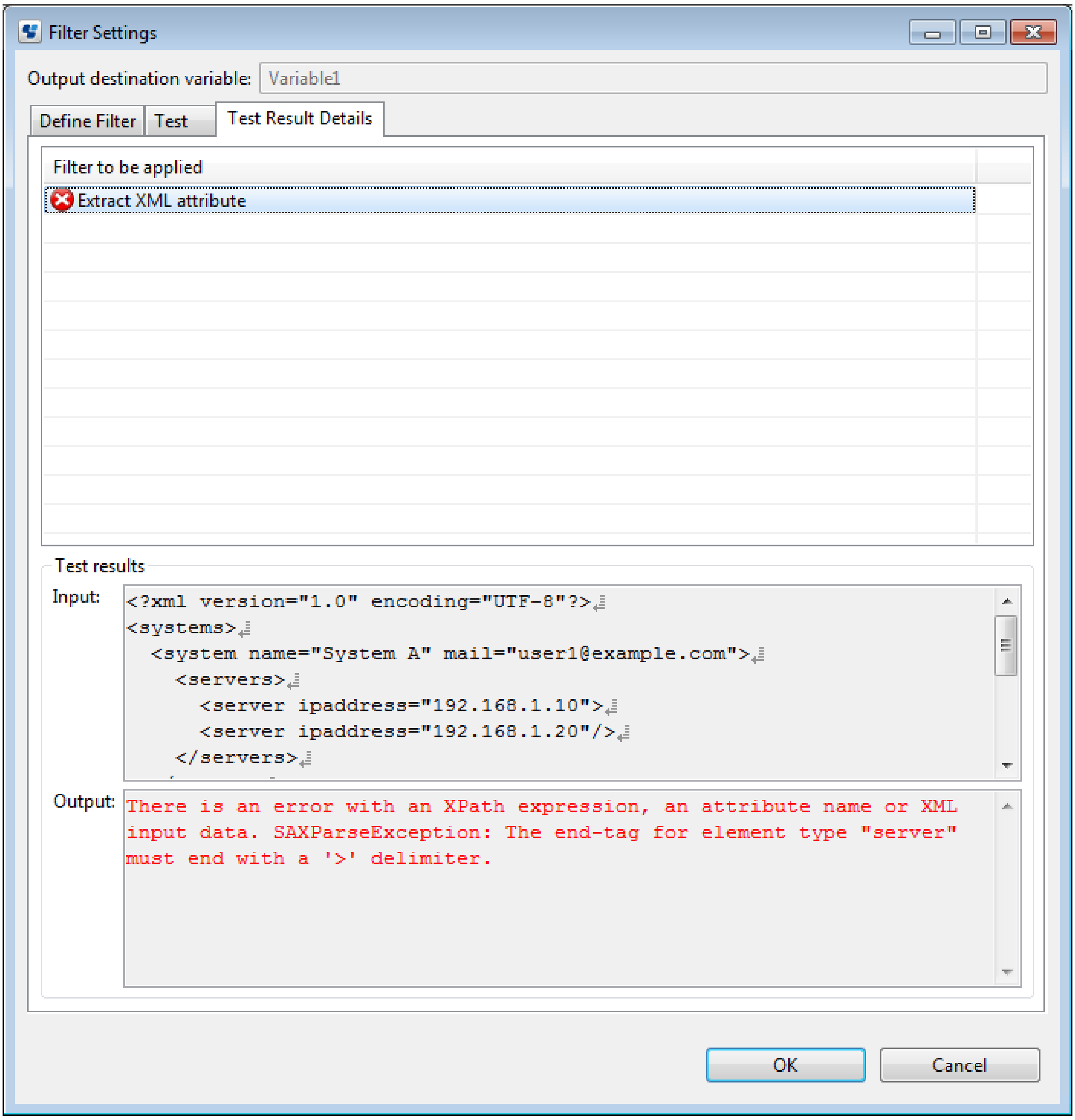
Filter to be applied: ✓ Extract XML attribute

Input:

```
<?xml version="1.0" encoding="UTF-8"?>
<systems>
  <system name="System A" mail="user1@example.com">
    <servers>
      <server ipaddress="192.168.1.10"/>
      <server ipaddress="192.168.1.20"/>
    </servers>
```

Output:

```
"192.168.1.10","192.168.1.20"
```



```
<?xml version="1.0" encoding="UTF-8"?>
<systems>
  <system name="System A" mail="user1@example.com">
    <servers>
      <server ipaddress="192.168.1.10">
      <server ipaddress="192.168.1.20"/>
    </servers>
```

```
There is an error with an XPath expression, an attribute name or XML
input data. SAXParseException: The end-tag for element type "server"
must end with a '>' delimiter.
```

6.4 Filter Details

The following filters are provided in Systemwalker Runbook Automation Studio:

- Uppercase and lowercase conversion
- Row count
- Append strings
- Extract strings
- Format
- Delete duplicate rows
- Replace
- Extract rows

- Sort
- Delete rows
- Delete spaces
- Extract CSV data
- Extract XML attributes
- Convert date format
- Operations

Point

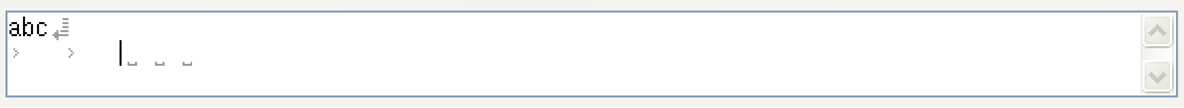
Specifying input data for the filter:

- If the input data is blank, the output result will be empty strings for the following filters:
    - **Convert upper/lower case** filter
    - **Extract string** filter
    - **Format** filter
    - **Remove duplicate lines** filter
    - **Replace** filter
    - **Extract lines** filter
    - **Sort** filter
    - **Remove lines** filter
    - **Remove space** filter
    - **Extract XML attribute** filter
- If the input data is blank, an error will occur when executing the following filters:
    - **Convert date format** filter
    - **Calculate** filter

About display of whitespace characters

It is possible to display whitespace characters (half-width spaces, full-width spaces, tabs, and linefeeds) in each text field, just as with a normal text editor. (Excludes fields that only accept numeric input.)

Display example:



abc

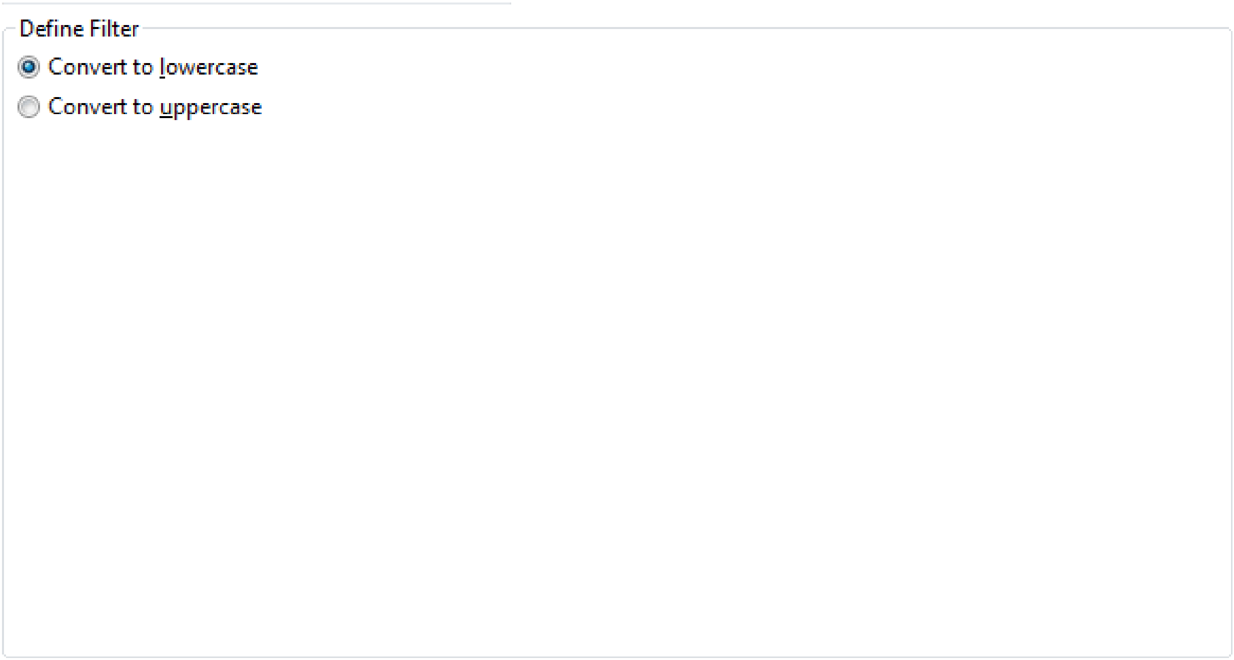
To enable/displayed display of whitespace characters, select **Window** >> **General** >> **Settings** >> **Editor** >> **Text Editor**.

Note

- When executing filters that process row-by-row, the linefeed code in the output result is converted to LF (linefeed).
- No linefeed code is attached to the end of the row if the filter execution results are output in a single row. A linefeed (LF) code will be attached to the end of each row when the filter execution results are output over multiple rows.

6.4.1 Convert upper/lower case Filter

This filter converts all characters in the results of an operation component or in filter output to either uppercase or lowercase.



Define Filter

● Convert to lowercase

○ Convert to uppercase

Convert to lowercase

Select to convert all characters to lowercase. (Default)

Convert to uppercase

Select to convert all characters to uppercase.

6.4.2 Line count Filter

This filter counts all rows in the results of an operation component or in filter output.

Rows consisting only of a linefeed character are also counted. There are no items to set for this filter.

Example

Example 1:

Input:	aaa[linefeed] bbb[linefeed]
Output:	2

Example 2:

Input:	aaa[linefeed] bbb
Output:	2

Example 3:

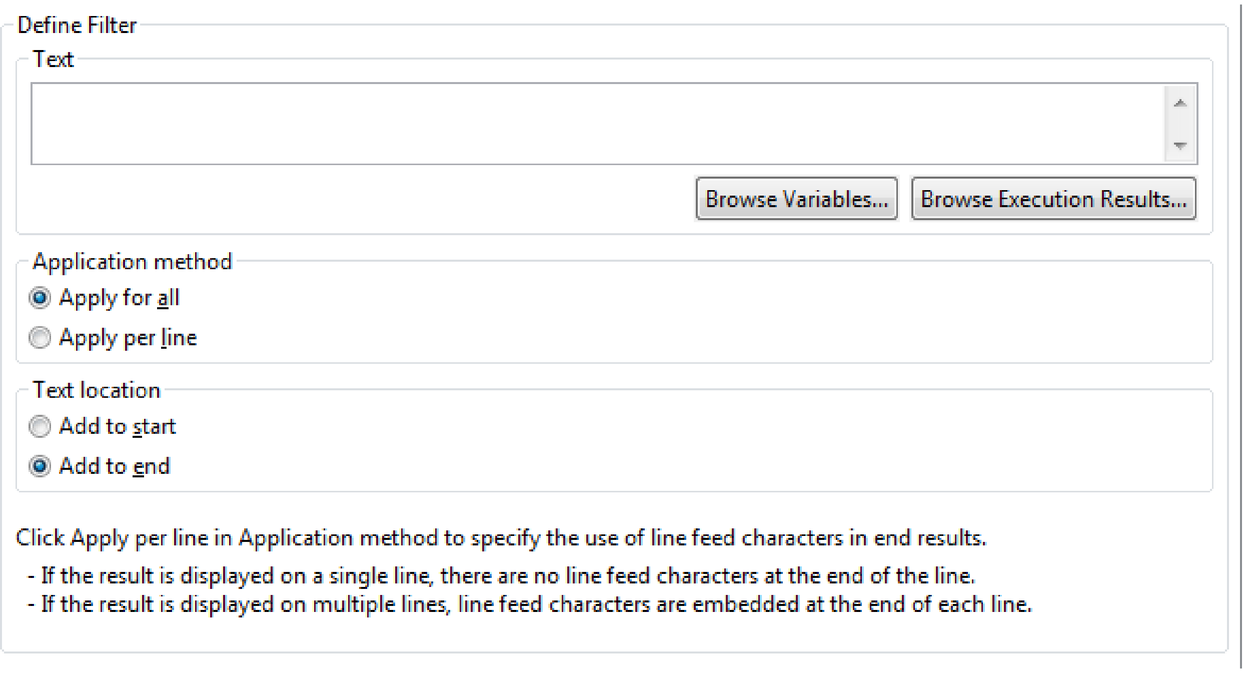
Input:	[linefeed]
Output:	1

Example 4:

Input:	
Output:	0

6.4.3 Add string Filter

This filter adds text or the content of variables to the results of an operation component or filter output.



Application method

● Apply for all

○ Apply per line

Text location

○ Add to start

● Add to end

Click Apply per line in Application method to specify the use of line feed characters in end results.

- If the result is displayed on a single line, there are no line feed characters at the end of the line.
- If the result is displayed on multiple lines, line feed characters are embedded at the end of each line.

Text

Enter the text to be added to the results of an operation component or filter output. To embed the variable (UDA) value and execution results of the previous operation component in the **Text** field when the filter is applied, follow the steps below:

- To embed the value, specify "@{uda: variable name}" or click the **Browse variables** button and select a variable from the **Browse variables** dialog box.

The types of variables (UDA) available are STRING, INTEGER, BOOLEAN, DATE, and XML.

If the variable (UDA) type is INTEGER, BOOLEAN, or DATE, the strings for these types are appended. For example, if the BOOLEAN-type variable (UDA) is specified, the string "true" or "false" is appended.

- To embed the execution results of the previous operation component, specify "@{:node name:execution results}" or click the **Browse execution results** button and select an execution result from the **Browse execution results** dialog box.

The filter will end in an error if the **Text** field is empty.

The input can be up to 512 characters long.

Applications

Select **Apply for all** to apply the filter to the entire filter input. Select **Apply per line** to apply the filter row-by-row.

The default is **Apply for all**.

Text location

Select **Add to start** to append text, variables (UDA), or execution results to the front of a row. Select **Add to end** to add to the end of a row.

The default is **Add to end**.

Note

Precautions when Testing Filters

- If "@{uda:variable name}" is specified in the **Text** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or the string "@{uda:variable name}" if no value is assigned.

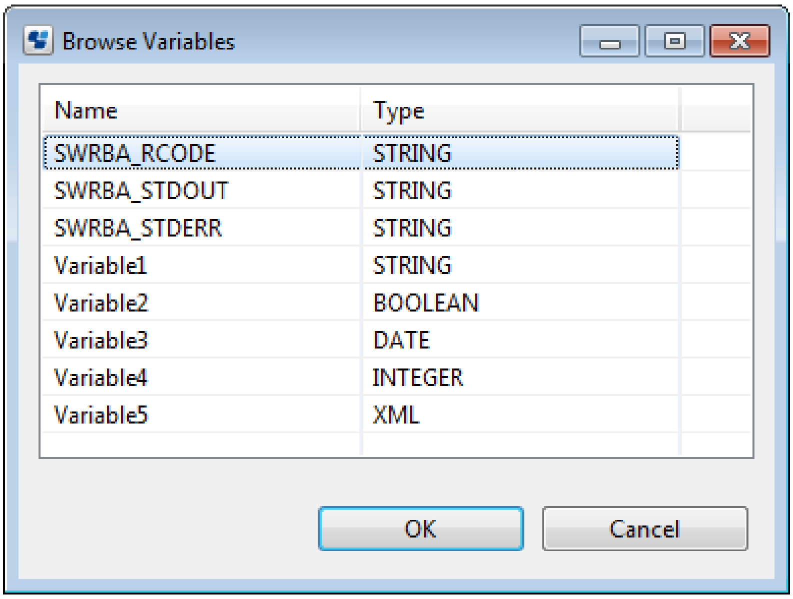
- If "@{:node name:execution results}" is specified in the **Text** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or the string "@{:node name:execution results}" if no value is assigned.

- When the filter is applied on the Management Server, these variables (UDA) or execution results are replaced with the stored values.

*1: Refer to "6.3.2 Testing Filters" for information on the **Set test variables**dialog box.

Figure 6.4 Browse variables dialog box



Name	Type
SWRBA_RCODE	STRING
SWRBA_STDOUT	STRING
SWRBA_STDERR	STRING
Variable1	STRING
Variable2	BOOLEAN
Variable3	DATE
Variable4	INTEGER
Variable5	XML

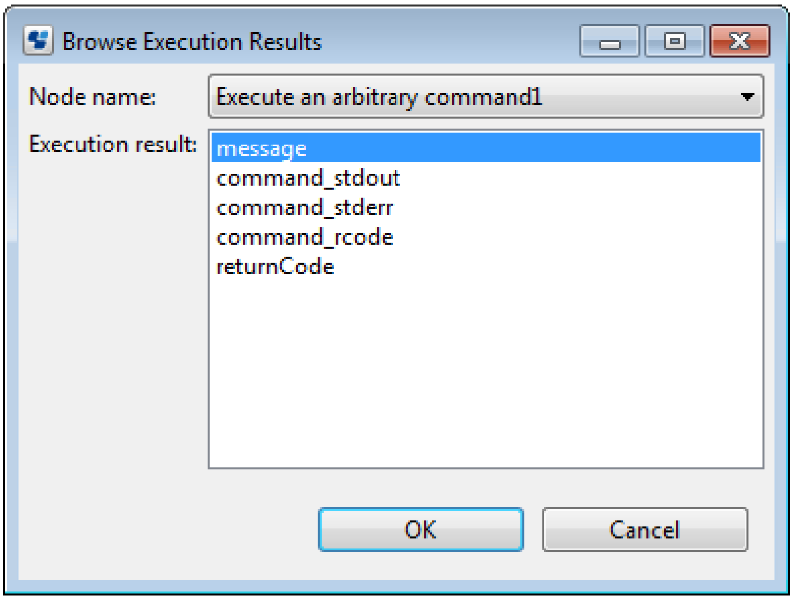
A list of variables (UDA) that can be selected is displayed in the **Variable** list.

Select the variable (UDA), click the **OK** button, and the string "@{uda:variable name}" is inserted into the cursor position.

Click the **Name** column to sort the list by the name of the variable (UDA). Click the **Type** column to sort by type. Each time you click the **Name** or **Type** columns, the sorting switches between ascending and descending. The default is ascending.

In the **Variable** list, variables starting with 2 underscores (system defined attributes) are not displayed.

Figure 6.5 Browse execution results dialog box



A list of nodes (operation component nodes) that can be selected is displayed in the **Node Name** combo box.

When you select the node name, a list of execution results for the node is displayed in **Execution results**.

Select the execution results, click the **OK** button, and the string "@{:node name:execution results}" is inserted into the cursor position.

Example 1:

Input:	`192.168.1.10[linefeed]`
	`192.168.1.20[linefeed]`

Text:	ping[space]
Applications	Applied overall
Location:	Added to the beginning of the row

Output:	`ping 192.168.1.10[linefeed]`
	`192.168.1.20[linefeed]`

Example 2:

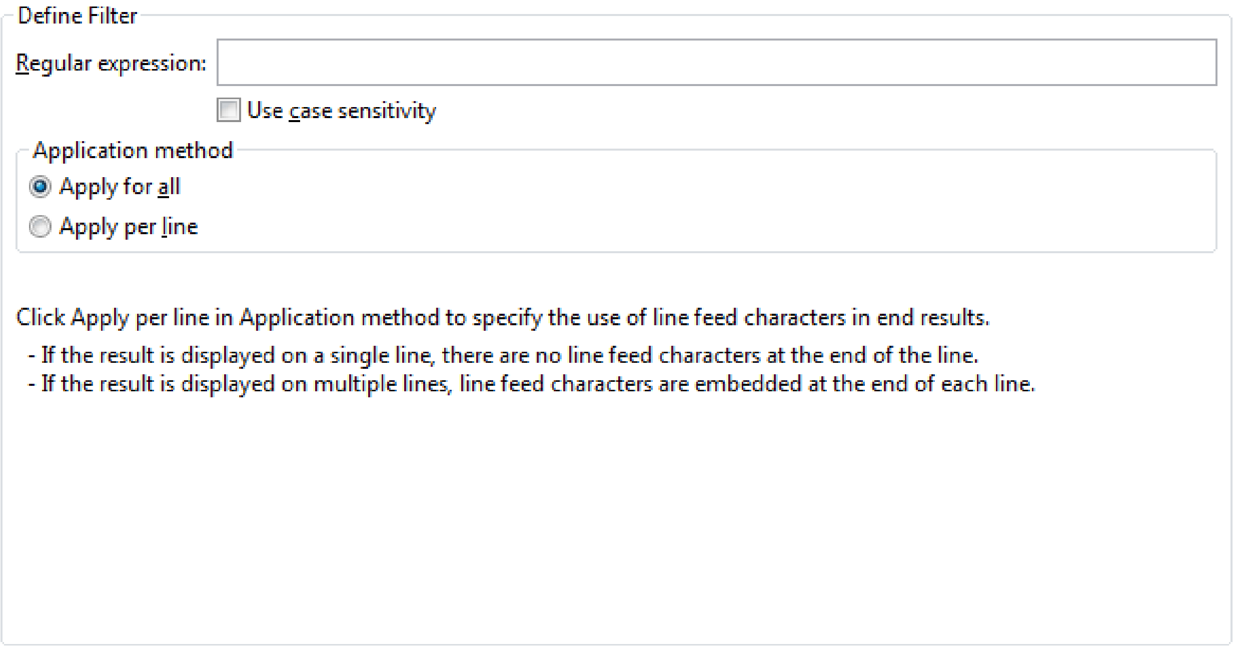
Input:	`192.168.1.10[linefeed]`
	`192.168.1.20[linefeed]`

Text:	ping[space]
Applications	Applied row-by-row
Location:	Added to the beginning of the row

Output:	`ping 192.168.1.10 [linefeed]`
	`ping 192.168.1.20 [linefeed]`

6.4.4 Extract string Filter

This filter extracts strings in the results of an operation component or in filter output that match regular expression patterns.



Define Filter

Regular expression:

□ Use case sensitivity

Application method

● Apply for all

○ Apply per line

Click Apply per line in Application method to specify the use of line feed characters in end results.

- If the result is displayed on a single line, there are no line feed characters at the end of the line.
- If the result is displayed on multiple lines, line feed characters are embedded at the end of each line.

Regular expression

Enter the regular expression used to extract the string.

The filter will end in an error if the **Regular expression** field is empty.

The input can be up to 512 characters long.

Point

How to write regular expressions:

Regular expression syntax	Matches
[abc]	Either a, b, or c
[a-z]	Any one lowercase alphabet
[A-Z]	Any one uppercase alphabet
[0-9]	A number between 0 and 9
\d	A number between 0 and 9
.	Any character
.*	Any string

In the syntax of regular expressions, dots (.) match any single character. Use the escape character (\) before the dot when the dot itself needs to be matched (i.e., \.)

Refer to the "**Java 2 Platform API Specification**" for information on regular expressions not mentioned above.

Use case sensitivity

Select to distinguish between upper and lower case in regular expressions.

The default is to ignore case.

Point

Specifying [a-zA-Z] in **Regular expression** when **Use case sensitivity** is selected is equivalent to specifying [a-z] when **Use case sensitivity** is cleared (see [Example 3] and [Example 4]).

Applications

Select **Apply for all** to apply the regular expressions to the entire filter input. Select **Apply per line** to apply row-by-row.

If **Apply per line** is selected and multiple strings match the regular expression, the extracted strings are output linked by linefeed characters (LF).

The default is **Apply for all**.

Example 1:

Input:	Ethernet adapter1:[linefeed]
	        IP Address. . . . . : 192.168.238.1[linefeed]
	        Subnet Mask . . . . : 255.255.255.0[linefeed]
	        Default Gateway . . . . :[linefeed]
	[linefeed]
	Ethernet adapter2:[linefeed]
	        IP Address. . . . . : 192.168.187.102[linefeed]
	        Subnet Mask . . . . : 255.255.255.0[linefeed]
	        Default Gateway . . . . : 192.168.187.1[linefeed]

Regular expression:	192\.168\.\d+\.\d+
Match case:	Off
Applications	Applied overall
Output:	192.168.238.1

Example 2:

Input:	Ethernet adapter1:[linefeed]
	        IP Address. . . . . : 192.168.238.1[linefeed]
	        Subnet Mask . . . . : 255.255.255.0[linefeed]
	        Default Gateway . . . . :[linefeed]
	[linefeed]
	Ethernet adapter2:[linefeed]
	        IP Address. . . . . : 192.168.187.102[linefeed]
	        Subnet Mask . . . . : 255.255.255.0[linefeed]
	        Default Gateway . . . . : 192.168.187.1[linefeed]

Regular expression:	192\.168\.\d+\.\d+
Match case:	Off
Applications	Applied row-by-row
Output:	192.168.238.1[linefeed] 192.168.187.102[linefeed] 192.168.187.1[linefeed]

Example 3:

Input:	1[linefeed] a[linefeed] A[linefeed]
Regular expression:	[a-zA-Z]
Match case:	On

Applications:	Applied row-by-row
Output:	`a[linefeed]A[linefeed]`

Example 4:

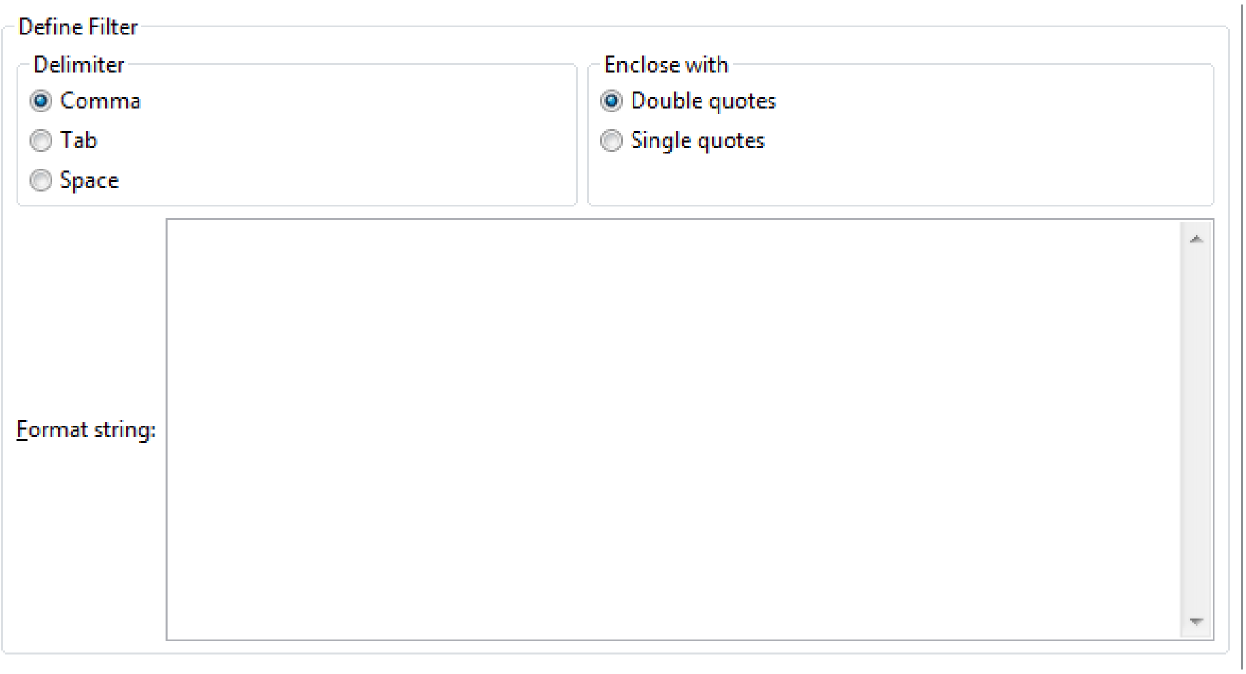
Input:	`1[linefeed]`
	`a[linefeed]`
	`A[linefeed]`

RegExp:	[a-z]
Match case:	Off
Applications:	Applied row-by-row

Output:	`a[linefeed]`
	`A[linefeed]`

6.4.5 Format Filter

This filter splits strings in the results of an operation component or in filter output according to delimiters, and then embeds these strings into strings with a specified format.



Delimiter

Select the delimiter to split the input data (results of an operation component or in filter output) into values (columns).

- Comma (Default)
- Tab
- Spaces

Enclose with

Select characters to enclose values:

- Double quotes (Default)
- Single quote

Information

- If delimiters are surrounded by enclosing characters, these stipulate the ranges of values. This is demonstrated with the following example data:

  "aaaa,bbbb","cccc","dddd"

  The input is divided into the following three values (columns) if Double quotes have been specified as the enclosing characters.

  1: aaaa,bbbb

  2: cccc

  3: dddd

  The input is divided into the following four values (columns) if Single quotes have been specified as the enclosing characters.

  1: "aaaa

  2: bbbb"

  3: "cccc"

  4: "dddd"

- Data that contains consecutive spaces is divided as follows if spaces have been specified as the delimiters:

  Before splitting: "aaaa"[[space][space]"bbbb"[space][space][space]"cccc"

  After splitting:

  1: "aaaa"

  2: Empty string

  3: "bbbb"

  4: Empty string

  5: Empty string

  6: "cccc"

Format string

Enter the format string where the divided text is to be embedded.

Specify the position for embedding using the format specifier ({n[*1]}).

If the number of pieces of text to embed exceeds the number of format specifiers ({n}), the extra text is ignored. If the number of format specifiers ({n}) exceeds the number of pieces of text to embed, the positions occupied by the extra specifiers are replaced with blanks. If there are multiple number format specifiers, then the same text is embedded wherever the same numbers appear.

The filter will end in an error if the **Format string** field is empty.

The input can be up to 512 characters long.

By using "[$number]", serial numbers (starting from 1) can be embedded according to the number of rows in the input data.

The format specifier is a value greater than 0.

Example 1:

Input:	`serverA[linefeed]`
	`serverB[linefeed]`

Column separator:	Comma
Text separator:	Double quotes
Formatting string:	<Parameter ID="**[$number]**"> <TargetUDA Name="hostname" Value="**{0}**"/> </Parameter>

Output:	`<Parameter ID="`**`1`**`">`
	`<TargetUDA Name="hostname" Value="`**`serverA`**`"/>`
	`</Parameter>`
	`<Parameter ID="`**`2`**`">`
	`<TargetUDA Name="hostname" Value="`**`serverB`**`"/>`
	`</Parameter>`

Example 2:

Input:	`"1" "server A"[linefeed]`
	`"2" "server B"[linefeed]`

Column separator:	Spaces
Text separator:	Double quotes
Formatting string:	<Parameter ID="**{0}**"> <TargetUDA Name="hostname" Value="**{1}**"/> </Parameter>

Output:	`<Parameter ID="`**`1`**`">`
	`<TargetUDA Name="hostname" Value="`**`server A`**`"/>`
	`</Parameter>`
	`<Parameter ID="`**`2`**`">`
	`<TargetUDA Name="hostname" Value="`**`server B`**`"/>`
	`</Parameter>`

Example 3:

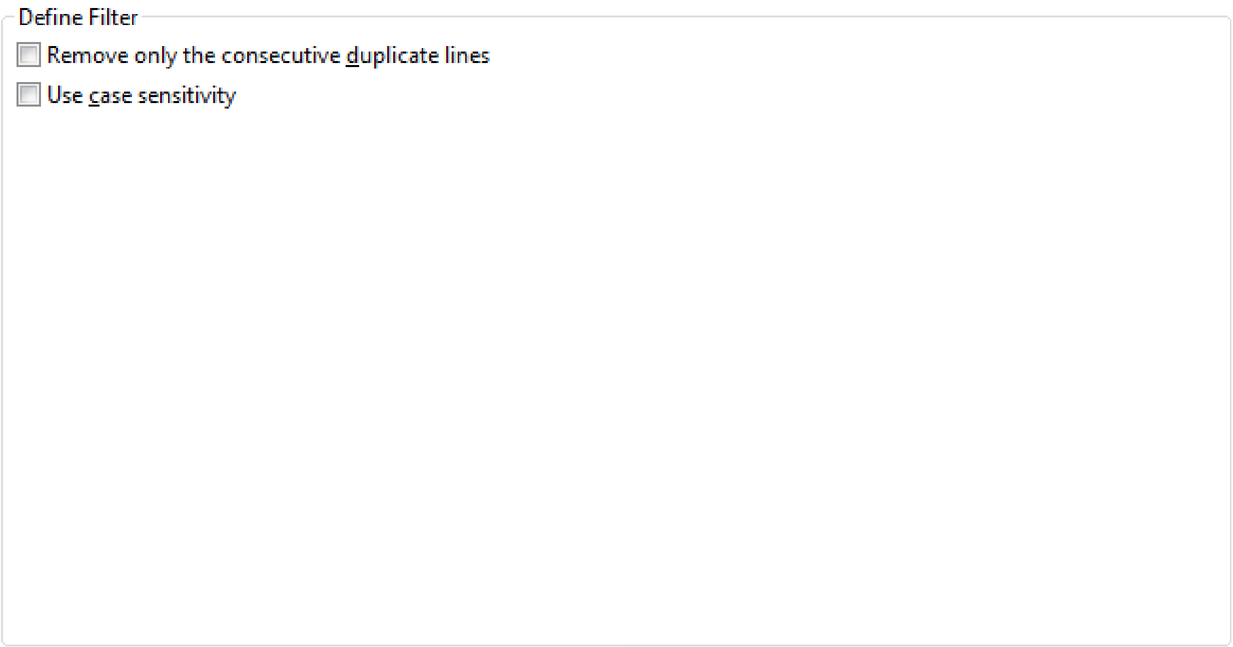
Input:	`1,serverA[linefeed]`
	`2,serverB[linefeed]`

Column separator:	Comma
Text separator:	Double quotes
Formatting string:	<Parameter ID="**{0}**" target="**{1}**"> <TargetUDA Name="hostname" Value="**{1}**"/> </Parameter>

Output:	`<Parameter ID="`**`1`**`" target="`**`serverA`**`">`
	`<TargetUDA Name="hostname" Value="`**`serverA`**`"/>`
	`</Parameter>`
	`<Parameter ID="`**`2`**`" target="`**`serverB`**`">`
	`<TargetUDA Name="hostname" Value="`**`serverB`**`"/>`
	`</Parameter>`

6.4.6 Remove duplicate lines Filter

This filter detects rows that are the same in the results of an operation component or in filter output and then deletes all but one of them. It is also possible to delete only those duplicate rows that occur consecutively.



Define Filter

□ Remove only the consecutive duplicate lines

□ Use case sensitivity

Remove only the consecutive duplicate lines

Select to apply the filter to consecutive duplicate rows only.

The default is off.

Use case sensitivity

Select to distinguish between upper and lower case when the same rows are detected.

The default is off.

Example 1:

Input:	AAAA[linefeed] BBBB[linefeed] CCCC[linefeed] BBBB[linefeed] BBBB[linefeed] DDDD[linefeed]
Delete consecutive duplicate rows:	Off
Match case:	Off
Output:	AAAA[linefeed] BBBB[linefeed] CCCC[linefeed] DDDD[linefeed]

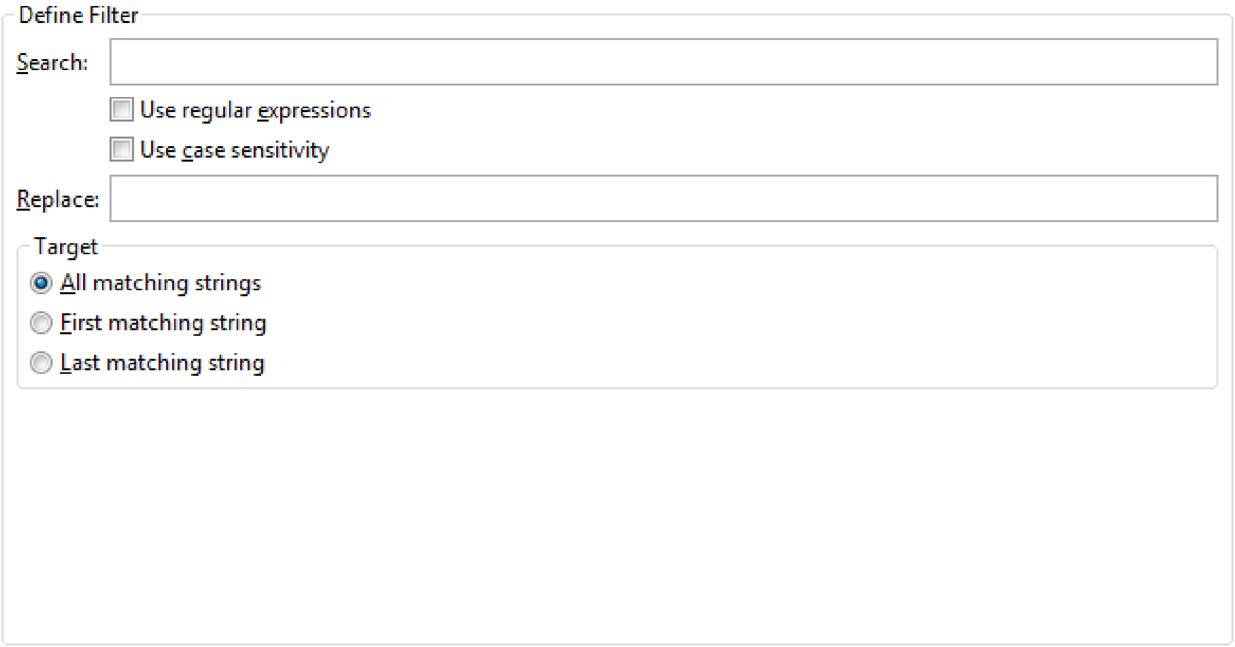
Example 2:

Input:	AAAA[linefeed] BBBB[linefeed] CCCC[linefeed] BBBB[linefeed] BBBB[linefeed] DDDD[linefeed]
Delete consecutive duplicate rows:	On
Match case:	Off

Output:	`AAAA[linefeed]`
	`BBBB[linefeed]`
	`CCCC[linefeed]`
	`BBBB[linefeed]`
	`DDDD[linefeed]`

6.4.7 Replace Filter

This filter finds the specified string in the results of an operation component or in filter output, and then replaces the string with another one.



Define Filter

Search:

□ Use regular expressions

□ Use case sensitivity

Replace:

Target

● All matching strings

○ First matching string

○ Last matching string

Search

Enter the string to search and replace.

The input can be up to 512 characters long.

The filter will end in an error if the **Search** field is empty.

Use regular expressions

Select when regular expressions are to be used in the **Search** field.

The default is off.

Use case sensitivity

Select to distinguish between upper and lower case in searches.

The default is off.

Replace

Enter the string that will replace the string (or string matching the regular expression) in the **Search** field.

The input can be up to 512 characters long.

If the **Replace** field is empty, the string (or string matching the regular expression) in the **Search** field will be replaced by spaces.

Target

Select the target for replacement.

Select **First matching string** to replace only the first match. Select **Last matching string** to replace only the last match. Select **All matching strings** to replace all matches.

- **First matching string**
- **Last matching string**
- **All matching strings** (Default)

Example 1:

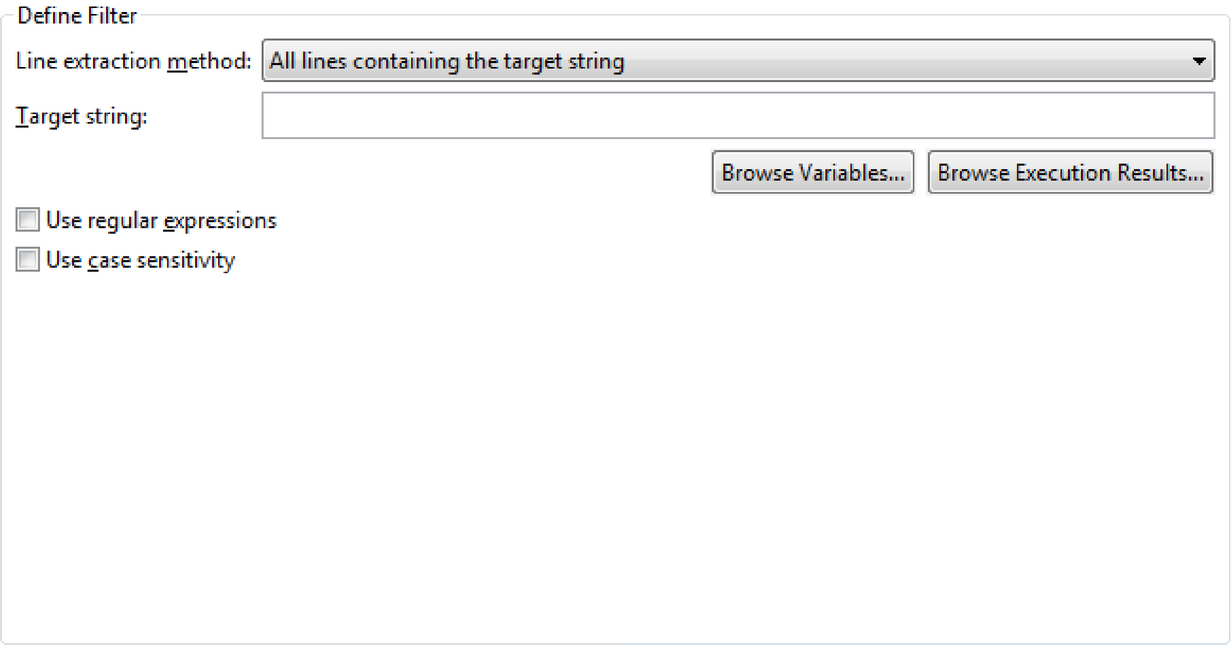
Input:	`Ethernet adapter1:`
	`        IP Address. . . . . . . . . . . . : 192.168.238.1`
	`        Subnet Mask . . . . . . . . . . . : 255.255.255.0`
	`        Default Gateway . . . . . . . . . :`
	
	`Ethernet adapter2:`
	`        IP Address. . . . . . . . . . . . : 192.168.187.102`
	`        Subnet Mask . . . . . . . . . . . : 255.255.255.0`
	`        Default Gateway . . . . . . . . . : 192.168.187.1`

Search:	192\.168\.\d+\.\d+
Use regular expressions:	On
Match case:	Off
Replace:	xxx.xxx.xxx.xxx
Target:	All matching strings

Output:	`Ethernet adapter1:`
	`        IP Address. . . . . . . . . . . . : xxx.xxx.xxx.xxx`
	`        Subnet Mask . . . . . . . . . . . : 255.255.255.0`
	`        Default Gateway . . . . . . . . . :`
	
	`Ethernet adapter2:`
	`        IP Address. . . . . . . . . . . . : xxx.xxx.xxx.xxx`
	`        Subnet Mask . . . . . . . . . . . : 255.255.255.0`
	`        Default Gateway . . . . . . . . . : xxx.xxx.xxx.xxx`

6.4.8 Extract lines Filter

This filter finds rows in the results of an operation component or in filter output that meet specified conditions and extracts them.



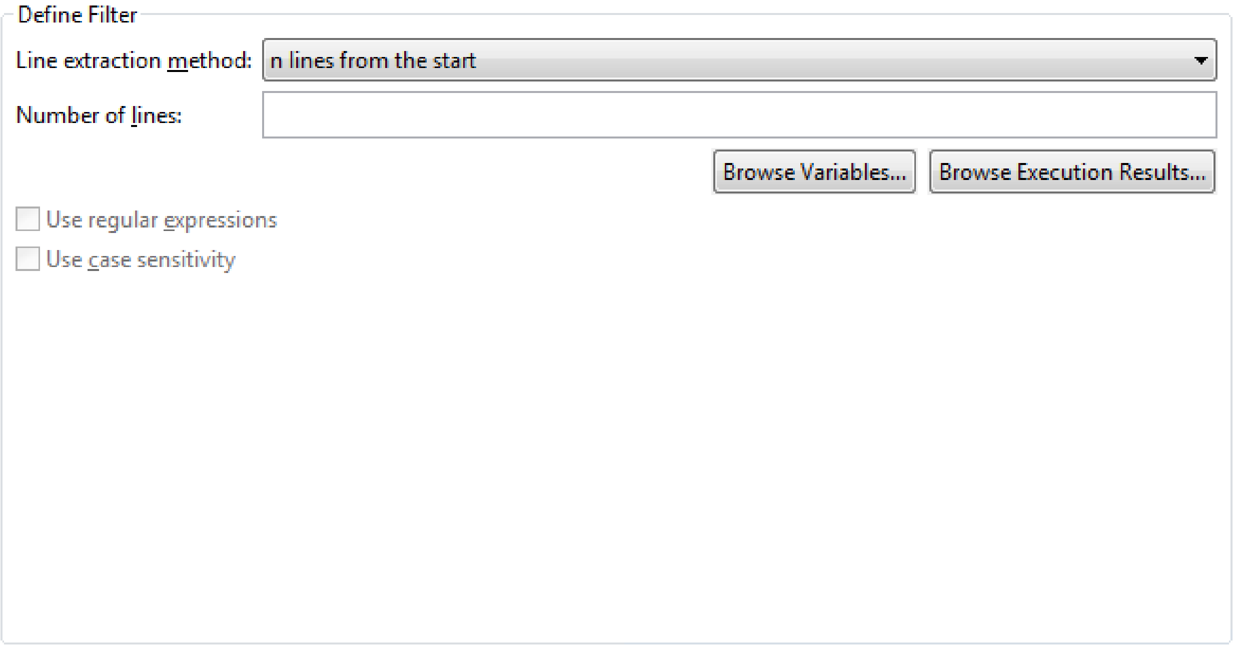
Define Filter

Line extraction method: All lines containing the target string

Target string:

□ Use regular expressions

□ Use case sensitivity



Define Filter

Line extraction method: n lines from the start

Number of lines:

□ Use regular expressions

□ Use case sensitivity

Line extraction method

Select the conditions for extracting the row from the following:

- **All lines starting with the target string**
- **All lines ending with the target string**
- **All lines containing the target string** (Default)
- **All lines that do not contain the target string**
- **Line to be specified**
- **First line**
- **Last line**
- **n lines from the start**

- **n lines from the end**

Target string

Enter the string used to extract rows. The input can be up to 512 characters long. To embed the variable (UDA) value and execution results of the previous operation component in the **Target string** field when the filter is applied, follow the steps below:

- To embed the value, specify "@{uda:UDA name}") or click the **Browse variables** button and select a variable from the **Browse variables** dialog box.

  The type of variable (UDA) available is STRING.

- To embed the execution results of the previous operation component, specify "@{:node name:execution results}" or click the **Browse execution results** button and select an execution result from the **Browse execution results** dialog box.

The filter will end in an error if the field is empty.

Line number/Number of lines

Enter row numbers for the rows to extract or enter a number of rows between 1 and 2147483647.To embed the variable (UDA) value and execution results of the previous operation component in the **Line number/Number of lines** field when the filter is applied, follow the steps below:

- To embed the value, specify "@{uda:UDA name}") or click the **Browse variables** button and select a variable from the **Browse variables** dialog box.

  The types of variables available are STRING and INTEGER. An error will occur if a non-numeric value is specified for a STRING type variable (UDA).

- To embed the execution results of the previous operation component, specify "@{:node name:execution results}" or click the **Browse execution results** button and select an execution result from the **Browse execution results** dialog box.

Use regular expressions

Select when regular expressions are to be used in the **Target string** field.

The default is off.

Use this field only if strings to be specified are in **Line extraction method**. (This is grayed out if unavailable)

Use case sensitivity

Select to distinguish between upper and lower case when searching for the string. The default is off.

Use this field only if strings to be specified are in **Line extraction method**. (This is grayed out if unavailable)

Note

Precautions when Testing Filters

- If "@{uda: variable name}" is specified in the **Line number/Number of lines** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or 1 if no value is assigned.

- If "@{:node name:execution results}"is specified in the **Line number/Number of lines** field:

  The test will use the value assigned in the**Set test variables** dialog box (*1), or 1 if no value is assigned.

- If "@{uda: variable name}"is specified in the **Target string** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or the string "@{uda:variable name}" if not value is assigned.

- If "@{:node name:execution results}" is specified in the **Target string** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or the string "@{:node name:execution results}" if no value is assigned.

- If either "@{uda:variable name}" or "@{:node name:execution results}" is specified in the **Target string** field, **Use regular expressions** is ignored.

- When the filter is applied on the Management Server, these variables (UDA) or execution results are replaced with the stored values.

*1: Refer to "6.3.2 Testing Filters" for information on the **Set test variables** dialog box.

Example 1:

Input:	`AAAA.AAAA[linefeed]`
	`AAAA$BBBB[linefeed]`
	`AAAA.CCCC[linefeed]`
	`AAAA@DDDD[linefeed]`

Extraction method for rows	All rows starting with the specified string
Specified string:	AAAA.
Use regular expressions:	Off
Match case:	Off

Output:	`AAAA.AAAA[linefeed]`
	`AAAA.CCCC[linefeed]`

Example 2:

Input:	`AAaa.AAAA[linefeed]`
	`AAAA$BBBB[linefeed]`
	`AAAA.CCCC[linefeed]`
	`AAaa@DDDD[linefeed]`

Extraction method for rows	All rows starting with the specified string
Specified string:	AAaa.
Use regular expressions:	On * When regular expressions are used, dots are considered to be a character in the target string.
Match case:	On

Output:	`AAaa.AAAA[linefeed]`
	`AAaa@DDDD[linefeed]`

Example 3:

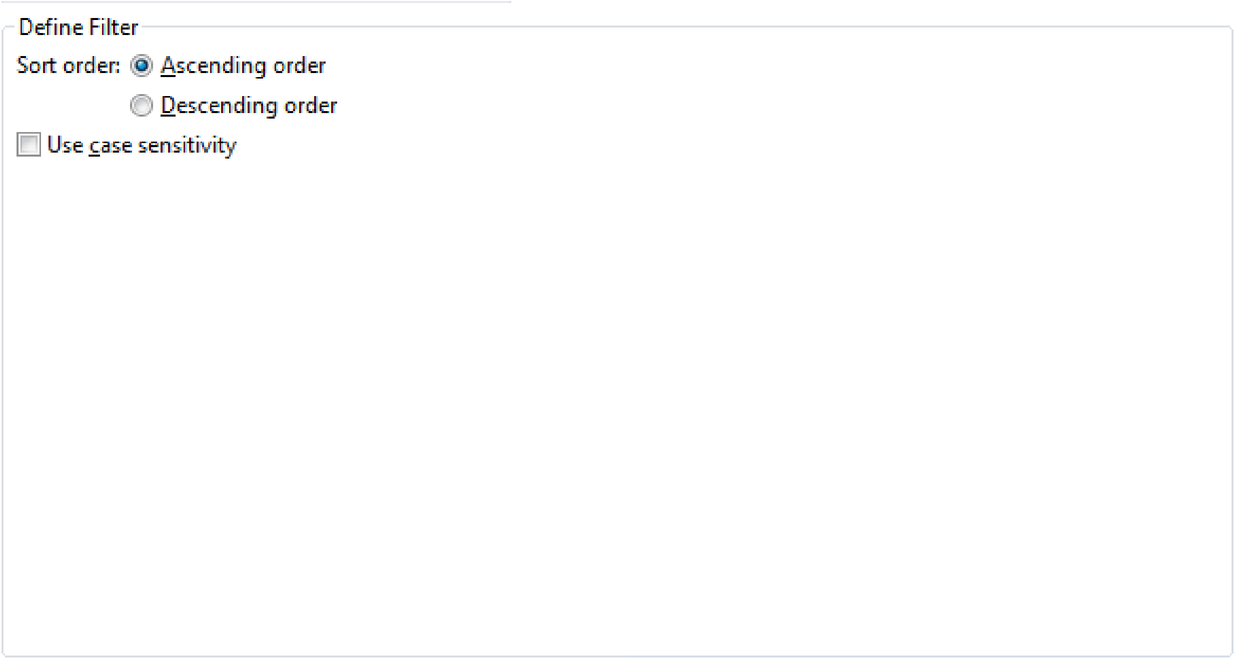
Input:	`AAAA[linefeed]`
	`BBBB[linefeed]`
	`CCCC[linefeed]`
	`DDDD[linefeed]`
	`EEEE[linefeed]`

Extraction method for rows	n rows from the first row
Row Count:	3

Output:	`AAAA[linefeed]`
	`BBBB[linefeed]`
	`CCCC[linefeed]`

6.4.9 Sort Filter

This filter sorts the results of an operation component or in filter output row by row.



Define Filter

Sort order: ● Ascending order

○ Descending order

□ Use case sensitivity

Ascending order

Select to sort in ascending order. (Default)

Descending order

Select to sort in descending order.

Use case sensitivity

Select to distinguish between upper and lower case when sorting.

The default is off.

Example 1:

Input:	`ID0002[linefeed]`
	`ID0001[linefeed]`
	`ID0004[linefeed]`
	`ID0003[linefeed]`

Ascending:	On
Descending:	Off
Case sensitive	Off

Output:	`ID0001[linefeed]`
	`ID0002[linefeed]`
	`ID0003[linefeed]`
	`ID0004[linefeed]`

Example 2:

Input:	`ID0002[linefeed]`
	`ID0001[linefeed]`
	`ID0004[linefeed]`
	`ID0003[linefeed]`

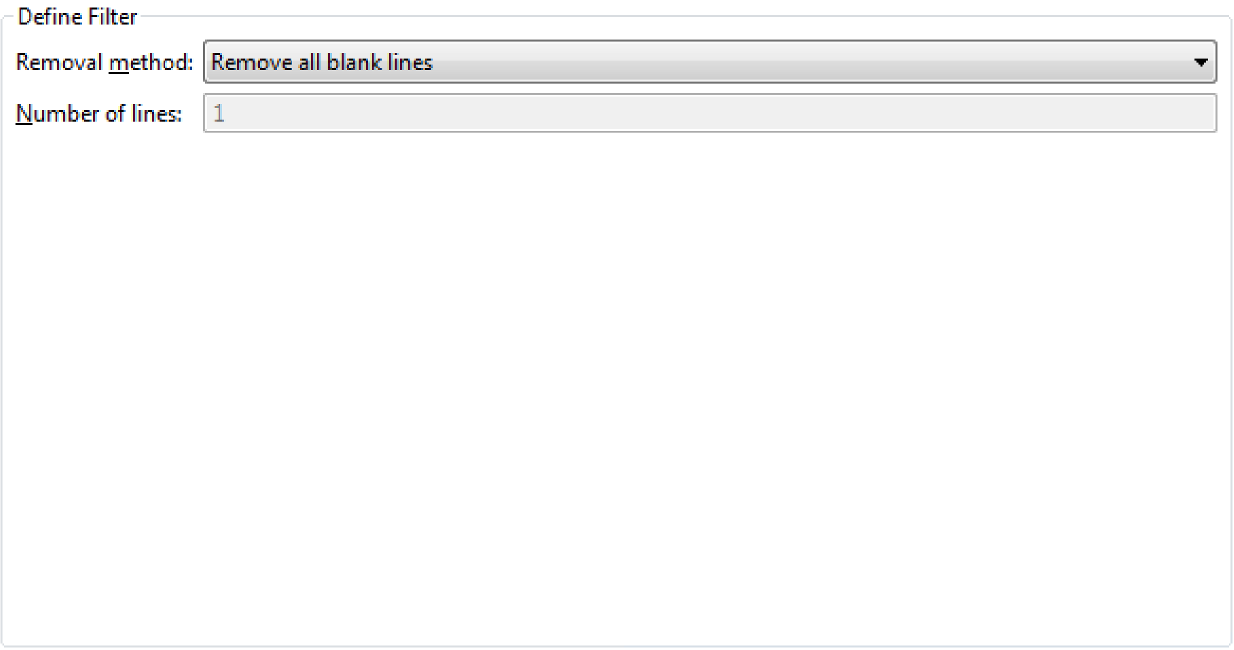
Ascending:	Off
Descending:	On
Case sensitive:	Off

Output:	ID0004[linefeed] ID0003[linefeed] ID0002[linefeed] ID0001[linefeed]

6.4.10 Remove lines Filter

This filter deletes rows in the results of an operation component or in filter output that are before or after a specified row.

All empty rows can also be deleted. It is also possible to specify a number of rows from the start or end to leave and then delete all other rows.



Define Filter

Removal method: Remove all blank lines

Number of lines: 1

Removal method

Select the way to delete from the following:

- Delete all empty rows (Default)
- Delete n rows from the first row
- Delete n rows from the last row

Number of lines

Enter row numbers (from 1). The range is from 1 to 2147483647.

This field is unavailable (grayed out) when **Remove all blank lines** is selected for **Removal method**.

The filter will end in an error if the **Number of lines** field is empty.

Example 1:

Input:	AAAA[linefeed] BBBB[linefeed] [empty row] CCCC[linefeed] DDDD[linefeed]
Deleting:	Delete all empty rows
Output:	AAAA[linefeed] BBBB[linefeed]

	`CCCC[linefeed]`
	`DDDD[linefeed]`

Example 2:

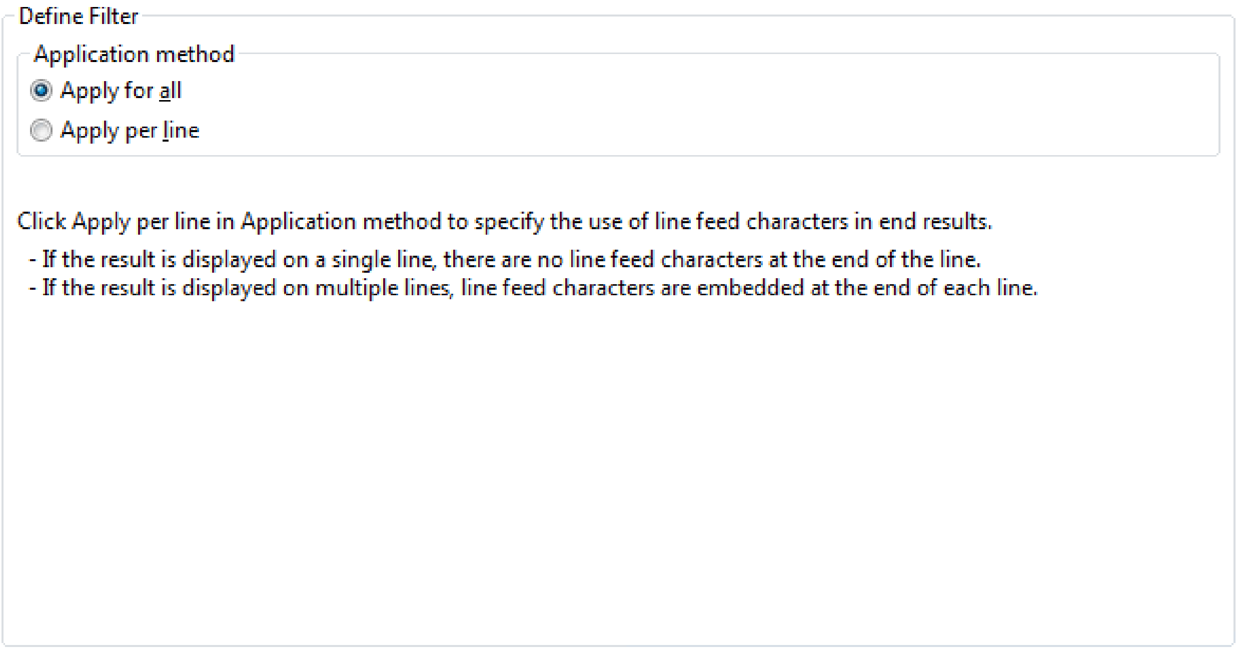
Input:	`AAAA[linefeed]`
	`BBBB[linefeed]`
	`CCCC[linefeed]`
	`DDDD[linefeed]`
	`EEEE[linefeed]`

Deleting:	Delete n rows from the first row
Row Count:	2

Output:	`CCCC[linefeed]`
	`DDDD[linefeed]`
	`EEEE[linefeed]`

6.4.11 Remove space Filter

This filter deletes white space (half-width spaces, full-width spaces, tabs, and linefeeds (CRLF/CR/LF)) found at the beginning or end of the results of an operation component or in filter output.



Define Filter

Application method

● Apply for all

○ Apply per line

Click Apply per line in Application method to specify the use of line feed characters in end results.

- If the result is displayed on a single line, there are no line feed characters at the end of the line.
- If the result is displayed on multiple lines, line feed characters are embedded at the end of each line.

Applications

Select **Apply for all** to apply the filter to the operation component results or the entire filter output. White space (half-width spaces, full-width spaces, tabs, and linefeeds) at the beginning and end of the input data is deleted.

Select **Apply per line** to apply the filter row-by-row. White space (half-width spaces, full-width spaces, and tabs) at the beginning and end of each row in the input data is deleted. The linefeeds at the ends of the rows are not deleted.

The default is **Apply for all**.

Example 1:

Input:	`[half-width space][full-width space]abc[tab][linefeed]`
	`[half-width space][full-width space]def[tab][linefeed]`

Applications:	Applied overall

Output:	`abc[tab][linefeed]`
	`[half-width space][full-width space]def`

Example 2:

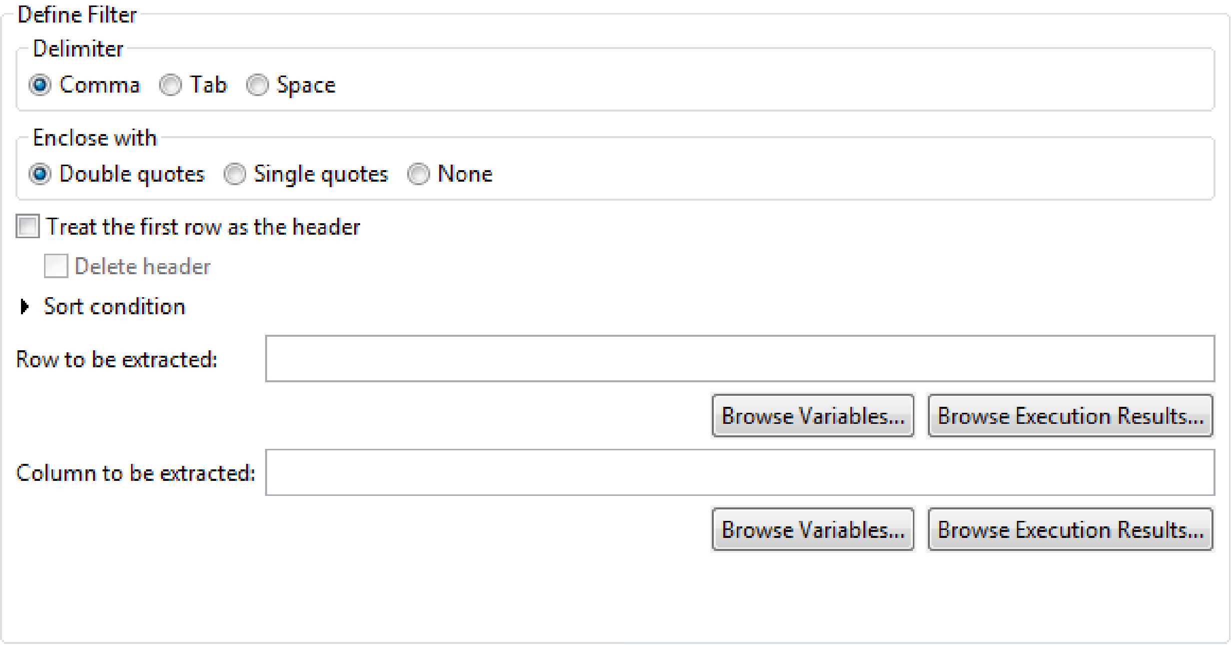
Input:	`[half-width space][full-width space]abc[tab][linefeed]`
	`[half-width space][full-width space]def[tab][linefeed]`

Applications:	Applied row-by-row

Output:	`abc[linefeed]`
	`def[linefeed]`

6.4.12 Extract CSV data Filter

This filter can perform table operations on the results of an operation component or in filter output, such as sorting columns, and selecting columns, rows and blocks.



Define Filter

Delimiter

● Comma ○ Tab ○ Space

Enclose with

● Double quotes ○ Single quotes ○ None

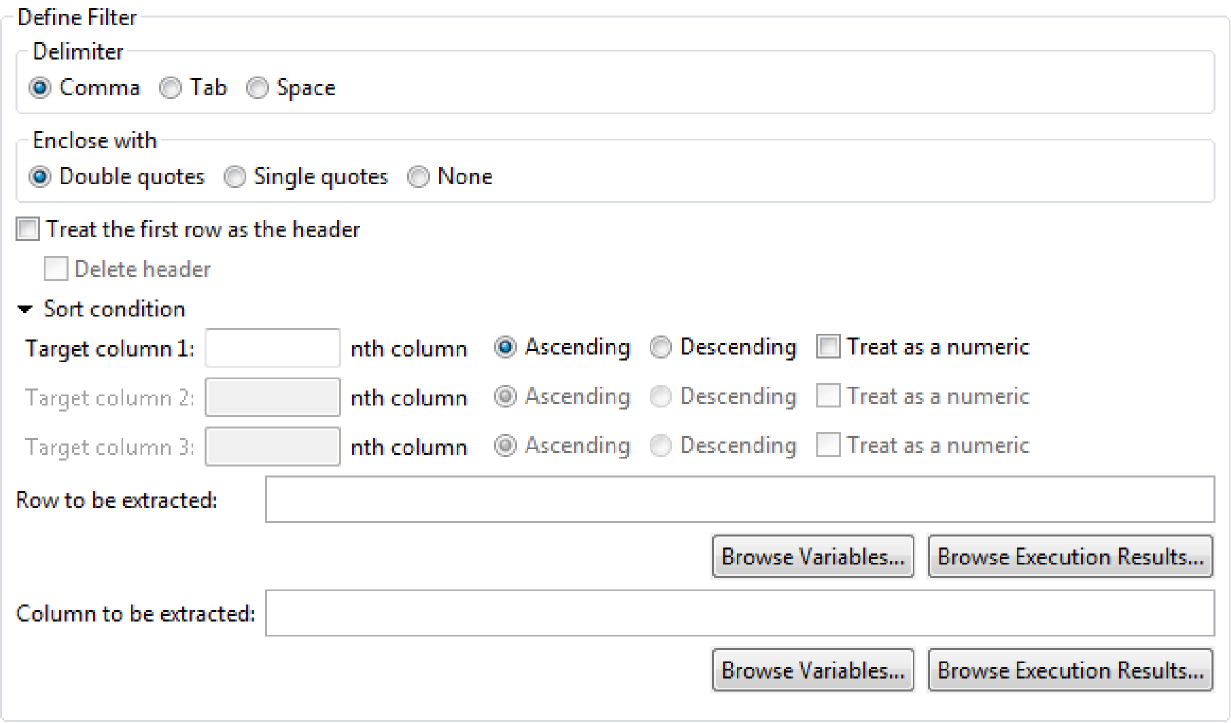
□ Treat the first row as the header

□ Delete header

▸ Sort condition

Row to be extracted:

Column to be extracted:



Define Filter

Delimiter

● Comma ○ Tab ○ Space

Enclose with

● Double quotes ○ Single quotes ○ None

□ Treat the first row as the header

□ Delete header

Sort condition

Target column 1: nth column ● Ascending ○ Descending □ Treat as a numeric

Target column 2: nth column ● Ascending ○ Descending □ Treat as a numeric

Target column 3: nth column ● Ascending ○ Descending □ Treat as a numeric

Row to be extracted:

Column to be extracted:

Delimiter

Select the character to split the input data into values (columns).

- Comma (Default)
- Tab
- Spaces

Enclose with

Select characters to enclose values:

- Double quotes (Default)
- Single quote

Treat the first row as the header

Select to treat the first row as the header. Select **Treat the first row as the header** to exclude the first row from sorting. The default is off.

Delete header

Select to delete the header row. The default is off.

This is available only when **Treat the first row as the header** has been selected.

Sort condition

Click **Sort condition** to expand (display) the settings items for the sort conditions.

Target column

To sort by column, enter up to three column numbers (starting from 1) in the desired priority order in the **Target column** field. Leave the field empty is no sort should occur.

All fields are empty by default. Only the left field can be entered initially. The center and right fields are unavailable (grayed out). Once a value is entered in Target column 1, Target column 2 can be entered. Then, once a value is entered in Target column 2, Target column 3 can be entered.

Ascending Order

Select to sort columns in ascending order. (Default)

Descending Order

Select to sort columns in descending order.

Treat as numbers

Select to sort as if the values were numbers. An error will occur if the value has not been provided (is empty) or contains non-numeric characters. The default is off.

Row to be extracted

The start and the end rows to be extracted can be entered using the formats listed in the table below. The default is empty. To embed the variable (UDA) value and execution results of the previous operation component in the **Row to be extracted** field when the filter is applied, follow the steps below:

- To embed the value: specify "@{uda: variable name}" or click the **Browse variables** button and select a variable from the **Browse variables** dialog box.

  The only variables that may be specified are of the STRING or INTEGER type.

- To embed the execution results of the previous operation component, specify "@{:node name:execution results}"or click the **Browse execution results** button and select an execution result from the **Browse execution results** dialog box.

Format (start row - end row)	Selected row
None	All rows
1-10	From row 1 to row 10
-10	From row 1 to row 10
1-	From row 1 to the last row

- The filter will end in an error if there is only a hyphen (-) or if there are 2 or more hyphens.
- An error will occur if a character other than hyphen (-) or a number (0-9) is entered.
- An error occurs on executing the filter if multiple "@{uda:variable name}" or "@{:node name:execution results}" are specified.

Column to be extracted

Enter the column to be extracted in the following format. The default is empty. To embed the variable (UDA) value and execution results of the previous operation component in the **Column to be extracted** field when the filter is applied, follow the steps below:

- To embed the value, specify "@{uda: variable name}" or click the **Browse variables** button and select a variable from the **Browse variables** dialog box.

  The only variables that may be specified are of the STRING or INTEGER type.

- To embed the execution results of the previous operation component, specify "@{:node name:execution results}" or click the **Browse execution results** button and select an execution result from the **Browse execution results** dialog box.

Format	Selected column
None	All columns
1,3	1st column (far left) and 3rd column
1-3	1st column (far left) to 3rd column
2-	2nd column to last column (far right)
-10	1st column (far left) to 10th column
1-3,6-8	1st column (far left) to 3rd column and 6th column to 8th column

- The filter will end in an error if there is only a hyphen (-) or if there are consecutive hyphens.

- The filter will end in an error if characters other than hyphens (-), commas (,), or numerals (0-9) are included.
- The filter will end in an error if there is only a comma (,) or if there are consecutive commas.
- If the same column is specified more than once, only the first time is valid. Therefore "1-5,2" is the same as specifying "1,2,3,4,5".
- An error occurs on executing the filter if the specified column number does not exist.
- An error occurs on executing the filter if multiple "@{uda:variable name}" or "@{:node name:execution results}" are specified.

Note

Precautions when Testing Filters

- If "@{uda: variable name}"is specified in the **Column to be extracted** field :

  The test will use the value assigned in the**Set test variables**dialog box (*1) or an empty string if no value is assigned (and as a result all rows will be extracted).

- If "@{Node name Execution results}"is specified in the **Column to be extracted** field:

  The test will use the value assigned in the **Set test variables**dialog box (*1) or an empty string if no value is assigned (and as a result all rows will be extracted).

- If "@{uda: variable name}"is specified in the **Column to be extracted** field :

  The test will use the value assigned in the **Set test variables** dialog box (*1) or an empty string if no value is assigned (and as a result all rows will be extracted).

- If "@{Node name Execution results}"is specified in the **Column to be extracted** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1) or an empty string if no value is assigned (and as a result all rows will be extracted).

- When the filter is applied on the Management Server, these variables (UDA) or execution results are replaced with the stored values.

*1: Refer to "6.3.2 Testing Filters" for information on the**Set test variables** dialog box.

Example 1:

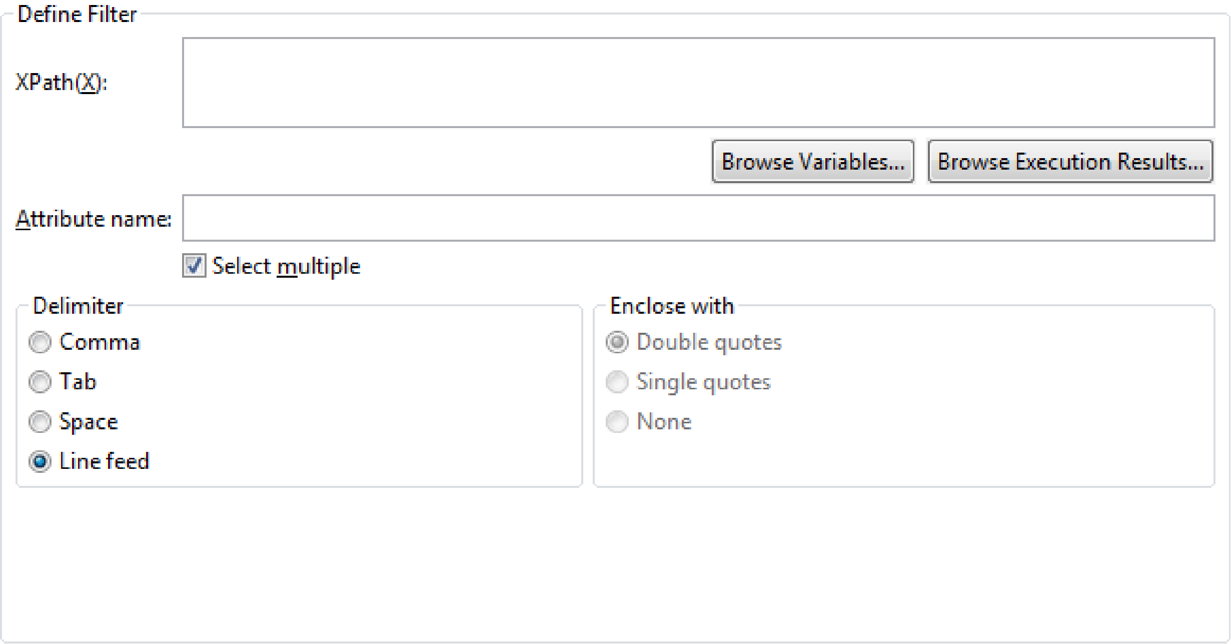
Input:	`"Server name","Administrator","Contact"[linefeed]`
	`"Server-C","User2","1111-2222"[linefeed]`
	`"Server-A","User1","1111-1111"[linefeed]`
	`"Server-B","User3","2222-1111"[linefeed]`
	`"Server-A","User2","1111-2222"[linefeed]`

Delimiter:	Comma
Enclosing character:	Double quotes
Treat first row as header:	On
Delete header:	On
Sorting conditions (target column 1):	First column, ascending
Sorting conditions (target column 2):	Second column, descending
Row to be extracted:	Not input
Column to be extracted:	Not input

Output:	`Server-A,User2,1111-2222[linefeed]`
	`Server-A,User1,1111-1111[linefeed]`
	`Server-B,User3,2222-1111[linefeed]`
	`Server-C,User2,1111-2222[linefeed]`

6.4.13 Extract XML attribute Filter

This filter extracts attribute values from the results of an operation component or in filter output that are in XML format.



[XPath]

Enter the path of the element that holds the attribute. The input can be up to 4096 characters long. To embed the variable (UDA) value and execution results of the previous operation component in the **XPath** field when the filter is applied, follow the steps below:

- To embed the value, specify "@{uda: variable name}" or click the **Browse variables** button and select a variable from the **Browse variables** dialog box.

  The types of variables (UDA) available are STRING/INTEGER.

- To embed the execution results of the previous operation component, specify "@{:node name:execution results}" or click the **Browse execution results** button and select an execution result from the **Browse execution results** dialog box.

The filter will end in an error if the field is empty.

Attribute name

Enter the name of the attribute to be used to extract the value. The input can be up to 512 characters long.

The filter will end in an error if the field is empty.

Select multiple

Select to extract all results when multiples exist. Only the first result is extracted if this checkbox is cleared.

The default is on.

Delimiter

Select the delimiter for extracted results (values). This is not used if only one item is extracted.

- Comma
- Spaces
- Tab
- Linefeed [LF] (Default)

Enclose with

Select the character for enclosing extracted results (values):

- Double quotes (Default)
- Single quote
- None

If "**Line feed**" is selected for **Delimiter**, **Enclose with** is grayed out and ignored.

Note

Precautions when Testing Filters

- If "@{uda:variable name}" is specified in the **XPath** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or the string "@{uda:variable name}" if no value is assigned.

- If "@{:node name:execution results}" is specified in the **XPath** field:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or the string "@{:node name:execution results}" if no value is assigned.

- When the filter is executed on the Management Server, these variables (UDA) or execution results are replaced with the stored values.

*1: Refer to "6.3.2 Testing Filters" for information on the **Set test variables** dialog box.

Example 1:

Input:	`<systems>`
	`  <system name="system A" mail="user1@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.1.10"/>`
	`      <server ipaddress="192.168.1.20"/>`
	`    </servers>`
	`  </system>`
	`  <system name="system B" mail="user2@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.2.10"/>`
	`      <server ipaddress="192.168.2.20"/>`
	`      <server ipaddress="192.168.2.30"/>`
	`    </servers>`
	`  </system>`
	`</systems>`

XPath:	/systems/system [@name='system A'] /servers/server
Property Name:	ipaddress
Select multiple:	On
Delimiter:	Linefeed character
Enclosed alphanumerics:	-

Output:	`192.168.1.10[linefeed]`
	`192.168.1.20[linefeed]`

Example 2:

Input:	`<systems>`
	`  <system name="system A" mail="user1@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.1.10"/>`
	`      <server ipaddress="192.168.1.20"/>`
	`    </servers>`
	`  </system>`
	`  <system name="system B" mail="user2@example.com">`

	`    <servers>`
	`      <server ipaddress="192.168.2.10"/>`
	`      <server ipaddress="192.168.2.20"/>`
	`      <server ipaddress="192.168.2.30"/>`
	`    </servers>`
	`  </system>`
	`</systems>`

XPath:	/systems/system [@name='system A'] /servers/server
Property Name:	ipaddress
Select multiple:	On
Delimiter:	Comma
Enclosed alphanumerics:	Double quotes
Output:	`"192.168.1.10","192.168.1.20"`

Example 3:

Input:	`<systems>`
	`  <system name="system A" mail="user1@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.1.10"/>`
	`      <server ipaddress="192.168.1.20"/>`
	`    </servers>`
	`  </system>`
	`  <system name="system B" mail="user2@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.2.10"/>`
	`      <server ipaddress="192.168.2.20"/>`
	`      <server ipaddress="192.168.2.30"/>`
	`    </servers>`
	`  </system>`
	`</systems>`

XPath:	/systems/system [@name='system A'] /servers/server
Property Name:	ipaddress
Select multiple:	Off
Delimiter:	Linefeed character
Enclosed alphanumerics:	-
Output:	`192.168.1.10[linefeed]`

[xample 4: If "System A" is set in variables (SystemName)

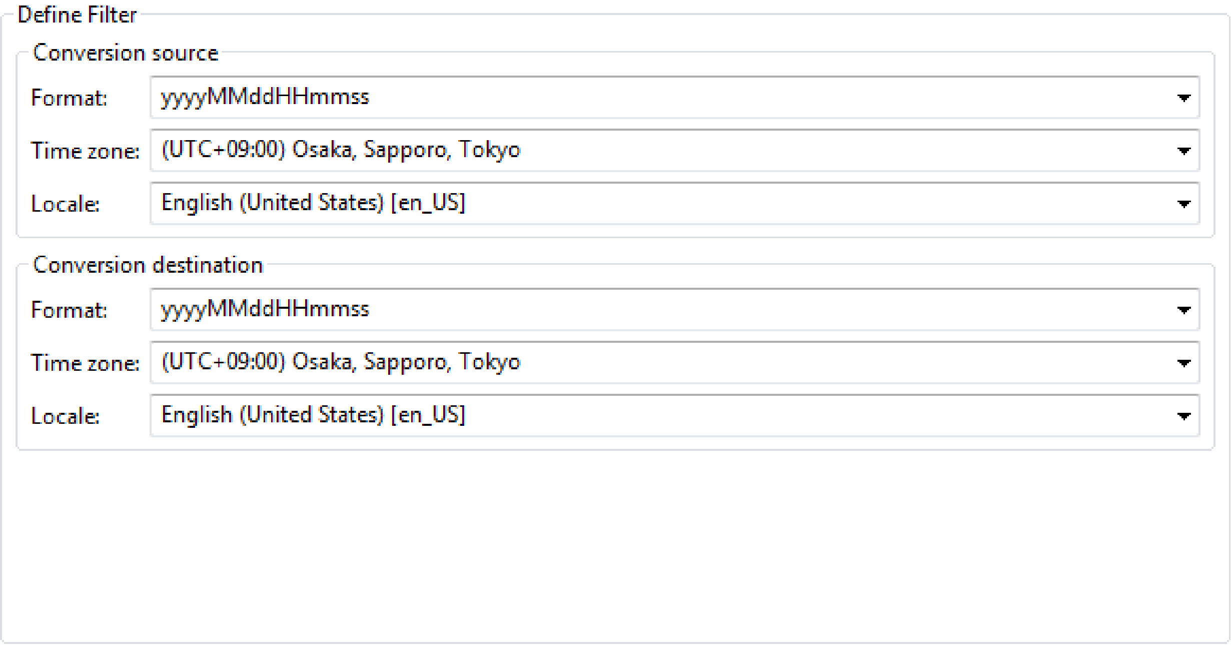
Input:	`<systems>`
	`  <system name="System A" mail="user1@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.1.10"/>`
	`      <server ipaddress="192.168.1.20"/>`
	`    </servers>`
	`  </system>`
	`  <system name="System B" mail="user2@example.com">`
	`    <servers>`
	`      <server ipaddress="192.168.2.10"/>`
	`      <server ipaddress="192.168.2.20"/>`
	`      <server ipaddress="192.168.2.30"/>`
	`    </servers>`
	`  </system>`
	`</systems>`

XPath:	/systems/system[@name='@{uda:SystemName}']/servers/server
Property Name:	ipaddress
Select multiple:	On
Delimiter:	Line feed
Enclosed alphanumerics:	-

Output:	`192.168.1.10[linefeed]`
	`192.168.1.20[linefeed]`

6.4.14 Convert date format Filter

This filter converts the date data in the results of an operation component or in filter output to different date formats.



Define Filter

Conversion source

Format: yyyyMMddHHmmss

Time zone: (UTC+09:00) Osaka, Sapporo, Tokyo

Locale: English (United States) [en_US]

Conversion destination

Format: yyyyMMddHHmmss

Time zone: (UTC+09:00) Osaka, Sapporo, Tokyo

Locale: English (United States) [en_US]

An error occurs on executing the filter if the input data does not match the source format (for example, when there are blank spaces at the end of the input data or when there is a linefeed code).

Conversion source

Format

Enter or select from the combo box the formatting string for the date to be received as input. The input can be up to 512 characters long. The one-byte alphabet included in format will be evaluated as date/time format pattern. The arbitrary one-byte alphabet which is not consistent with date/time format pattern can not be included in it.

The filter will end in an error if the **Format** field is empty.

- yyyyMMddHHmmss
- yyyy/MM/dd HH:mm:ss
- yyyy/MM/dd
- yyyy-MM-dd
- yyyy-MM-ddZZ
- HH:mm:ss
- HH:mm:ssZZ

- EEE, dd MMM yyyy HH:mm:ss Z

Information

- y indicates the year. (Example: "2011"; "11")
- M indicates the month. (Example: "09")
- d indicates a day in a month. Example: "12")
- H indicates an hour in a day (0 to 23). (Example: "08")
- m indicates the minutes. (Example: "40")
- s indicates the seconds. (Example: "59")
- Z indicates the timezone. (Example: "+0900")
- E indicates the day of the week. (Example: "Monday")

Example

Here are examples of date and time format patterns.

Date and time patterns (example)	How the string appears
yyyyMMddHHmmss	20110912084000
yyyy/MM/dd HH:mm:ss	2011/09/12 08:40:00
EEE, dd MMM yyyy HH:mm:ss Z	Mon, 12 9 2011 08:40:00 +0900
yyyyMMddHHmmssZ	20110912084000+0900

Refer to the "**Java 2 Platform API Specification**" for information on patterns not mentioned above.

Time Zone

Select or enter the time zone. The default is the timezone of the system.

The time zone can also be specified in the "GMT+09:00" format. It will be considered that GMT has been specified in case there is an error with the format of the specified time zone.

- (UTC-10:00) Hawaii
- (UTC-09:00) Alaska
- (UTC-08:00) Pacific standard time (United States and Canada)
- (UTC-07:00) Mountain standard time (United States and Canada)
- (UTC-06:00) Central time (United States and Canada)
- (UTC-05:00) Eastern Standard time (United States and Canada)
- (UTC) Universal Time Coordinated
- (UTC+09:00) Osaka, Sapporo, Tokyo

Locale

Select the locale to use for date conversion from the combo box. This can also be entered directly, using the two-letter language code and two-letter country code separated by an underscore. The default is the locale of the system. If there is an error with the format of the specified locale, it will be considered that the system locale has been specified.

- Japanese (Japan) [ja_JP]
- English (Ireland) [en_IE]

- English (United States) [en_US]
- English (United Kingdom) [en _ GB]
- English (India) [en_IN]
- English (Australia) [en_AU]
- English (Canada) [en_CA]
- English (New Zealand) [en_NZ]
- English (South Africa) [en_ZA]

Information

An example of the locale format is "de_DE" indicative of the German (Germany) locale.

Conversion destination

Format

Enter the formatting string for the data after conversion. The input can be up to 512 characters long. Refer to **Format** for **Conversion source** for a list of available formats.

The filter will end in an error if the **Format** field is empty. The one-byte alphabet included in format will be evaluated as date/time format pattern. The arbitrary one-byte alphabet which is not consistent with date/time format pattern can not be included in it.

Time Zone

Select or enter the timezone. The default is the timezone of the system. Refer to **Time zone** for **Conversion source** for a list of available time zones.

Locale

Select the locale to use for date conversion from the combo box. This can also be entered directly, using the two-letter language code and two-letter country code separated by an underscore. The default is the locale of the system. Refer to **Locale** for **Conversion source** for a list of available locales.

Example 1:

Input:	20111013223400
Format (source):	yyyyMMddHHmmss
Timezone (source):	(UTC+09:00) Osaka, Sapporo, Tokyo
Locale (source):	Japanese (Japan) [ja_JP]
Format (target):	yyyy/MM/dd HH:mm:ss
Timezone (target):	(UTC+09:00) Osaka, Sapporo, Tokyo
Locale (target):	Japanese (Japan) [ja_JP]
Output:	2011/10/13 22:34:00

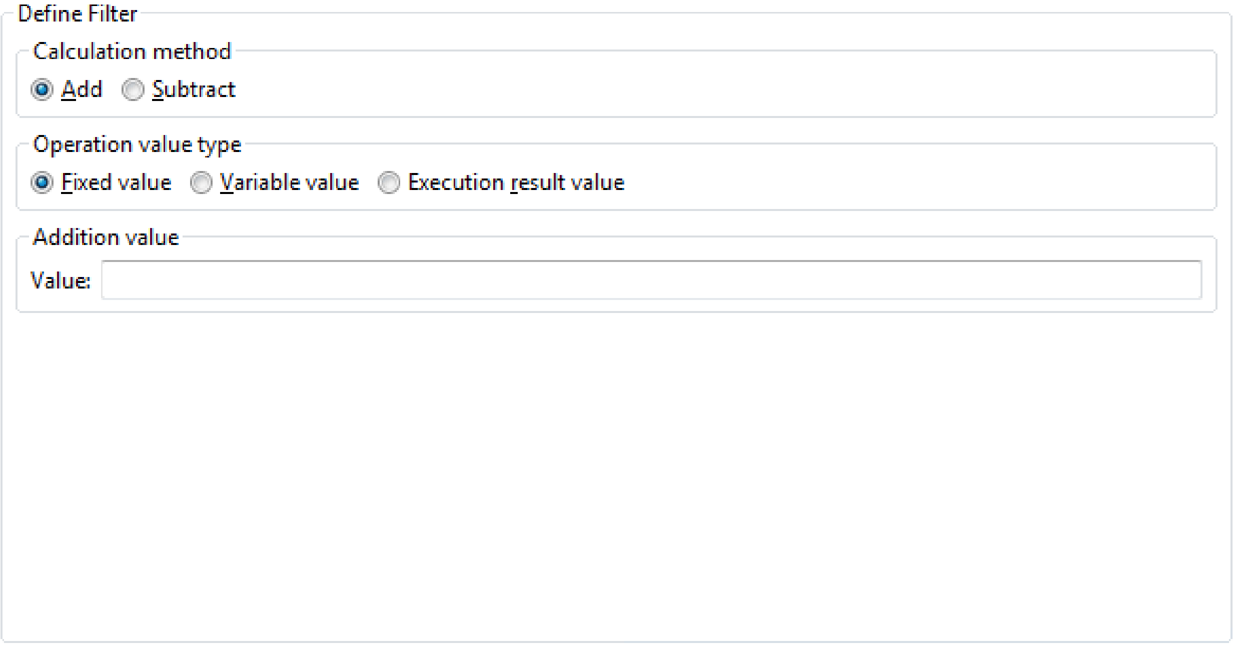
Example 2:

Input:	20111013223400
Format (source):	yyyyMMddHHmmss
Timezone (source):	(UTC+09:00) Osaka, Sapporo, Tokyo
Locale (source):	Japanese (Japan) [ja_JP]
Format (target):	EEE, dd MMM yyyy HH:mm:ss Z
Timezone (target):	(UTC+09:00) Osaka, Sapporo, Tokyo
Locale (target):	Japanese (Japan) [ja_JP]

Output:	Thu, 13 10 2011 22:34:00 +0900

6.4.15 Calculate Filter

This filter takes values in the results of an operation component or in filter output (fixed values, variable values, or execution result values (integers)) and adds or subtracts them.



Define Filter

Calculation method

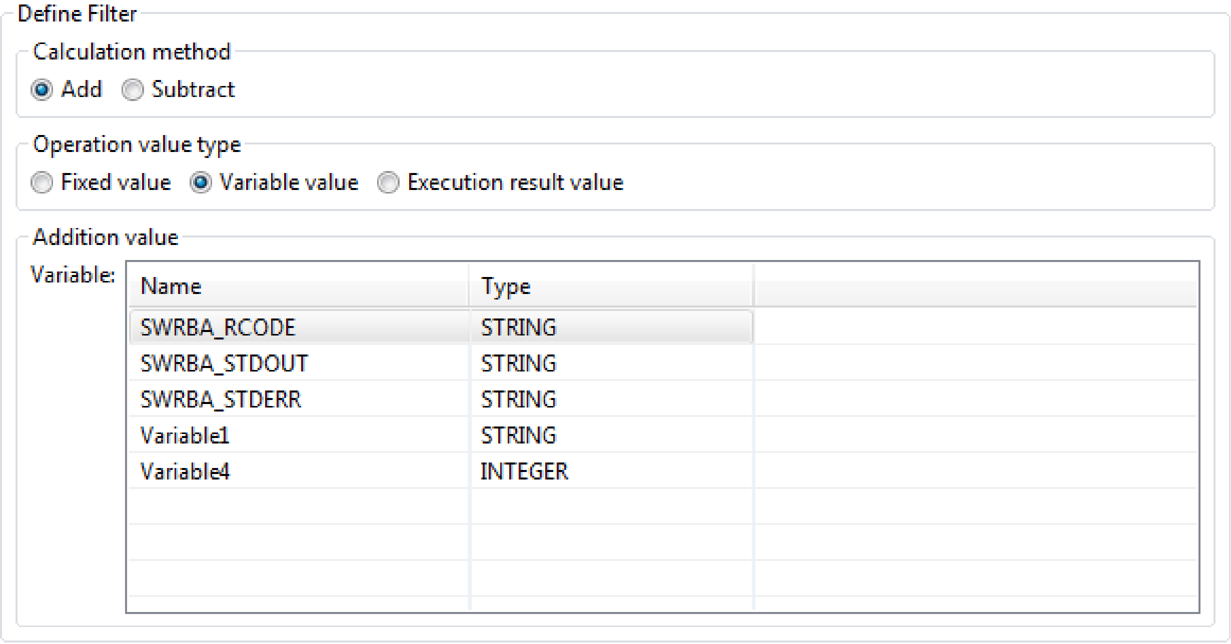
● Add ○ Subtract

Operation value type

● Fixed value ○ Variable value ○ Execution result value

Addition value

Value:



Define Filter

Calculation method

● Add ○ Subtract

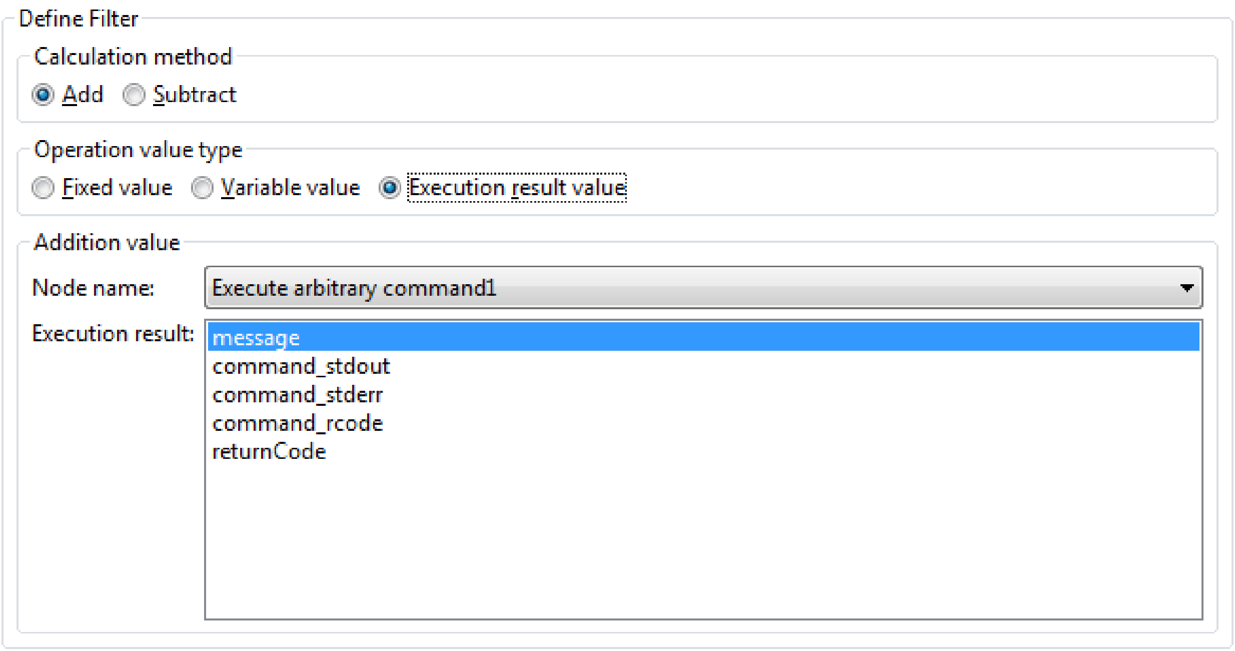
Operation value type

○ Fixed value ● Variable value ○ Execution result value

Addition value

Variable:

Name	Type
SWRBA_RCODE	STRING
SWRBA_STDOUT	STRING
SWRBA_STDERR	STRING
Variable1	STRING
Variable4	INTEGER



Define Filter

Calculation method

● Add ○ Subtract

Operation value type

○ Fixed value ○ Variable value ● Execution result value

Addition value

Node name: Execute arbitrary command1

Execution result:
- message
- command_stdout
- command_stderr
- command_rcode
- returnCode

Calculation method

Select one of the following operations:

- Add (Default)
- Subtract

Operation value type

Select the type of value for addition or subtraction:

- Fixed value (Default)
- Variable value
- Execution results value

Addition value/Subtraction value

The title of the group changes from **Calculation method** to either **Addition value** or **Subtraction value**.

Fixed value

Enter the value (number) to be added or subtracted. The range is from 0 to 2147483647.

An error occurs on executing the filter if this field is empty.

Variable value

Select the variable that holds the value for addition or subtraction. The types of variables displayed in the list are STRING and INTEGER only.

Execution results value

Select the execution result that holds the value for addition or subtraction.

A list of nodes (operation component nodes) that can be selected is displayed in the **Node Name** combo box. When you select the node name, a list of execution results for the node is displayed in **Execution results**.

Note

- Operations can only be performed on integers.

- The range for operations is between -2147483648 and 2147483647. An overflow occurs if 1 is added to 2147483647 and the result of the operation will be -2147483648.
- When a STRING-type variable (UDA) is specified, the variable value may not include characters other than numbers. The filter will end in an error if characters other than numbers are included.
- The execution values may not include characters other than numbers. The filter will end in an error if characters other than numbers are included.

Precautions when Testing Filters

- If **Operation value type** is set to **Variable value**:

  The test will use the value assigned in the **Set test variables** dialog box (*1), or 0 if no value is assigned.

- If **Operation value type** is set to **Execution result value**:

  The test will use the value assigned in the**Set test variables** dialog box (*1), or 0 if no value is assigned.

- When the filter is applied on the Management Server, these variables (UDA) or execution results are replaced with the stored values.

*1: Refer to "6.3.2 Testing Filters" for information on the **Set test variables** dialog box.

Example 1:

Input:	1
Types of operation:	Addition
Types of operation value:	Fixed value
Added value:	1
Output:	2

Example 2:

Input:	1
Types of operation:	Subtraction
Types of operation value:	Fixed value
Subtraction value:	1
Output:	0

Example 3:

Input:	0
Types of operation:	Subtraction
Types of operation value:	Fixed value
Subtraction value:	1
Output:	-1

Example 4: When 1 has been set in the variable (AddValue)

Input:	1
Types of operation:	Addition
Types of operation value:	Variable value
Variable name:	AddValue

Output:	2

Example 5: When 1 has been set in the execution result (returnCode)

Input:	1
Types of operation:	Addition
Types of operation value:	Execution results value
Node name:	Execute xxxx
Execution result	returnCode
Output:	2

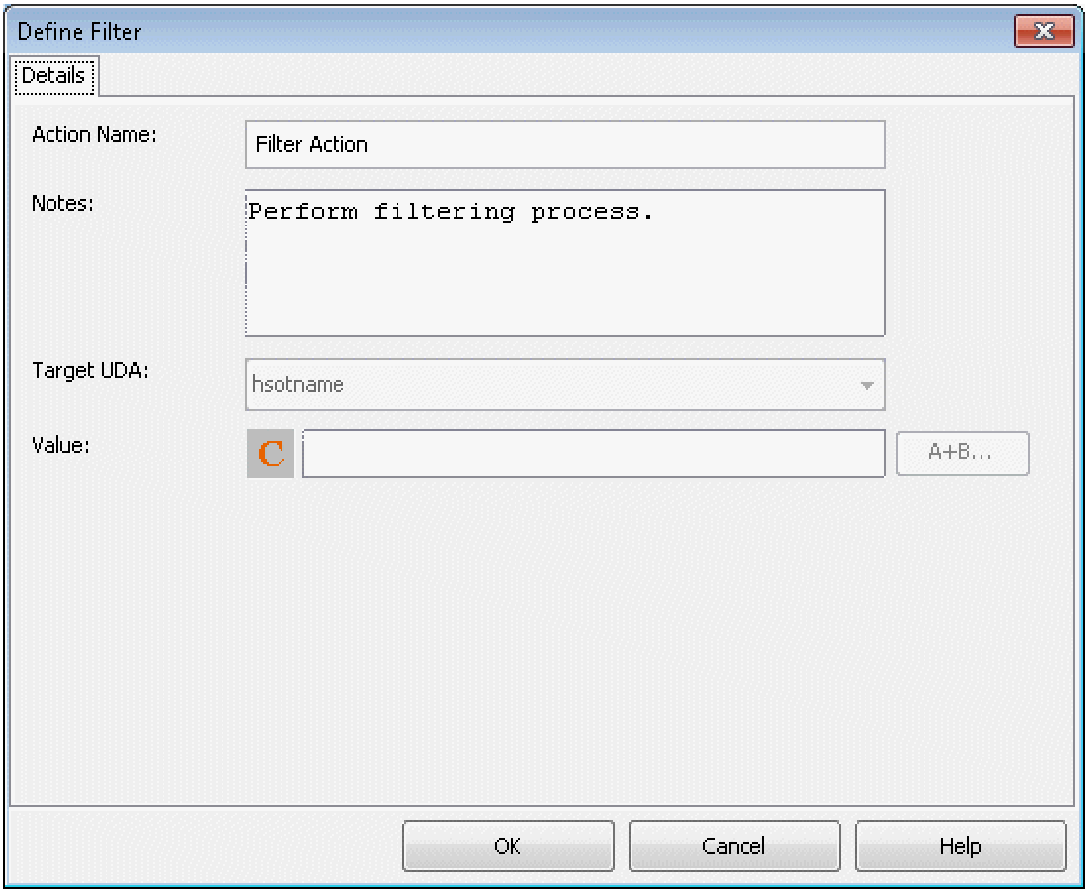
6.5 Defining Operation Component Filters (Using Java Scripts)

Using filter definitions that use Java scripts makes it possible to write Java scripts to process the execution results (output information) of operation components into any form of data.

Use is explained below.

1. Click the Settings >> Define filter (V14.1.x compatibility) >> Add button.

   The Define Filter dialog box is displayed.



Define Filter

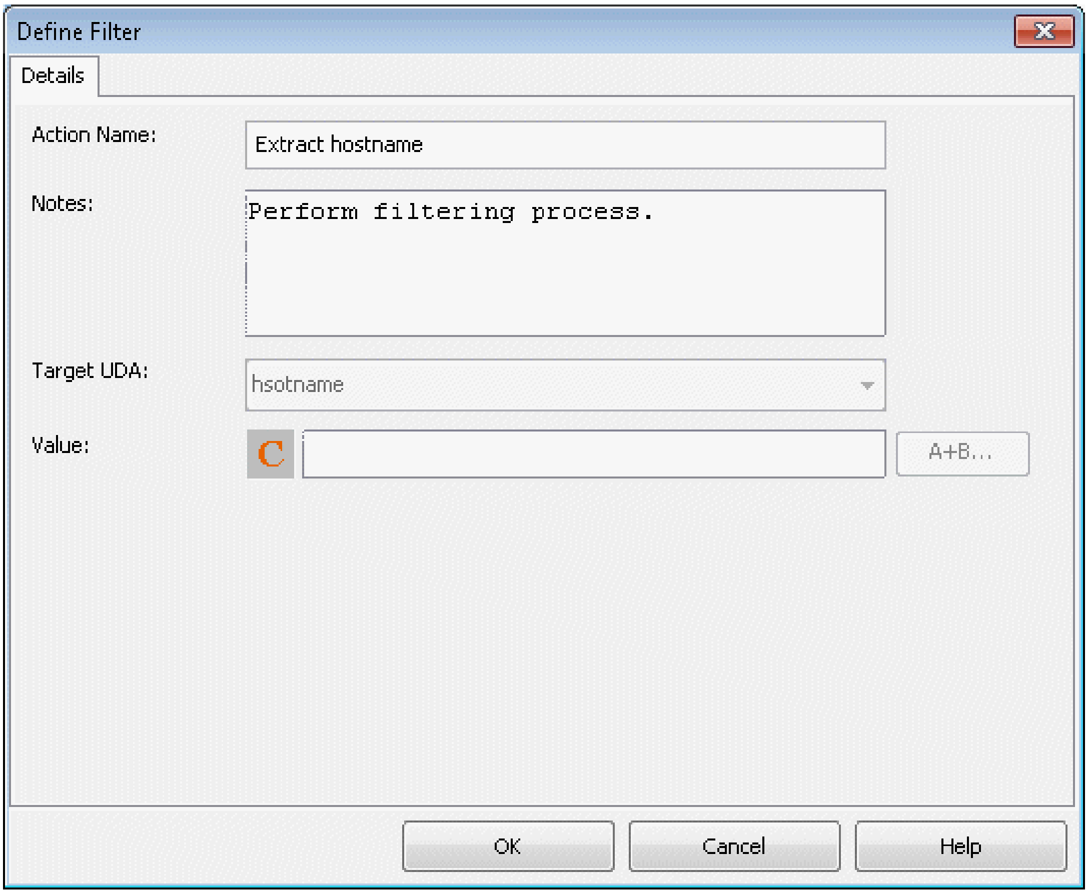
Action Name: Filter Action

Notes: Perform filtering process.

Target UDA: hsotname

Value:

2. Enter a filter name in the **Action Name** field.



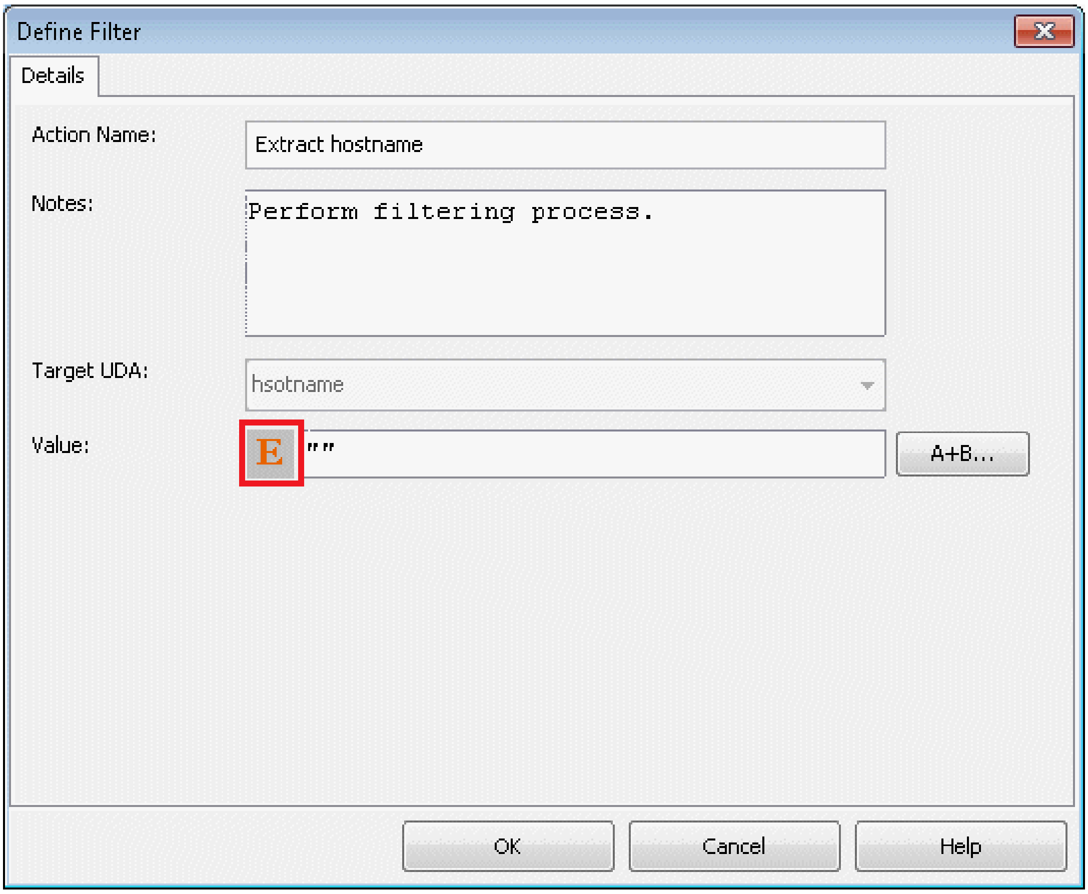
Action Name: Extract hostname

Notes: Perform filtering process.

Target UDA: hsotname

Value:

3. Click the **Expression Mode** button to change to the **E** button.



Action Name: Extract hostname

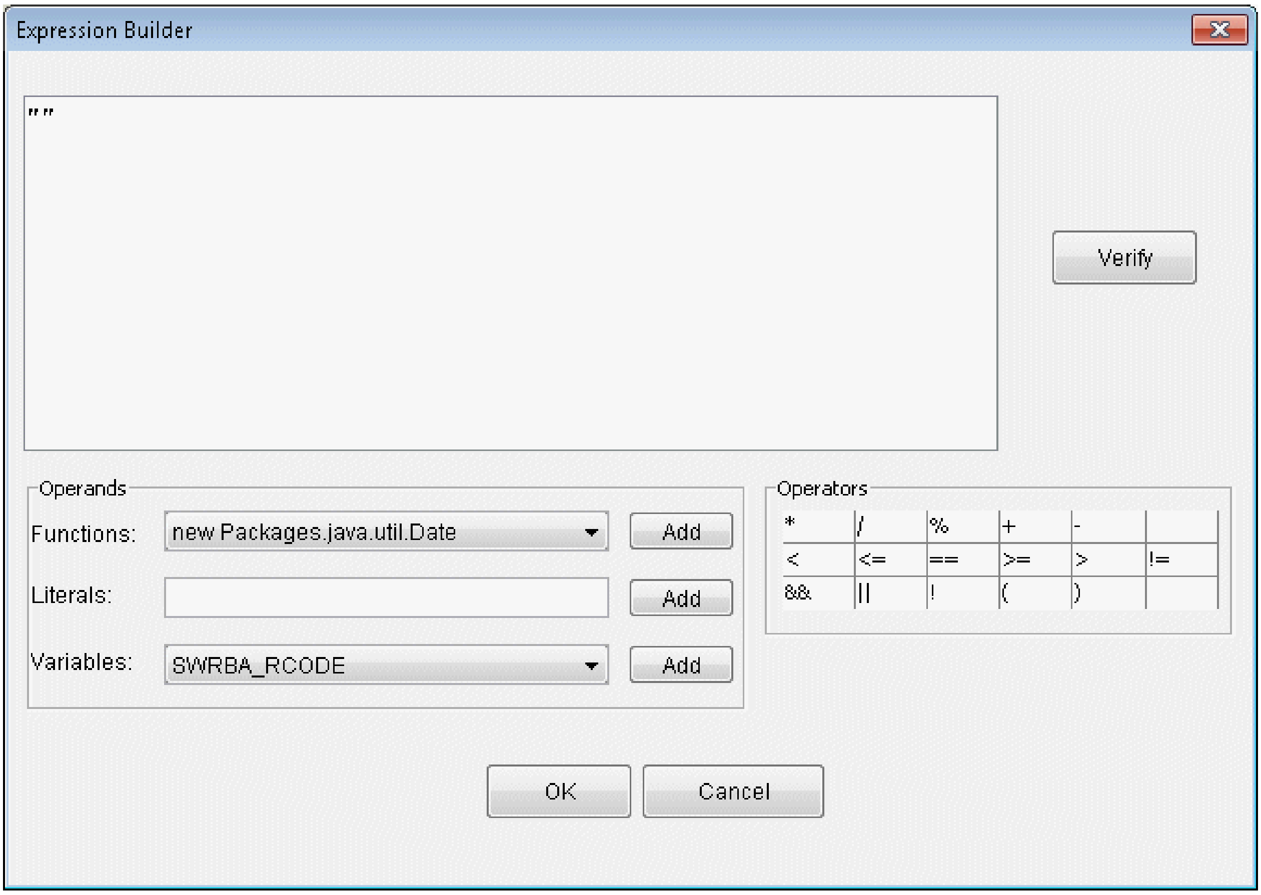
Notes: Perform filtering process.

Target UDA: hsotname

Value: E ""

4. Click the **A+B...** button.

   The **Expression Builder** dialog box will open.



Expression Builder

""

Operands

Functions: new Packages.java.util.Date

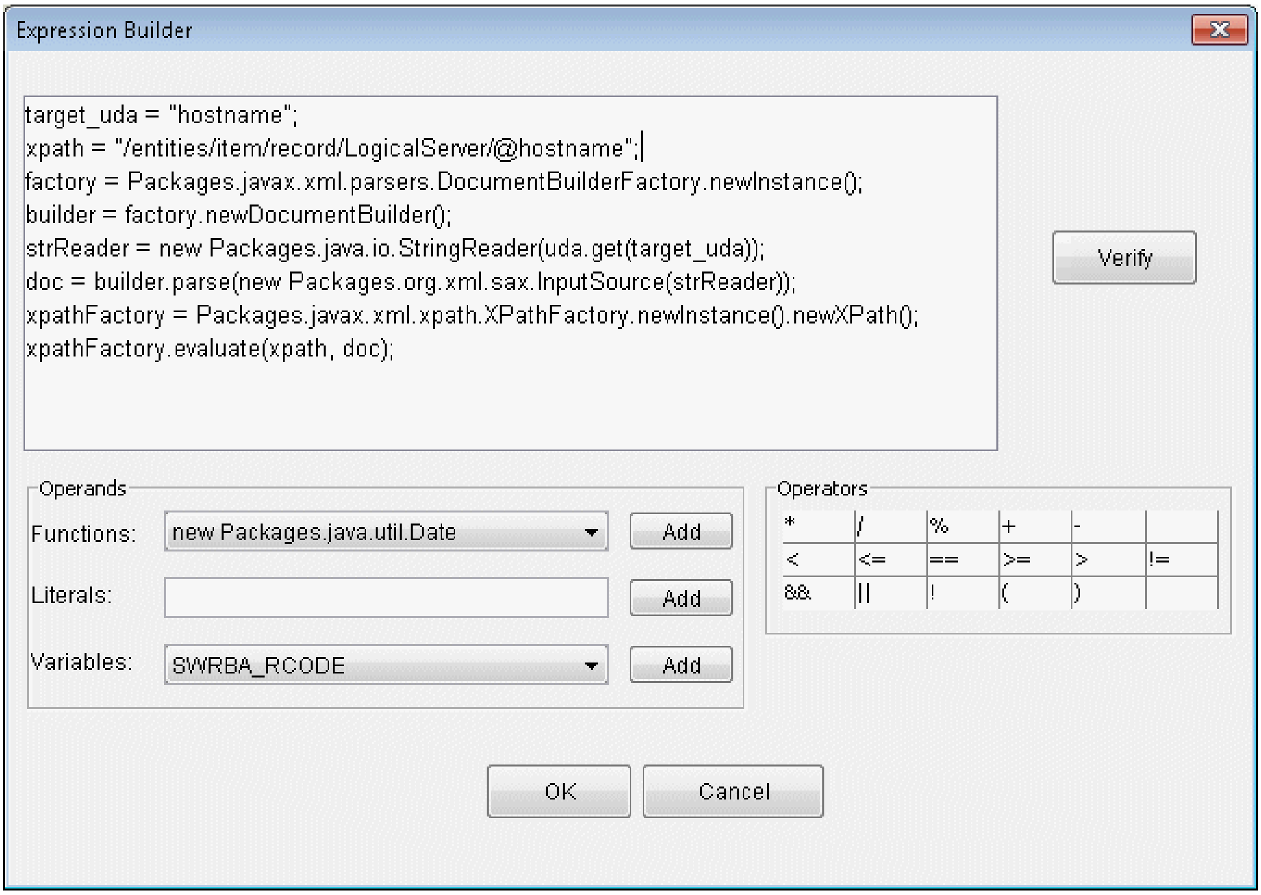
Literals:

Variables: SWRBA_RCODE

Operators

*	/	%	+	-	
<	<=	==	>=	>	!=
&&	||	!	(	)	

5. Enter JavaScript into the text field.



```
target_uda = "hostname";
xpath = "/entities/item/record/LogicalServer/@hostname";
factory = Packages.javax.xml.parsers.DocumentBuilderFactory.newInstance();
builder = factory.newDocumentBuilder();
strReader = new Packages.java.io.StringReader(uda.get(target_uda));
doc = builder.parse(new Packages.org.xml.sax.InputSource(strReader));
xpathFactory = Packages.javax.xml.xpath.XPathFactory.newInstance().newXPath();
xpathFactory.evaluate(xpath, doc);
```

Functions: new Packages.java.util.Date

Literals:

Variables: SWRBA_RCODE

6. Click the **OK** button in the **Expression Builder** dialog box.

7. Click the **OK** button in the **Define Filter** dialog box.

   The name of the filter created is displayed in **Settings** - **Filter Name**.

6.6 General Procedures

The following sections explain how to zoom, how to undo and redo changes, and how to print.

6.6.1 Using Undo and Redo

You can easily undo and redo changes to nodes, arrows, swimlanes, and groups.

To undo and redo a change:

- To undo the most recent change, right click the empty space in the Process Definition Editor and select Undo.
- If you decide you do not want to undo the change, right click the empty space in the Process Definition Editor and select Redo.

6.6.2 Zooming

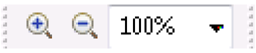
With zooming, you can increase or decrease the magnification of the Process Definition Editor.

To zoom, do one of the following:

- To increase magnification, click the Zoom In button in the toolbar.
- To decrease magnification, click the Zoom Out button in the toolbar.
- Select a predefined zooming level in the toolbar.

- Type a zooming level in the toolbar's zooming text field. Press the <Enter> key to apply the change.

Figure 6.6 Zooming Tools in the Toolbar



6.6.3 Printing Process Definitions

To print a process definition:

1. Open the process definition.
2. Select **File** >> **Print**.
3. Check the printer settings and change them according to your needs.
4. Click **OK**.

6.7 Adding and Editing Nodes

The palette contains buttons for all types of nodes that you can add to your process definition. You can move nodes and align them easily.

The following sections tell you how to add and remove nodes, how to define their basic properties and how to align them.

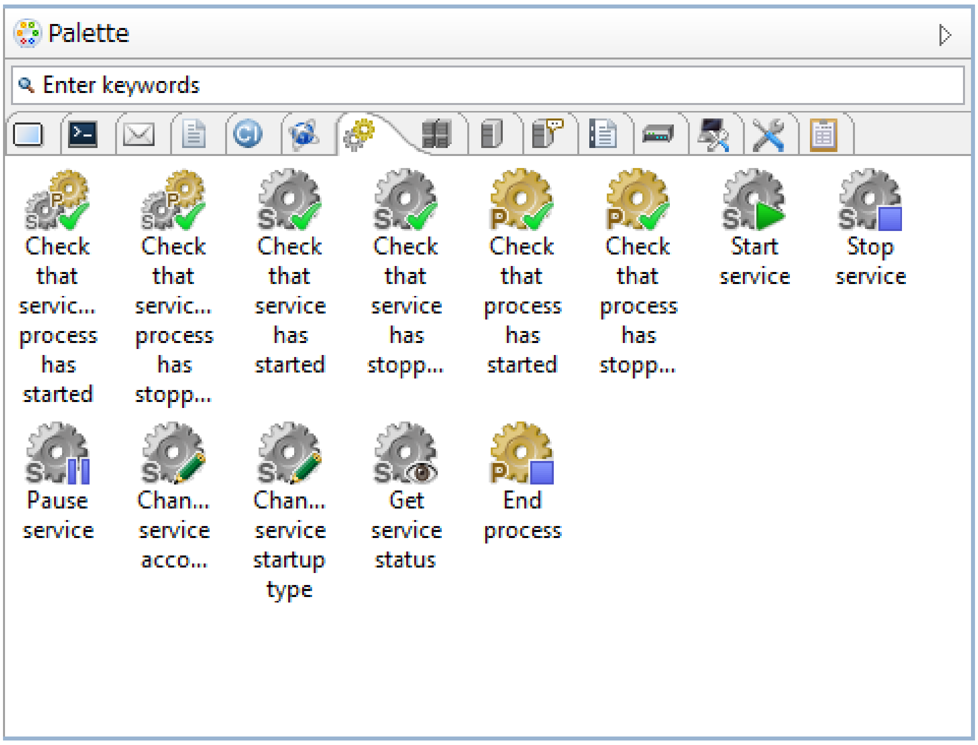
6.7.1 Adding Nodes

For each node that you want to add to your process definition, select the type of node you want to add from the palette. Ensure to add at least one Exit Node.

This section explains how to add nodes.

1. Display the process definition in the Process Definition Editor.
2. The palette is displayed.
3. Select the category for the node to be added from the **Category** tab.

Figure 6.7 Palette used to select the type of node

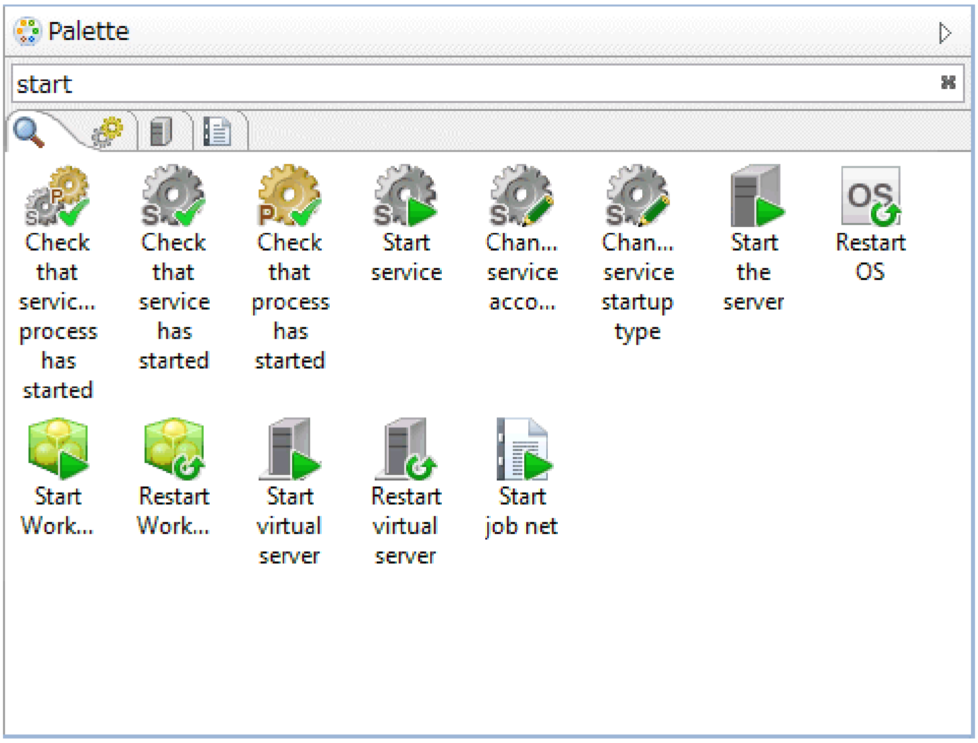

Palette
Enter keywords
Check that servic... process has started
Check that servic... process has stopp...
Check that service has started
Check that service has stopp...
Check that process has started
Check that process has stopp...
Start service
Stop service
Pause service
Chan... service acco...
Chan... service startup type
Get service status
End process

4. Click on the type of node to be added.
5. In the Process Definition Editor, point to the area where the node is to be placed. Click to add the node.

Filtering the Process Definition Editor Palette

Enter a keyword in the **Enter keywords** field. The **Filter** tab in the palette displays only those nodes that include the keyword entered in the **Enter keywords** field in the name of the node or in the description. Click the **x** button on the right to cancel the filter and restore the initial display. Select the **category** tab to select nodes from the filtering results for each category.

Figure 6.8 Filtered palette



6.7.2 Defining the Node Name and Description

You can change the default name of a node and provide a description.

To define the name and description for a node:

- To define the node name, select the node in the Process Definition Editor. Click its name and type the new name.

  The node name cannot be longer than 64 characters.

  For some nodes, e.g. for AND nodes, the node name is not displayed on the node symbol, but you can type a name in the Properties view.

- To provide a description, select the node and type the description in the Properties view.

6.7.3 Setting Activity Level Priority

You can set priority at the activity level for Activity Nodes, Voting Activity Nodes, and Compound Activity Nodes. By default, an activity is given a priority of -1. However, its priority can be changed to an integer greater than or equal to 0.

You can set priority at the activity level either through the General tab in the Properties view or by using a Java Action. For more information on setting activity level priority using Java Action, refer to section 11.2.4 Setting Priority for an Activity.

To set activity priority through the General tab:

1. Select the node to display the Properties view for the nodes.
2. Select the General tab in the Properties view.

3. Enter the priority value in the Priority field. This will set the priority for the activity.

6.7.4 Aligning Nodes

You can align two or more nodes.

To align nodes:

1. In the Process Definition Editor, select the nodes to be aligned. Keep the <Shift> or <Ctrl> key pressed while selecting nodes.
2. Depending how you want to align the nodes, click one of the following buttons in the toolbar:

Button	Description
	Aligns elements to the left edge of the element that you selected last (**Align Left**).
	Aligns elements vertically through the middle of the elements (**Align Center**).
	Aligns elements to the right edge of the element that you selected last (**Align Right**).
	Aligns elements to the top edge of the element that you selected last (**Align Top**).
	Aligns elements horizontally through the center of the elements (**Align Middle**).
	Aligns elements to the bottom edge of the element that you selected last (**Align Bottom**).

You can also align nodes with swimlanes or groups. Select the nodes and swimlanes/groups to be aligned and click the appropriate Align button in the toolbar. For information on how to work with swimlanes, refer to section 6.9 Adding and Editing Swimlanes. For information on how to work with groups, refer to section 6.10 Adding and Editing Groups.

6.7.5 Cutting, Copying, and Pasting Nodes

You can cut, copy and paste some nodes at a time. As the Start Node is mandatory and there is only one of it, you cannot cut the Start Node.

- **To cut a node:**

  Select the node. Right click and select Cut from the pop-up menu.

  The node is removed from the process definition along with its associated arrows.

- **To copy a node:**

  Select the node. Right click and select Copy from the pop-up menu.

- **To paste a node:**

  Select Edit >> Paste.

  The node is inserted near the original node.

Note

- When you paste a node that you copied or cut from another process definition, you need to add all User Defined Attributes that the pasted node is using in the source process definition.
- When an operation component node is copied and then pasted, the following message will be displayed. There is no need to take any action, however, since the displaying of the message does not mean that there is a problem.
  - Cannot add the following UDAs because UDA with the same name and identifier already exists;
    SWRBA_RCODE
    Please confirm and update them manually if necessary.

6.7.6 Moving Nodes

You can move nodes from one location to another.

To move a node:

1. Select the node to be moved.
2. Drag the node to the desired location. Systemwalker Runbook Automation Studio automatically adjusts all connected arrows.

You can also move several nodes at once. Hold down the <Shift> key or <Ctrl> key to select all nodes to be moved.

6.7.7 Removing Nodes

You can remove all nodes from your process definition except the Start Node.

To remove a node:

- Right click the node to be removed. Select Delete from the popup-menu.

The node is removed along with its associated arrows.

6.8 Adding and Editing Arrows

Arrows connect the nodes in your process definitions. With arrows, you define the flow of events.

The following sections tell you how to add and remove arrows, how to define their basic properties and how to change their shape.

6.8.1 Adding Arrows

Using the arrow button in the toolbar, you can draw arrows to connect nodes. Your process definition must have at least two nodes before you can draw any connecting arrows. You cannot draw an arrow that does not connect two nodes.

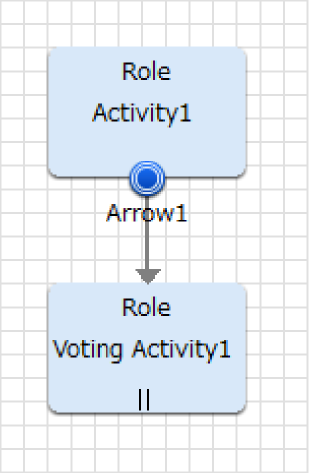
To add an arrow:

1. Click the Arrow button in the toolbar.
2. Move the cursor to the arrow's source node. From the cursor's shape you can recognize where you can start drawing the arrow.
3. Drag the cursor from the source node to the target node.

Every node has several connection points for the arrow to snap in. You can see the connection points when you point to the edges of the node.

The arrow cursor is still active and you can draw as many arrows as you wish. To disable it, press the <Esc> key or click the Select button in the toolbar.

Figure 6.9 Adding an Activity Node arrow

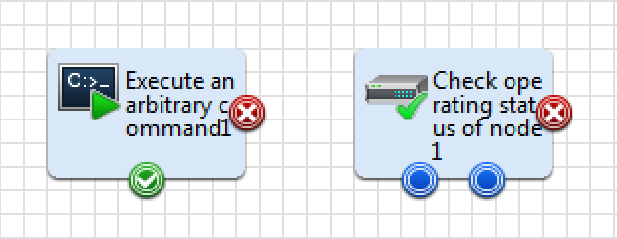


Adding an operation component arrow

When you select an operation component from the palette and move it to the Process Definition Editor, operation components with connection points that have symbols indicating normal or abnormal attached are positioned.

The number and positioning of the symbols depends on the type of operation component.

Figure 6.10 Adding an operation component arrow


C:>_
Execute an arbitrary command1
Check operating status of node 1

- Put the cursor over the symbol to see a tool tip displaying the default name of the arrow.
- The names of arrows are the default names defined for each type of operation component. This arrow name can be changed later.
- If there are not enough arrows, a message indicating this appears in the **Problems** view when you save the process definition or click the **Validate** button.
- Arrows cannot be added on sides without symbols.
- No more arrows can be added once arrows have been added to all symbols. If further branching is required, connect a Conditional node or Complex Conditional node to the symbol.
- Irrespective of the execution results of the operation component node (normal or abnormal), you can still draw arrows for the same succeeding node.
- Operation component nodes positioned in the process definition by older versions of Studio will be displayed as they were in the older version. However, only one arrow may be added to operation component nodes of older versions.
- The operation component nodes of older versions and operation component nodes with symbols may be used together.
- If the version of this product is later than V15, then the operation components positioned with Studio will all be operation component nodes with symbols. You may not position operation component nodes using the format of older versions.
- Operation component nodes positioned in the process definition by older versions of Studio may not be updated to operation component nodes with symbols. To use the operation component nodes with symbols, first delete the operation component nodes from the older version, then position the new operation component node.

The types of symbols and their meaning are as follows:

Symbol (if the version of this product is 15 or later)	Symbol (for older versions)	Meaning
●	●	Indicates selection.
✓	●	Indicates normal.
✗	●	Indicates failure or error.

6.8.2 Defining the Arrow Name

When you add an arrow, a default name for the action associated with the arrow is automatically added. You can change the default name.

Note

- The outgoing arrows of the following nodes must have different names:
    - Activity Node
    - Voting Activity Node
    - Compound Activity Node
    - Conditional Node
    - Complex Conditional Node

To define the arrow name:

1. Select the arrow.
2. Click the arrow's name and type the new name.

   Alternatively, you can type the new name in the Properties view.

6.8.3 Changing the Arrow Shape

You can change the shape of an arrow by dragging it. When the arrow is dragged, its drag point (the point that can be dragged) will automatically be either added or deleted.

To change the arrow shape:

1. Select the arrow so that its bend points become visible.
2. To reshape the arrow, drag the bend points to the desired location.

6.8.4 Reconnecting Arrows

You can change the source and the target of an arrow. For example, you might want an arrow to snap to another connection point of a node.

To move an arrow:

1. If you want the arrow to start from another location, drag the starting point of the arrow to the desired target.
2. If you want the arrow to point to another location, drag the arrow head to the desired target.

Note

The outgoing arrows of the following nodes must have different names:

- Activity Node
- Voting Activity Node
- Conditional Node
- Complex Conditional Node

Therefore, you might need to change the arrow name before reconnecting the arrow.

6.8.5 Removing Arrows

To remove an arrow:

1. Select the arrow to be removed.
2. Right click and select Delete.

6.9 Adding and Editing Swimlanes

With Systemwalker Runbook Automation Studio, you can use swimlanes to visually group activities performed by the same Role.

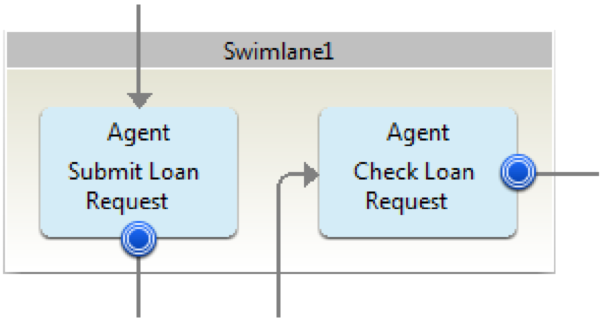
The following sections explain how to add and remove swimlanes, and how to change their appearance.

6.9.1 Adding Swimlanes

To add a swimlane:

1. Display the process definition in the Process Definition Editor.
2. Click the swimlane button in the palette.
3. In the Process Definition Editor, drag to draw the swimlane.

Figure 6.11 Adding Swimlanes



6.9.2 Defining the Swimlane Title

The swimlane title typically indicates who performs the activities placed on the swimlane. You can change the default title that is automatically generated when you add a swimlane.

To define a swimlane title:

1. Select the swimlane.
2. Click the swimlane's title and type the new title.

Alternatively, you can type the new title in the Properties view.

6.9.3 Aligning Swimlanes

You can align two or more swimlanes.

To align swimlanes:

1. Select the swimlanes to be aligned. Keep the <Shift> or <Ctrl> key pressed while you click the swimlanes.
2. Click one of the following buttons in the toolbar:

Button	Description
	Aligns elements to the left edge of the element that you selected last (**Align Left**).
	Aligns elements vertically through the middle of the elements (**Align Center**).
	Aligns elements to the right edge of the element that you selected last (**Align Right**).
	Aligns elements to the top edge of the element that you selected last (**Align Top**).
	Aligns elements horizontally through the center of the elements (**Align Middle**).
	Aligns elements to the bottom edge of the element that you selected last (**Align Bottom**).

You can also align nodes with swimlanes. Select the nodes and swimlanes to be aligned and click the appropriate Align button in the toolbar.

6.9.4 Adjusting Swimlane Size

You can manually adjust the height and width of swimlanes. Also, you can scale swimlanes to the same width or height.

To adjust swimlane size, do one of the following:

- To adjust the height and width manually:
  a. Select the swimlane.
  b. Point to one of the sizing handles and drag the swimlane to the new size.
- To scale swimlanes to the same width or height:
  a. Select the swimlanes to be scaled. Keep the <Shift> or <Ctrl> key pressed while selecting the swimlanes.
  b. Click one of the following buttons in the toolbar:

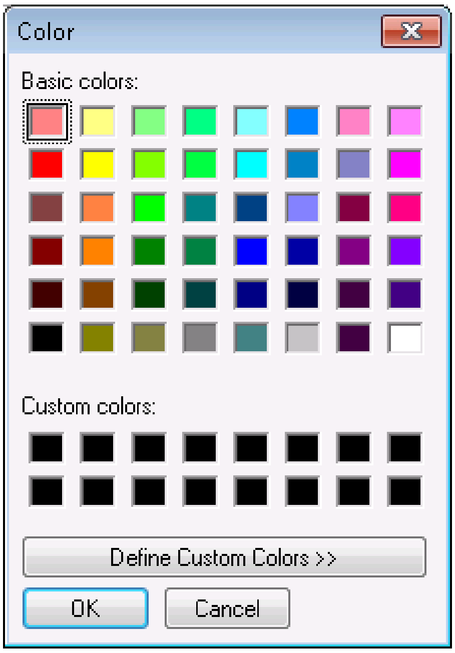
Button	Description
↕	Adjusts the height to the height of the swimlane that you selected last.
↔	Adjusts the width to the width of the swimlane that you selected last.

6.9.5 Changing Swimlane Color

To change the color of a swimlane:

1. Select the swimlane.
2. In the Properties view, select color.
3. Click the button that appears in the Value column to open the color palette.

Figure 6.12 Picking a Color from the Palette



4. Pick a color from the palette.
5. Click OK.

6.9.6 Changing Swimlane Style

You can place the swimlane title on the left-hand side or on the upper side of a swimlane.

To modify this setting as a process definition for all swimlanes:

1. Click the Process Definition Editor to make it active.
2. Do one of the following:
   - To place the title on the left-hand side, select **Edit** >> **Swimlane Style** >> **Left**.
   - To place the title on the upper side, select **Edit** >> **Swimlane Style** >> **Top**.

You can set your preferred style for Process Definition Editors that you open in the future. Select **Window** >> **Preferences** and then **Interstage BPM Studio** >> **Appearance**.

6.9.7 Cutting, Copying and Pasting Swimlanes

You can cut, copy, and paste a single swimlane at a time.

1. Cutting a swimlane:

   Select the swimlane. Right click and select **Cut** from the pop-up menu.
2. Copying a swimlane:

   Select the swimlane. Right click and select **Copy** from the pop-up menu.
3. Pasting a swimlane:

   Select **Edit** >> **Paste**.

   The swimlane is inserted near the original swimlane.

6.9.8 Moving Swimlanes

You can move swimlanes from one location to another.

To move a swimlane:

1. Click the swimlane to be moved.
2. Drag the swimlane to the desired location.

You can also move several swimlanes at once. Hold down the <Shift> key or <Ctrl> key to select all swimlanes to be moved.

6.9.9 Removing Swimlanes

To remove a swimlane:

1. Select the swimlane.
2. Right click and select **Delete** from the pop-up menu.

6.10 Adding and Editing Groups

Systemwalker Runbook Automation Studio, you can use groups to visually group activities according to custom categories.

The following sections explain how to add and remove groups, and how to change their appearance.

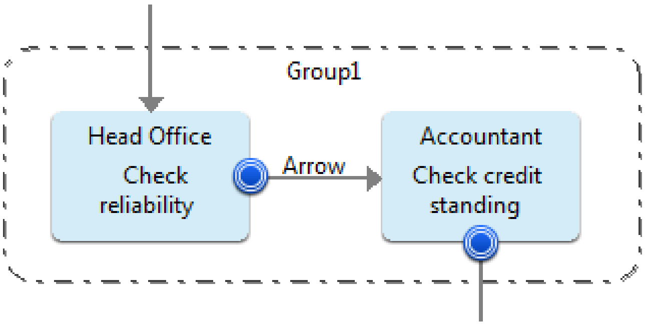
6.10.1 Adding Groups

To add a group:

1. Display the process definition in the Process Definition Editor.
2. Click the group button in the palette.

3. In the Process Definition Editor, drag to draw the group.

Figure 6.13 Adding Groups



6.10.2 Defining the Group Title

The group title typically indicates the category of activities that are grouped. You can change the default title that is automatically generated when you add a group.

To define a group title:

1. Select the group.
2. In the Properties view, enter a new title in the name field.

6.10.3 Aligning Groups

You can align two or more groups.

To align groups:

1. Select the groups to be aligned. Keep the <Shift> or <Ctrl> key pressed while you click the groups.
2. Click one of the following buttons in the toolbar:

Button	Description
	Aligns elements to the left edge of the element that you selected last (**Align Left**).
	Aligns elements vertically through the middle of the elements (**Align Center**).
	Aligns elements to the right edge of the element that you selected last (**Align Right**).
	Aligns elements to the top edge of the element that you selected last (**Align Top**).
	Aligns elements horizontally through the center of the elements (**Align Middle**).
	Aligns elements to the bottom edge of the element that you selected last (**Align Bottom**).

You can also align nodes with groups. Select the nodes and groups to be aligned and click the appropriate Align button in the toolbar.

6.10.4 Adjusting Group Size

You can manually adjust the height and width of groups. Also, you can scale groups to the same width or height.

To adjust group size, do one of the following:

- To adjust the height and width manually:
  - Select the group.
  - Point to one of the sizing handles and drag the group to the new size
- To scale groups to the same width or height:
  - Select the groups to be scaled. Keep the <Shift> or <Ctrl> key pressed while selecting the groups.

- Click one of the following buttons in the toolbar:

Button	Description
	Adjusts the height to the height of the group that you selected last.
	Adjusts the width to the width of the group that you selected last.

6.10.5 Cutting, Copying and Pasting Groups

You can cut, copy, and paste a single group at a time.

1. Cutting a group:

   Select the group. Right click and select **Cut** from the pop-up menu.

2. Copying a group:

   Select the group. Right click and select **Copy** from the pop-up menu.

3. Pasting a group:

   Select **Edit** >> **Paste**.

   The group is inserted near the original group.

6.10.6 Moving Groups

You can move groups from one location to another.

To move a group:

1. Click the group to be moved.
2. Drag the group to the desired location.

You can also move several groups at once. Hold down the <Shift> key or <Ctrl> key to select all groups to be moved.

6.10.7 Removing Groups

To remove a group:

1. Select the group.
2. Right click and select **Delete** from the pop-up menu.

6.11 Adding Annotations

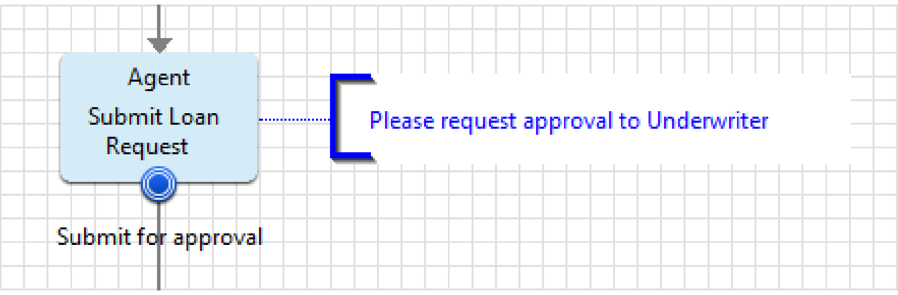
You can add annotations to make comments on your process definition and explain important aspects of your process design.

To add an annotation:

1. Display the process definition in the Process Definition Editor.
2. Click the annotation button in the palette.
3. In the Process Definition Editor, drag to draw the annotation.
4. Click the annotation and type your comments.
5. If the annotation refers to a particular node, you can connect them to visually represent their relation. To do so:
   a. Click the Arrow button in the toolbar.
   b. Drag the cursor from the annotation to the node it refers to.

The following example shows an annotation that has been connected to a node.

Figure 6.14 Adding Annotations and Connecting them to Nodes



Note

You can connect only one annotation to a node.

6. You can resize the annotation as follows:

   a. Select the annotation.

   b. Point to one of the sizing handles and drag the annotation to the new size.

6.12 Enabling Recall for Nodes

The recall functionality provides you with the option to enable or disable recall for completed work items. By default recall is enabled. This feature is available only for Activity, Voting, and Delay node

Note

- Only completed work items can be recalled.
- Only single level recall of work item is possible. Multiple level recalls of work items cannot be done.

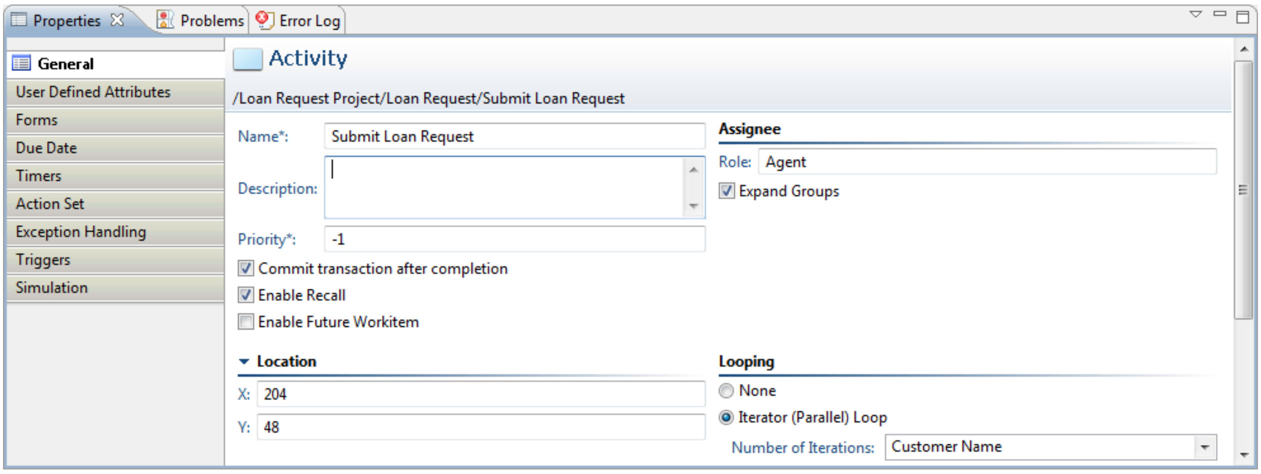
To enable recall for an activity:

1. Select the Activity, Voting, or Delay Node to display the Properties view for the node.
2. In the General tab, select or clear the **Enable Recall?** check box to enable or disable recall respectively.

6.13 Enabling Future Work Items on Activity Node

The following Future Work Items check box are displayed, but it is not to use on Systemwalker Runbook Automation

Figure 6.15 To enable Future Work Items on an activity node:



/Loan Request Project/Loan Request/Submit Loan Request

Name*: Submit Loan Request

Description:

Priority*: -1

☑ Commit transaction after completion

☑ Enable Recall

☐ Enable Future Workitem

Location

X: 204

Y: 48

Assignee

Role: Agent

☑ Expand Groups

Looping

○ None

● Iterator (Parallel) Loop

Number of Iterations: Customer Name

6.14 Assigning Process Instance Owners

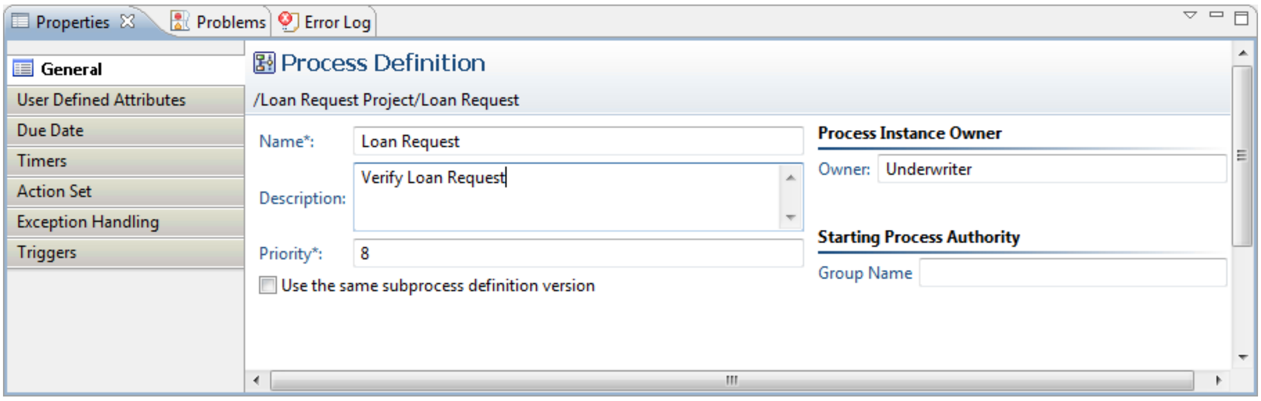
Process instance owners can edit the process instances that they own while the process instances are running.

By default, the owner of the process definition is the owner of the process instances that are created from the process definition. However, you can assign process instance ownership to another Role.

To assign process instance owners:

1. Click the empty space in the Process Definition Editor to display the **Properties** view for the process definition.
2. Select the **General** tab in the **Properties** view.
3. Type the name of the Role in the **Owner** field in the **Process Instance Owner** area. This will assign the process instance to the owner.

Figure 6.16 Assigning Process Instance Owners



/Loan Request Project/Loan Request

Name*: Loan Request

Description: Verify Loan Request

Priority*: 8

☐ Use the same subprocess definition version

Process Instance Owner

Owner: Underwriter

Starting Process Authority

Group Name

6.15 Assigning Process Start Authority

To assign process start authority, only user belonged to that group and system administrator can start published automated process start authority.

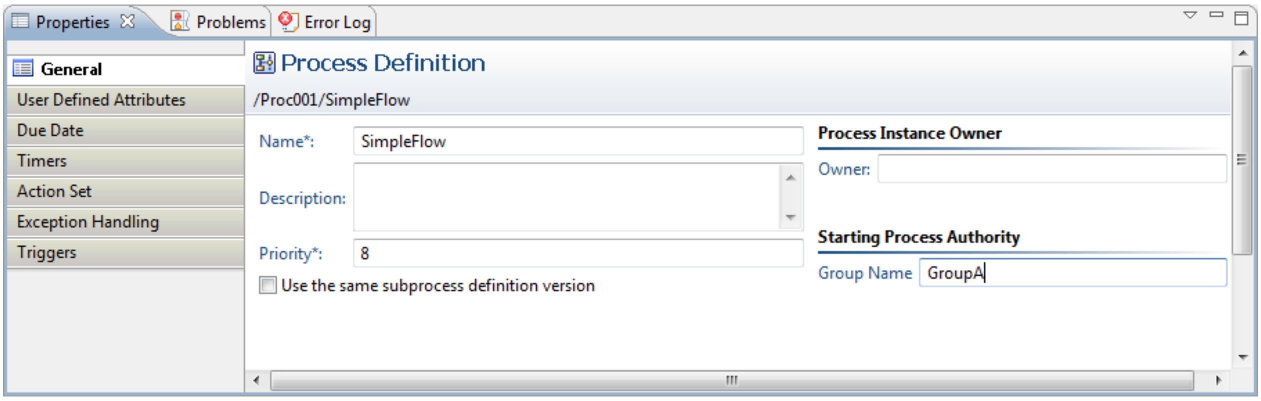
User (except of system administrator) who not belonged to group which set process start authority, cannot start process automated set. if omit setting process definition authority, all user start published Automated Operation Process.

This section explains how to assign process start authority.

1. Click the empty space in the Process Definition Editor to display the Properties view for the process definition.
2. Select the general tab on properties view.

3. Specify group name which user can start process belonged, to **process start authority group.**

Figure 6.17 To assign process start authority



Process Definition

/Proc001/SimpleFlow

Name*: SimpleFlow

Description:

Priority*: 8

☐ Use the same subprocess definition version

Process Instance Owner

Owner:

Starting Process Authority

Group Name GroupA

Group name on process start authority specifies one group name. The upper limit of length of group name is 200 characters.

6.16 Assigning Activities to Roles

When modeling an activity, you assign it to a Role according to who is responsible for completing the activity. Each activity (that is each Activity Node, Voting Activity Node, and Compound Activity) must be assigned to a Role. If an activity is not assigned, the process instances created from the process definition will go into error state.

You can assign an activity

- To a Role
- To particular users in a Role

This section explains how to assign activities to Roles. Refer to section 11.2.1 Assigning an Activity to a User for instructions on how to assign an activity to particular users.

To assign activities to roles:

1. Select the Activity Node, Voting Activity Node, or Compound Activity.
2. In the **General** tab of the **Properties** view type the name of the role, in the **Role** field in the **Assignee** area.

6.17 Configuring Work Item Generation

At run time, when an Activity Node or Compound Activity becomes active, work items are generated and assigned to the users having the Role to which the Activity Node has been assigned. The number of generated work items depends on how the Activity Node has been configured. These are the options:.

6.18 Specifying User Defined Attributes

User Defined Attributes (UDAs) contain the data that process participants need to access, modify or add. UDAs are global variables that are defined on process definition level so that all nodes within a process instance can access all UDAs.

Using UDAs, you can specify the behavior of nodes and store data for process execution.

UDAs hold values in a running process. Process participants provide the values through forms, JavaScripts or Java Actions. UDAs consist of a name and an identifier (ID). Both values are stored in the database. By default, UDA identifiers are automatically set by the system. However, you can also specify your own identifiers. For every user interaction, the UDA name is used. Whenever necessary, the name is automatically mapped to the UDA identifier. The identifier is used for all purposes that do not allow special characters, for example when creating QuickForms or using JavaScript.

For example, a purchase requisition process might have UDAs for the article to be purchased, the quantity and cost. You can set the values by filling in a form.

To specify User Defined Attributes:

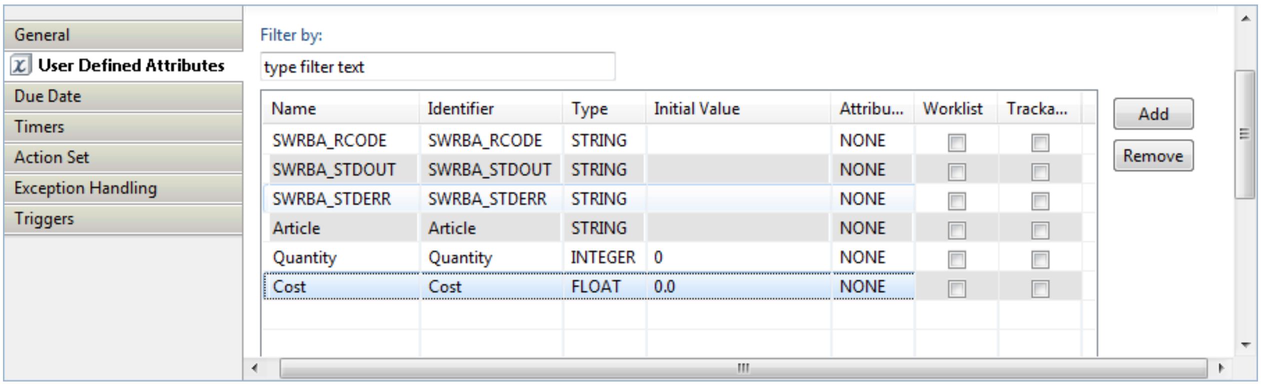
1. Click the empty space in the Process Definition Editor to display the Properties view for the process definition.
2. Select the **User Defined Attributes** tab.
3. Click **Add**. Specify the following parameters:

Field	Description
Name	Name of the UDA The UDA name is descriptive and user-defined. It can contain up to 64 characters, including special characters. The name must not start with two underscores (__) because two underscores are used as a prefix of UDAs created and maintained by the system.
Identifier	Identifies a UDA by providing a unique name. By default, UDA Identifiers are system-generated. However, you can also specify your own IDs. UDA Identifiers can contain up to 32 characters. Identifiers must not contain any special characters (all characters except 'a' - 'z', 'A' - 'Z', '0' - '9'). If you do not specify a value for the ID, it is formed automatically by taking the name and removing all of the characters except for ASCII letters. This process is called "UDA sanitization". If the "sanitized name" (ID) is longer than 32 characters, empty or not unique, the ID will be composed of the prefix 'uda<number>', for example 'uda1'. The number is incremented by one each time a new ID is created. **Note:** The Identifier field is only displayed if you select the **Show Identifier** check box in the Properties view.
Type	UDA data types You can distinguish the following data types: BIGDECIMAL, BOOLEAN, DATE, FLOAT, INTEGER, LONG, STRING, XML
Initial Value	Initial value of the UDA Process participants can change this value. If you select a UDA of type XML, two additional buttons are displayed. You can use these buttons to configure the XML data type. For more information, refer to section 6.18.1 Specifying User Defined Attributes of Type XML. You can set an initial value within the following ranges: - INTEGER Type: - **Max Value:** 2147483647 - **Min Value:** - 2147483648 - LONG Type: - **Max Value:** 9223372036854775807 - **Min Value:** - 9223372036854775808 - Float Type: The number of significant figures is 8. - BIGDECIMAL Type: The number of significant figures is 17.
Worklist	Indicates if the UDA is a Worklist UDA. Worklist UDAs allow process participants to filter and sort the worklist and to retrieve worklist items faster. If you select a UDA of type XML, this column will be disabled. **Note:** Worklist UDAs of type STRING cannot have values that are longer than 256 characters.
Trackable	Enables or disables UDA tracking. If you enable UDA tracking, changes made to any UDA value during process execution will be recorded. **Note:** When designing the process definition, you define the initial setting for all process instances created from that process definition.

4. Repeat the last step for each UDA that you want to specify.

   The following screen shows the Properties view for an Activity Node, in which three UDAs have been specified for the node:

   Figure 6.18 Specifying UDAs



Name	Identifier	Type	Initial Value	Attribu...	Worklist	Tracka...
SWRBA_RCODE	SWRBA_RCODE	STRING		NONE	□	□
SWRBA_STDOUT	SWRBA_STDOUT	STRING		NONE	□	□
SWRBA_STDERR	SWRBA_STDERR	STRING		NONE	□	□
Article	Article	STRING		NONE	□	□
Quantity	Quantity	INTEGER	0	NONE	□	□
Cost	Cost	FLOAT	0.0	NONE	□	□

5. If you want to display the UDA identifiers, select the **Show Identifier** checkbox.

   A new Identifier field is added to the **User Defined Attributes** tab. You can specify new IDs or change the IDs of existing UDAs.

   - Specify a new ID: Select an empty row in the **Identifier** field, and type in a new ID.

   - Change an existing ID: Select an existing ID in the **Identifier** field, and type in another ID.

   Note

   If you change an ID, ensure that you also update the IDs that are referenced in JavaScript expressions, QuickForms, or data mappings.

   If you do not display the UDA identifiers, or leave the **identifier** fields empty, default identifiers (for example, 'Variable1', 'Variable2', 'Variable3') will be automatically generated.

6. If you want to rename a UDA, double-click its name and change it as required.

7. If you want to remove a UDA, select it and click **Remove**.

   You can select multiple UDAs using the <Shift> or <Ctrl> keys.

   Note

   You cannot rename or remove UDAs that are referenced in the process definition. If a UDA cannot be renamed or removed, a message is displayed telling you so. The message also displays the locations where the UDA is being used.

6.18.1 Specifying User Defined Attributes of Type XML

In Systemwalker Runbook Automation Studio, you can create User Defined Attributes (UDAs) of type XML to enhance the flexibility of the system.

XML data types allow you to invent as many different elements and attributes as you need. In addition, UDAs of type XML are easy to use, unlimited and self-defining.

To specify a UDA of type XML:

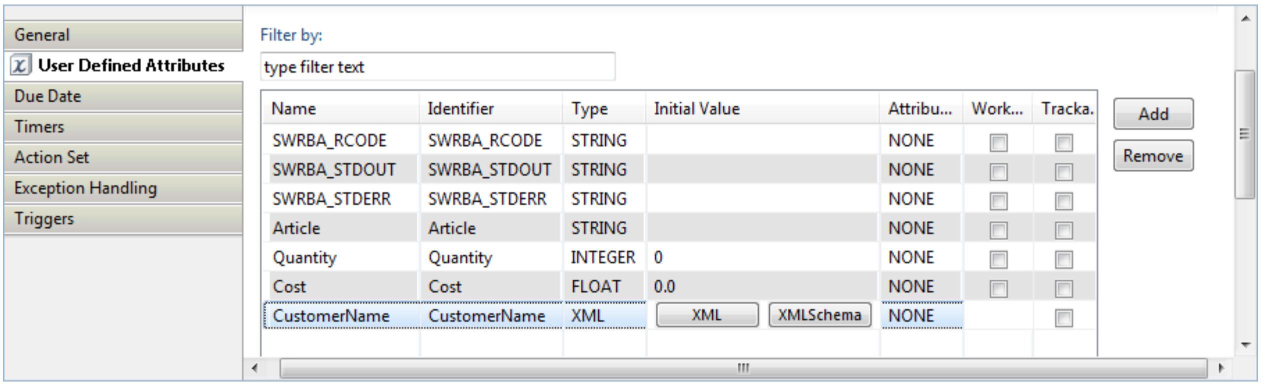
1. Click the empty space in the Process Definition Editor or select a node, to display the **Properties** view for the process definition or the selected node respectively.

2. Select the **User Defined Attributes** tab.

3. Click **Add**, and select XML as type for the UDA.

   In the **Properties** view, the **Initial Value** column displays the **XML** and **XML Schema** buttons. If you select a non-XML UDA type, these buttons are not available.

   Figure 6.19 Displaying UDAs



Name	Identifier	Type	Initial Value	Attribu...	Work...	Tracka.
SWRBA_RCODE	SWRBA_RCODE	STRING		NONE	□	□
SWRBA_STDOUT	SWRBA_STDOUT	STRING		NONE	□	□
SWRBA_STDERR	SWRBA_STDERR	STRING		NONE	□	□
Article	Article	STRING		NONE	□	□
Quantity	Quantity	INTEGER	0	NONE	□	□
Cost	Cost	FLOAT	0.0	NONE	□	□
CustomerName	CustomerName	XML	XML XMLSchema	NONE		□

Note

UDAs of type XML cannot be Worklist UDAs. Hence, if you select a UDA of type XML, the **Worklist** column will be disabled.

4. In the **Name** and **Identifier** fields, specify the general parameters for the UDA of type XML. Refer to section 6.18 Specifying User Defined Attributes for a detailed description of these parameters.

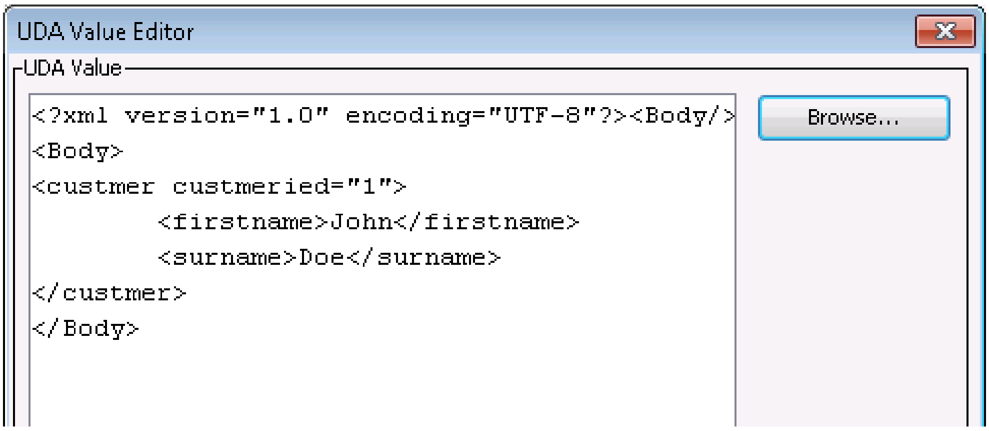
5. In the **Initial Value** column, select the **XML** button. The **UDA Value Editor** dialog box is displayed.

6. Type in a value for the UDA, or browse for an XML file.

   - Type in a UDA value: In the **UDA Value** field, type in a value for the UDA.
   - Browse for a file: Click the **Browse** button to select a file from the local file system. The **UDA Value Editor** dialog displays the content of the file. Select a value from the content of this file, and click **OK**.

     The following example shows how to specify an XML value in the **UDA Value Editor** dialog:

     Figure 6.20 Specifying a UDA value



```
<?xml version="1.0" encoding="UTF-8"?><Body/>
<Body>
<custmer custmeried="1">
        <firstname>John</firstname>
        <surname>Doe</surname>
</custmer>
</Body>
```

7. Optional: In the **Initial Value** column, select the **XML Schema** button.

   The **XML Schema Editor** dialog is displayed. You can use this dialog to type in an XML Schema for the UDA, or browse for a schema file (.xsd) in the local file system.

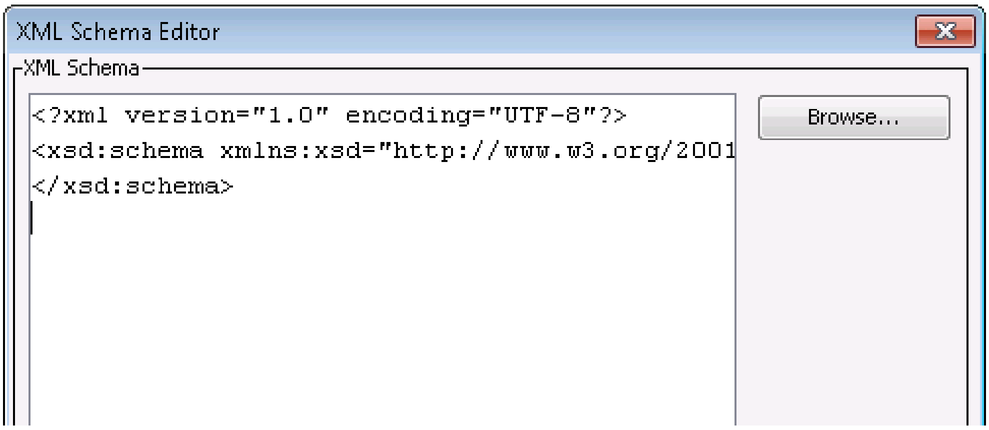
8. Type in an XML Schema for the UDA, or browse for an XML file.

   - Type in an XML Schema: In the **XML Schema** field, type in the XML Schema for the UDA.

- Browse for a file: Click the **Browse** button to select a file from the local file system. The **XML Schema Editor** dialog displays the content of the file. Select a value from the content of this file, and click **OK**.

The following example shows how to specify an XML Schema in the **XML Schema Editor** dialog:

Figure 6.21 Specifying an XML Schema



```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001
</xsd:schema>
```

9. Repeat the last step for all UDAs of type XML that you want to specify.

   Similarly to all other UDA types, you can do the following:

   - Display the UDA identifier: If you want to display or change the UDA identifier, select the **Show Identifier** check box. The identifiers are then displayed in a separate Identifier column.

   - Rename the UDA: If you want to rename the UDA, double-click its name and type in a new name.

   - Remove a UDA: If you want to remove a UDA, select it and click **Remove**.

6.19 Looping Definitions

In Process Definitions, a node may need to generate multiple node instances to eliminate the need to create individual nodes.

Usually, only one node instance is generated for each node within the process definition at the time of execution.

Two iterative methods can be defined in looping definitions-Iterator (Parallel) Loop and Sequential Loop-while a single node definition can generate multiple node instances. The nodes that define this looping are called the Iterator (Parallel) Loop node and Sequential Loop node.

In the Iterator (Parallel) Loop node, multiple node instances are generated simultaneously in parallel. In the Sequential Loop node, node instances are generated sequentially till a certain condition is satisfied.

For example, the order management process definition requires that the same activity (such as "approve") be repeated for all items included in an order. Defining the Iterator (Parallel) Loop in the "approve" Activity Node eliminates the need to add a new Activity Node for each item included in the order. In the Iterator (Parallel) Loop node, user defined attributes can be used to specify settings for the number of node instances for the process definition design.

In the process definition for shipments of orders received, an activity or subprocess to add goods to goods shipped may, in some cases, execute repeatedly till there are no more items in the order. In such cases, the Sequential Loop node can be used to generate node instances till the specified condition is satisfied.

In the Sequential Loop node, the repeating condition and the maximum number of repetitions can be specified, either separately or collectively, to perform settings for the number of node instances for the process definition design.

The following items describe how to define the Iterator (Parallel) Loop and Sequential Loop.

6.19.1 Iterator (Parallel) Loop

In the Iterator (Parallel) Loop node, multiple node instances are generated simultaneously in parallel. The Activity Node, Subprocess Node, and Chained-Process Node can each be defined as the Iterator (Parallel) Loop node.

The following is a description of how to define the Iterator (Parallel) Loop node.

- For Activity Nodes and Subprocess Nodes:

  1. Select the node you want to define as the Iterator (Parallel) Loop node.
  2. On the **General** tab in the **Properties** view, select **Iterator (Parallel) Loop** under **Looping**.
  3. Select an INTEGER-type user-defined attribute (UDA) from the **Number of Iterations** drop-down list. The integer value of the selected UDA is deemed to be the number of node instances generated.

- For Chained-Process Nodes:

  1. Select the node you want to define as the Iterator (Parallel) Loop node.
  2. Select the **General** tab in the **Properties** view.
  3. Select an INTEGER-type UDA from the **Number of Iterations** drop-down list under **Iterator Setting**. The integer value of the selected UDA is deemed to be the number of node instances generated.

Note

- If a UDA is not selected from the **Number of Iterations** drop-down list under **Iterator (Parallel) Loop**, the selected node will not be defined as the Iterator (Parallel) Loop node.
- The number of node instances generated by the Iterator (Parallel) Loop node is the value of the user-defined attribute UDA selected from the **Number of Iterations** drop-down list. The maximum value that can be set for a UDA is related to the number of people-in-charge specified in roles. Do not allow the product of people-in-charge and the number of node instances to exceed a value of 2000, even though this affects the setup status of the server environment and system load. For example, if there are 10 people-in-charge, the value of the UDA should be kept within 200.
- Only one (outward-facing) arrow can be drawn from the Activity Node defined as the Iterator (Parallel) Loop node.
- It is not possible to set a trigger for the Activity Node defined as the Iterator (Parallel) Loop node.

Select **None** under **Looping** to disable the Iterator (Parallel) Loop node.

The Iterator (Parallel) Loop is represented by three vertical parallel lines. Below is a representation of an Activity Node, Subprocess Node, and Chained-Process Node when defined as the Iterator (Parallel) Loop node.

Figure 6.22 Activity Node

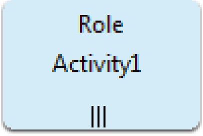


Figure 6.23 Subprocess Node

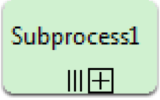
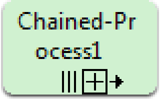


Figure 6.24 Chained-Process Node



6.19.2 Sequential Loop

In the Sequential Loop node, node instances are generated sequentially till a certain condition is satisfied. The Activity Node, Subprocess Node, and Compound Activity Node can be defined as the Sequential Loop node.

The following is a description of how to define the Sequential Loop node.

1. Select the node you want to define as the Sequential Loop node.

2. On the **General** tab in the **Properties** view, select **Sequential Loop** under **Looping**.

3. Settings for the repeating condition and the maximum number of repetitions for generating node instances can be performed as follows.

   - To specify settings for the repeating condition, select the **Condition** check box and type a JavaScript expression for the condition. It is also possible to create an expression by using the **Expression Builder** dialog box to specify it directly into the field. Node instances will be generated for as long as the specified repeated condition remains true (valid).

     Note

     The condition is assessed prior to generation of the node instance. If the condition is false from the very first time it is assessed, no node instances are generated.

   - To specify settings for the maximum number of repetitions, select the **Number of Iterations** check box and either type a value in the field or select an INTEGER-type UDA from the drop-down list. The integer value of the specified value or the selected UDA is deemed to be the number of node instances generated. Either addition or subtraction can be selected for the maximum number of repetitions counter.

     - If the **Increment counter** is selected, the repetitions counter begins at 1 and increases till the value specified for the maximum number of repetitions is reached. The loop ends when the maximum number of repetitions for the repetitions counter is reached.

     - If the **Decrement counter** is selected, the repetitions counter begins at the value specified for the maximum number of repetitions and decreases till it reaches 1. The loop ends when the repetitions counter reaches 1.

   Note

   - It is possible to specify either the repeating condition or the count for the maximum number of repetitions, or both. If specifying both, the loop will end when one of the conditions is reached.

   - If neither is specified, it will not be possible to define the selected node as a Sequential Loop node.

4. Select exception handling from the drop-down list while the node instances are being repeatedly executed.

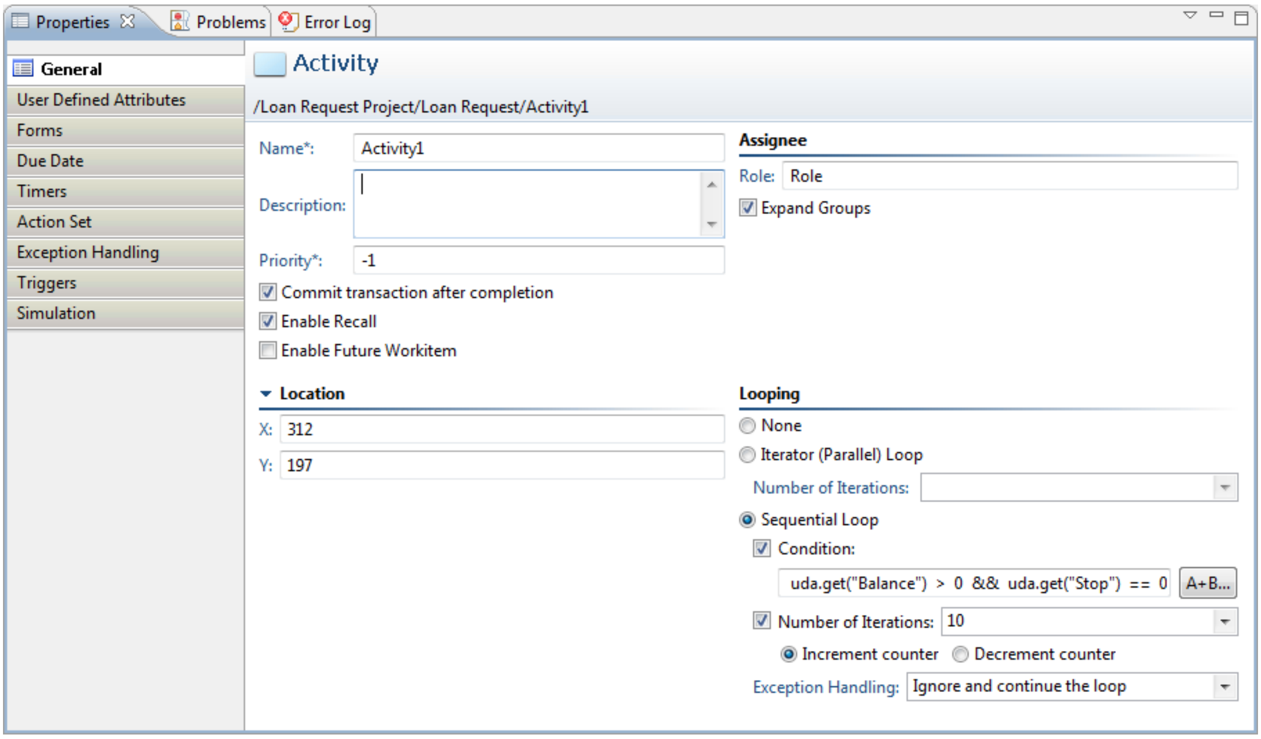
   - **None**: Exceptions are processed in accordance with the settings for handling node exceptions. For further information, see "Operating options for exception handling" covered later in this chapter.

   - **Ignore and continue the loop**: Ignore errors and continue the loop.

   - **Break the loop**: End the loop.

In the example below, both **Conditions** and **Number of Iterations** are defined for the Sequential Loop.

Figure 6.25 Sequential Loop definition



Activity

/Loan Request Project/Loan Request/Activity1

Name*: Activity1

Description:

Priority*: -1

☑ Commit transaction after completion

☑ Enable Recall

☐ Enable Future Workitem

Location

X: 312

Y: 197

Assignee

Role: Role

☑ Expand Groups

Looping

○ None

○ Iterator (Parallel) Loop

Number of Iterations:

● Sequential Loop

☑ Condition:

uda.get("Balance") > 0 && uda.get("Stop") == 0 A+B...

☑ Number of Iterations: 10

● Increment counter ○ Decrement counter

Exception Handling: Ignore and continue the loop

Note

- Only 1 (outward-facing) arrow can be drawn from the Sequential Loop node.
- It is not possible to specify settings for a due date and/or timer in the Activity Node that defined the Sequential Loop node.

Select **None** under **Looping** to disable the Sequential Loop node.

The Sequential Loop node is represented by a round arrow. Below is a representation of an Activity Node, Subprocess Node, and Compound Activity Node when defined as a Sequential Loop node.

Figure 6.26 Activity Node

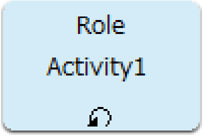


Figure 6.27 Subprocess Node

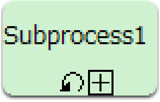
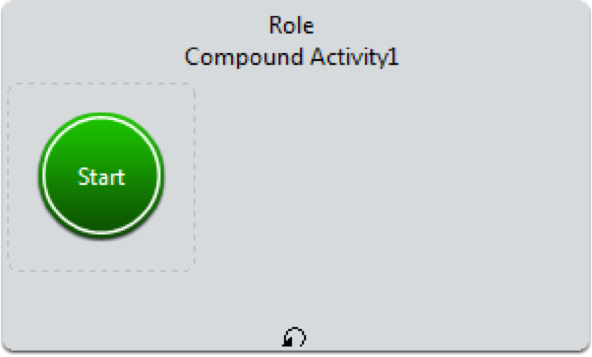


Figure 6.28 Compound Activity Node



Operating options for exception handling

The table below indicates the status of loop operations, process instances, and loop node instances when executing the specified exception processes using an Activity Node as the Sequential Loop node.

Process where the error occurred	Specified exception process	Loop operation	State of loop node instance	State of process instance
Beginning action (Role action agent)	Process exceptions based on the program	Stopped in the loop node instance where the error occurred.	Error	Error
	Ignore error, and continue loop	Continue loop in accordance with the settings for the loop condition and upper limit value of the loop count	Finished	Running
	End the loop	End the loop, and activate the next node	Interrupted	Running
Finishing action	Process exceptions based on the program	Failure of selected operations	Running	Running
	Ignore error, and continue loop	Continue loop in accordance with the settings for the loop condition and upper limit value of the loop count	Finished	Running
	End the loop	End the loop, and activate the next node	Interrupted	Running

The table below indicates the status of loop operations, process instances, and loop node instances when executing the specified exception processes using a Subprocess Node as the Sequential Loop node.

Process where the error occurred	Specified exception process	Loop operation	State of loop node instance	State of parent process	State of child process
Starting action	Process exceptions based on the program	Stopped in the loop node instance where the error occurred.	Error	Error	Not created
	Ignore error, and continue loop	Continue loop in accordance with the settings for the loop condition and upper limit value of the loop count	Finished	Running	Not created

Process where the error occurred	Specified exception process	Loop operation	State of loop node instance	State of parent process	State of child process
	End the loop	End the loop, and activate the next node	Interrupted	Running	Not created
Finishing action	Process exceptions based on the program	Stopped in the loop node instance where the error occurred.	Paused	Error	Paused
	Ignore error, and continue loop	Continue loop in accordance with the settings for the loop condition and upper limit value of the loop count	Finished	Running	Finished
	End the loop	End the loop, and activate the next node	Interrupted	Running	Finished

The table below indicates the status of loop operations, process instances, and loop node instances when executing the specified exception processes using a Compound Activity Node as the Sequential Loop node.

Process where the error occurred	Specified exception process	Loop operation	State of loop node instance	State of process instance
Beginning action (Role action)	Process exceptions based on the program	Stopped in the loop node instance where the error occurred; the instance of the Compound Activity node's child node remains as is	Error	Error
	Ignore error, and continue loop	Continue loop in accordance with the settings for the loop condition and upper limit value of the loop count; the instance of the Compound Activity node's child node remains as is	Finished	Running
	End the loop	End the loop, and activate the next node; the Compound Activity node's child node instance remains as is	Interrupted	Running
Finishing action	Process exceptions based on the program	If the Compound Activity node finishes normally after all the child node instances are complete, the loop stops at its designated loop node instance and the Compound Activity node and its valid child node is paused	Paused	Error
		An error occurs when the Compound Activity node work item is selected	Standing by at the subprocess	Running
	Ignore error, and continue loop	Continue loop in accordance with the	Finished	Running

Process where the error occurred	Specified exception process	Loop operation	State of loop node instance	State of process instance
		settings for the loop condition and upper limit value of the loop count; the state of the child node instance will be as indicated below: - If the Compound Activity Node finishes normally after all the child node instances are complete, the state of the child node instance remains the same - When the Compound Activity Node work item is selected, the valid child node instance will be in the "finished" state; the state of other child node instances remains the same		
	End the loop	Ends the loop and activates the next node; the state of the child node instance will be as indicated below: - If the Compound Activity Node finishes normally after all child node instances are complete, the state of the child node instance remains the same - When the Compound Activity Node work item is selected, the valid child node instance will be in the "finished" state; the state of other child node instances remains the same	Interrupted	Running

6.20 Using Compound Activity Node

When you want to refer to the process state in terms that are larger in scope than its individual activities, a Compound Activity Node is used. A Compound Activity Node is a container that contains various nodes and arrows.

A Compound Activity Node contains child nodes. A Compound Activity Node must have a child Start Node and at least one child Exit Node. Also, no arrow transition is allowed from child nodes inside the Compound Activity Node to the nodes outside the compound activity.

A Compound Activity Node can have any node except another compound node.

Following are the features of a Compound Activity Node:

- You can resize a Compound Activity Node by selecting it and dragging its boundaries.
- You must define the arrow of the same name as that of the Exit node defined in Compound Activity as the outgoing arrow of the Compound Activity.
- A Compound Activity Node can have only one Start node.
- Default name of a Compound Activity Node is Compound Activity 1. Suffix digit is incremented according to the creation of compound nodes in a process definition.
- Compound Activity Node does not support Iterator Node, Recall, Voting, Triggers.
- The Compound Activity Node offers no support for the Iterator (Parallel) Loop node, reclaiming, voting rules, or triggers.
- You can add the following to a Compound Activity Node:

1. UDAs: Refer to 6.18 Specifying User Defined Attributes for more details.
2. Action Set: Refer to 11.1 Using Java Actions for more details.
3. Timers: Refer to 6.22 Using Due Dates and Timers for more details.
4. Due Date: Refer to 6.22 Using Due Dates and Timers for more details.
5. Forms: Refer to Chapter 8 Using Forms for more details.
6. Priority: Refer to 6.7.3 Setting Activity Level Priority for more details.

6.20.1 Creating Compound Activity Node

To create a compound activity:

1. Open a process definition.
2. Click **Compound Activity** in the **Palette** and then click in the editor.

   A Compound Activity Node area is displayed in the editor with a child Start node in it.
3. Do any one of the following while creating a new process definition or editing an existing process definition:

   - Create child nodes within the Compound Activity Node as you normally create the nodes using the **Palette** and connect them using arrows.
   - Drag the boundaries of the Compound Activity Node so that its area covers any required existing nodes of the process definition. The nodes covered by the boundaries of the Compound Activity Node are treated as its child nodes.

   Note

   - If all the four corners of a node are moved into the Compound Activity Node area then that node is treated as a child node. You can select a child node and drag it outside the Compound Activity Node area to exclude it from the Compound Activity Node.
   - There should be at least one child Exit node inside the Compound Activity Node.
   - The number of child Exit nodes inside the Compound Activity Node and the number of outgoing arrows from the Compound Activity Node should be the same.
   - When you move the Compound Activity Node, nodes and arrows inside the Compound Activity Node also move together. But Swimlanes, Annotation and Groups do not move together. Delete operation is similar.
4. Connect the Compound Activity Node to the node outside the compound activity.

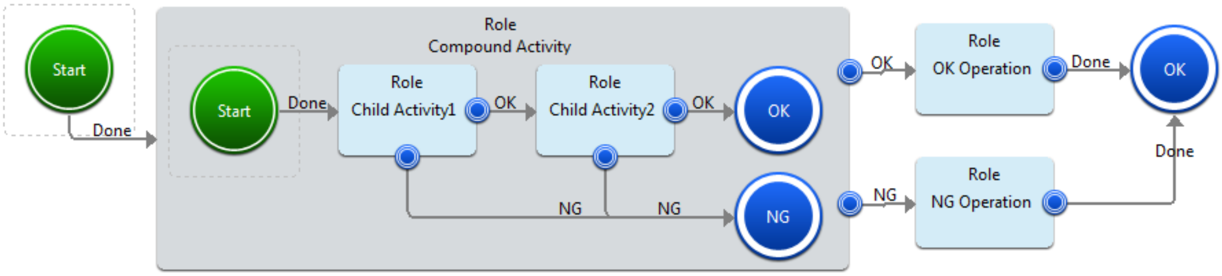
Note

The name of the child Exit node should be same as that of the arrow connecting the compound activity and the node outside the Compound Activity Node.

5. Click **Save** to save the process.

The following figure shows the Compound Activity Node:

Figure 6.29 Compound Activity Node



6.21 Setting Form Information

Form information is specified when you operate issue of a voucher of process instance or activity work in business application window (such as JSP or servlet). Form information is associated Start node, Activity node, Voting Activity node and Compound Activity node, and it is taken out business application via APS.

Explaining how to set form information.

1. Open **Property** view of the node described above.
2. Select **Form** tab.**Form**
3. Click **Add**, and set the following information.
   - **Form name**:Input any form name.
   - **Form pass** :Input the path (absolute path or relative URL)
4. If you have some form information to be set, repeat above procedure 3.

6.22 Using Due Dates and Timers

Due dates specify when an activity is due to be completed once it has become active. They also specify what will happen when the due date is reached and the activity has not been completed. For example, the activity could be escalated to other users or an email could be sent. Due dates can be defined for Process Definitions, Activity Nodes, Voting Activity Nodes, and Compound Nodes.

Timers trigger certain actions when they expire. They can be used, for example, with Delay Nodes to suspend process execution for a certain amount of time. Timers may operate only once or repeatedly. They can be defined for the process definition as a whole, for individual activities (Activity Nodes, Voting Activity Nodes, and Compound Nodes), for and Delay Nodes

When defining due dates and timers, you specify an absolute or relative time. A relative time can be based on the regular calendar or on a business calendar that counts only business hours.

The following sections explain how to define due dates and timers. They also provide instructions on using your own business calendars.

6.22.1 Defining Due Dates

When defining a due date, you can choose between the following due date types:

- **Absolute**: Sets the due date to an absolute time, for example to January 1, 2007, 00:00:00.

- **Calendar**: Sets a relative due date based on the regular calendar. The due date is calculated relative to the time the activity becomes active. The time is counted using all seven days of the week and 24 hours of the day. As such, they can expire outside the normal business hours.

- **Business**: Sets a relative due date based on a business calendar. The time is counted using only business days and hours. This ensures that activities are due only during normal business hours.

  Systemwalker Runbook Automation Studio determines business days and hours using a business calendar. Refer to section 6.22.4 Creating Your Own Business Calendars for more information on business calendars.

- **Advanced**: Sets the due date according to an expression that you specify. The expression defines an absolute or relative due date. The time can be counted using the regular calendar or a business calendar.

To define a due date:

1. Do any of the following:

   - Select the Activity Node, Voting Activity Node, or Compound Activity Node to display the **Properties** view for the nodes.

   - Click in the blank space of the editor of the Process Definition to display the **Properties** view for the Process Definition.

2. Select the **Due Date** tab.

3. Specify one of the following due date types:

   - An absolute date based on the regular calendar.

     Select **Absolute** and type the date and time.

     Note

     While setting the Absolute time (Year, Month, Day, Time), if the value of time is larger than that 24:00:00, it is converted as a day, and added to the Day field.

     Note

     To define expiration time as an expression, you can enter the Java Script expression in the **Expression** field. Refer to 6.22.2 Defining Timers for details.

   - A relative date based on the regular calendar.

     Select **Calendar** and specify after how many days and at what time the activity is due to be completed once it has become active.

     Note

     While setting the Calendar timer (Day, Time), if the value of time is larger than that 24:00:00, it is converted as a day, and added to the Day field.

     Note

     To define expiration time as an expression, you can enter the Java Script expression in the **Expression** field. Refer to 6.22.2 Defining Timers for details.

     The maximum value of the Calendar timer (Day, Time) is the following: Calendar timer (Day, Time) is converted into the millisecond. This value is not to exceed the Long.MAX_VALUE.

   - A relative date based on the business calendar.

     Select **Business** and specify after how many business days and at what business time the activity is due to be completed once it has become active.

For the business time, you have different options. You can specify an absolute time, a time relative to the current time, a time relative to opening time, or a time relative to closing time.

- An absolute or relative date specified with an expression.

  Select **Advanced** and type the expression.

Note

If you set the expression in Advanced timer and if this expression is expressible as Business timer then this expression is displayed as the Business timer.

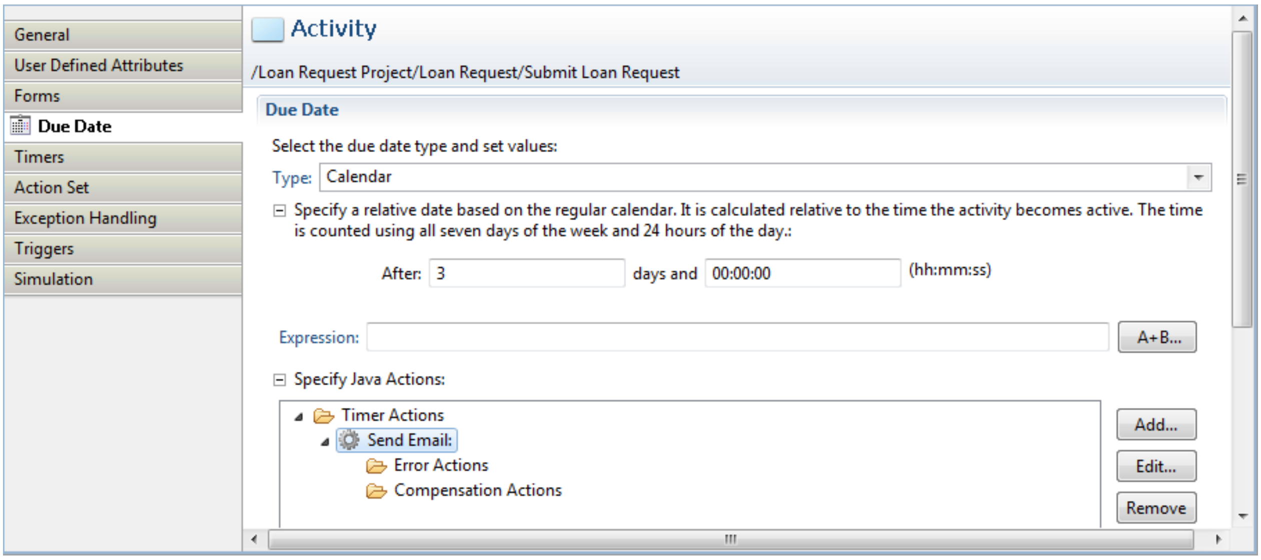
For information on date codes that you can use in the expression, refer to section 6.22.3 Time and Day Codes for Advanced Due Dates and Timers.

4. Click **Add** and select the Java Actions to be executed when the due date is reached and the activity has not been completed.

   By default, an empty **Timer Actions** folder is displayed. You can add regular Java Actions, Error Actions, and Compensation Actions to the **Action** list. For information on Java Actions, refer to section 11.1 Using Java Actions.

   The following dialog shows an example of a due date. If the activity is not completed after three business days, an email is sent.

Figure 6.30 Defining a Due Date



Activity

/Loan Request Project/Loan Request/Submit Loan Request

Due Date

Select the due date type and set values:

Type: Calendar

Specify a relative date based on the regular calendar. It is calculated relative to the time the activity becomes active. The time is counted using all seven days of the week and 24 hours of the day.:

After: 3 days and 00:00:00 (hh:mm:ss)

Expression:

Specify Java Actions:

- Timer Actions
  - Send Email:
    - Error Actions
    - Compensation Actions

6.22.2 Defining Timers

Timers trigger certain actions when they expire. They may execute only once or repeatedly. You can use timers with the following elements:

- Process Definitions

  These timers start running whenever a new process instance is created from the process definition containing the timer.

- Activity Nodes, Voting Activity Nodes, and Compound Activity Nodes

  These timers start running when the node becomes active.

- Delay Nodes

  Timers are used with Delay Nodes to suspend process execution for a certain amount of time.

  These timers start running when the Delay Node becomes active.

Note

In some environment that you use, setting time by timer and actually time which canceled timer, they may have a margin of error one minute per one hour.

Note

Before a timer starts, it calculates an expiration time based on its settings. The timer is supposed to execute at its expiration time, and it usually does. If, however, the calculated expiration time is in the past relative to the timer start time, timers that execute only once will execute upon starting, and a periodic timer will fail to execute. A timer having an expiration time before its start time is considered an error, so the timer's process instance will go into error state. Periodic Timers must have expiration times that are in the future because they represent a repeating loop that could become an infinite loop in such a case.

When defining a timer, you can choose between the following timer types:

- **Absolute**: Absolute time can either be set in the UDAs (Year, Month, Day, Time format) or to the value defined in the Java Script expression. If you set absolute time in both (Year, Month, Day, Time format and Java Script expression) and for example, the Java Script expression returns the value of 4 hours then the Absolute timer will expire after 4 hours from the node activation time. If you do not define the Java Script expression or the Java Script expression returns an invalid value, Absolute timer will expire at the absolute time defined in the Year, Month, Day, Time format (UDAs).

- **Calendar**: Sets the duration after which the timer will expire based on the regular calendar. The time is counted using all seven days of the week and 24 hours of the day. As such, they can expire outside the normal business hours. These timers start when the time set in the UDAs (days and hours) is elapsed after the node or process instance to which they are assigned becomes active. The time for Calendar timer can also be set relative to the value defined in the Java Script expression. If both the values (UDAs and Java Script expression) are set then the value returned by the Java Script expression will be taken as the reference and the value from the UDAs will be taken as the relative time. If Java Script expression is not set, timer will expire after the time set in the UDAs is elapsed from the time when the node to which it is assigned becomes active.

- **Business**: Sets the duration after which the timer will expire based on a business calendar. These timers start when the node or process instance to which they are assigned becomes active. The time is counted using only business days and hours. These timers can expire only during normal business hours.

  Note

  Systemwalker Runbook Automation Studio determines business days and hours using a business calendar. Refer to section 6.22.4 Creating Your Own Business Calendars for more information on business calendars.

- **Advanced**: Sets the timer according to an expression that you specify. The expression defines an absolute or relative timer. The time can be counted using the regular calendar or a business calendar.

To define a timer:

1. Do one of the following:

   - To define a timer for a process definition, click the empty space in the Process Definition Editor to display the **Properties** view for the process definition.

   - To define a timer for a node, select the node in the Process Definition Editor to display the **Properties** view for the node.

   Point

   You can also select the Delay Node from the Palette and drag and drop it on the relevant Activity Node or Voting Activity Node. Then you need to enter a name for the timer and define its properties later on using the **Properties** view for the node.

2. Select the **Timers** tab.

3. Click **Add** in the **All Timers** area, to add a new timer.

   This displays the **Timer Details** area where you can customize the settings for the timer.

4. By default, the new timer is named as Timer1, you can modify the name in the **Name** field.

5. To modify details for the timer, ensure that the timer is selected in the **All Timers** list.

6. Optional: Describe the purpose of the timer in the **Description** field.

7. To set the timers, do any of the following per requirements:

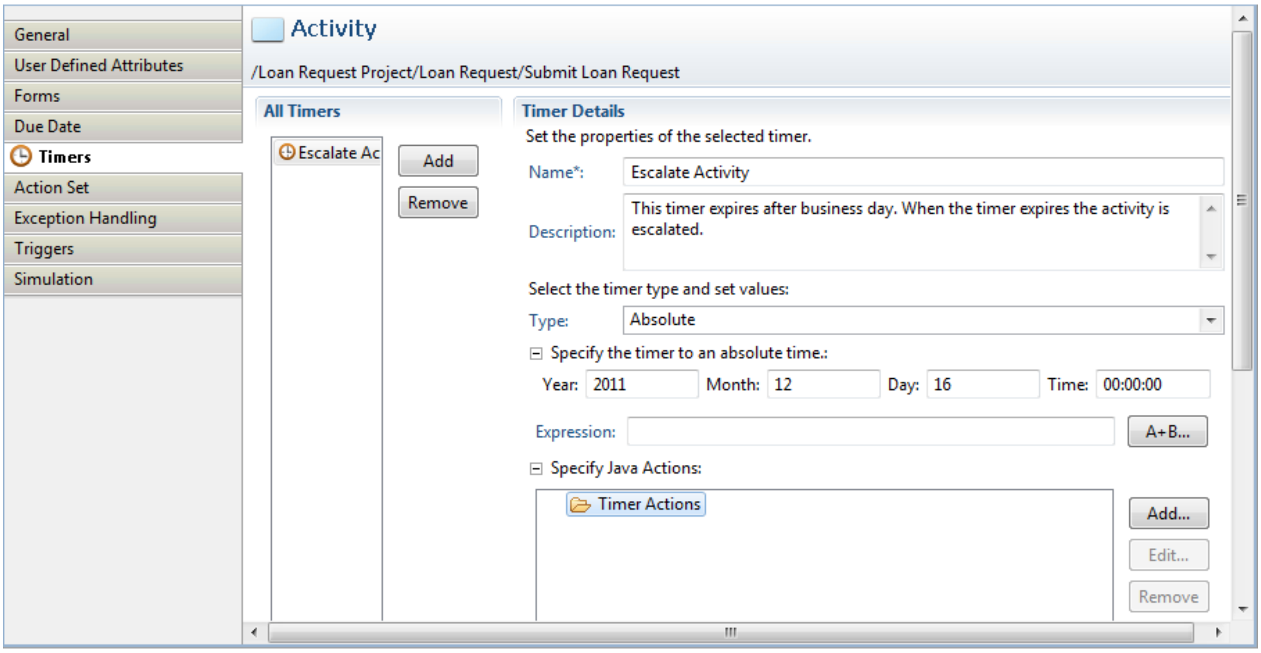
   - If you want to set the **Absolute** timer based on the regular calendar:

     1. Select **Absolute** timer in the **Type** drop-down list of the **Timer Details** area.

     2. Enter year, month, day and time in **Year**, **Month**, **Day** and **Time** fields respectively. These are the UDA values for the timer.

     3. To define expiration time as an expression, enter the Java Script expression in the Expression field. Click the **A+B...** button to define the Java Script Expression. You can either directly enter the Java Script expression in the **Expression** field or click **A+B...** button to build it. Upon clicking the **A+B...** button, **Interstage BPM Expression Builder** dialog is displayed. Build the Java Script expression using the operands and operators in this dialog For example, DateAdd(Packages.java.util.Date(), 4, \"hh\").getTime()" will expire the Absolute timer after 4 hours from the node activation time. The following figure shows **Expression** and **A+B...** fields for Absolute timer.

Figure 6.31 Defining an Absolute Timer



/Loan Request Project/Loan Request/Submit Loan Request

Set the properties of the selected timer.

Name*: Escalate Activity

Description: This timer expires after business day. When the timer expires the activity is escalated.

Select the timer type and set values:

Type: Absolute

⊟ Specify the timer to an absolute time.:

Year: 2011 Month: 12 Day: 16 Time: 00:00:00

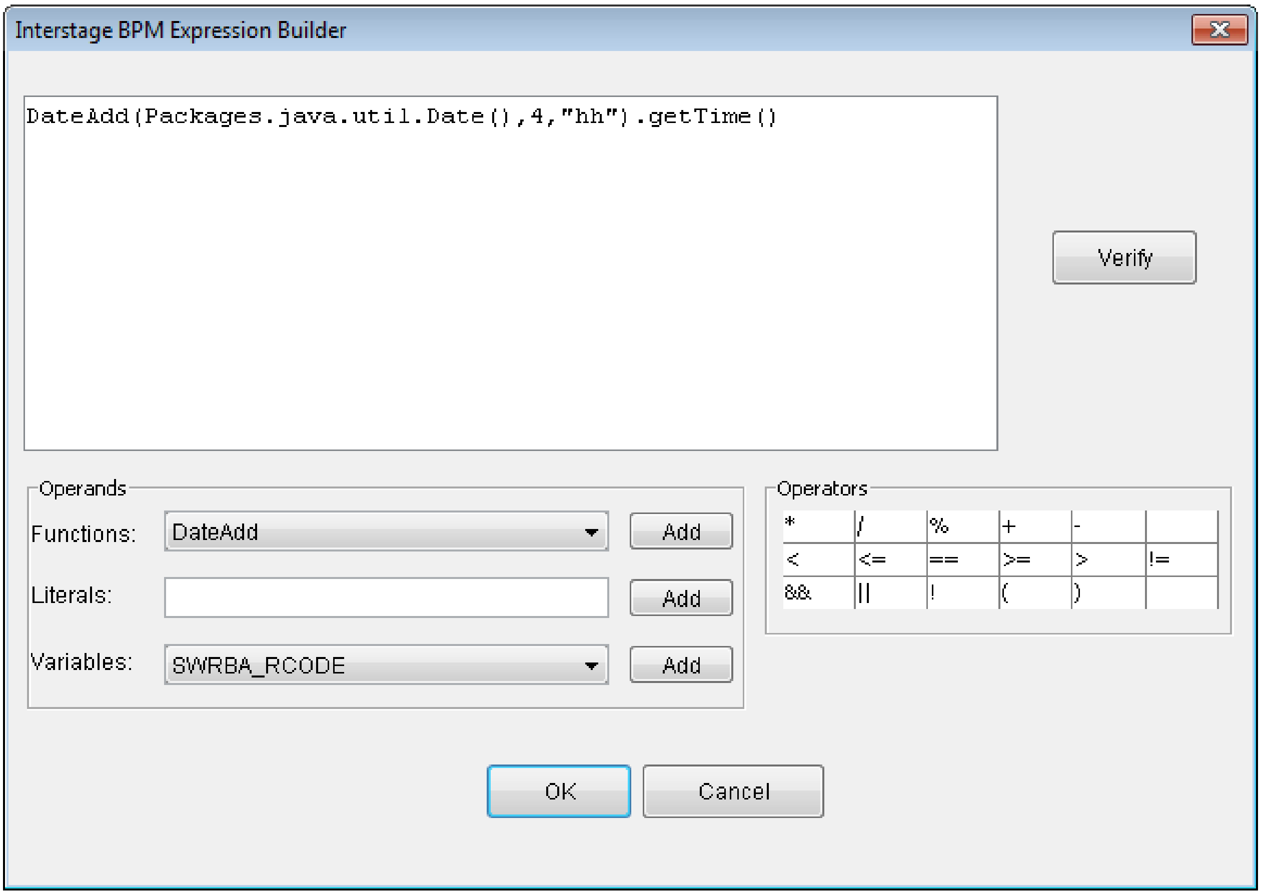
Expression:

⊟ Specify Java Actions:

Timer Actions

The following figure shows the **Interstage BPM Expression Builder**.

Figure 6.32 Building Java Script Expression



```
DateAdd(Packages.java.util.Date(),4,"hh").getTime()
```

Functions: DateAdd

Literals:

Variables: SWRBA_RCODE

Note

While setting the Absolute time (Year, Month, Day, Time), if the value of time is larger than 24:00:00, it is converted as a day, and added to the Day field. The method used to set the timer is explained in relative time from when the node or process instance was activated.

- If you want to set the **Calendar** (relative) timer based on the regular calendar:

    1. Select **Calendar** timer in the **Type** drop-down list of the **Timer Details** area.

    2. Specify after how many days and hours the timer needs to expire. For example, if you want the timer to expire after 2 days and 3 hours from the time when the node is activated or from the time defined by the reference value returned by the Java Script expression, enter 2 days and 03.00.00 in the respective fields.

    3. To set the Calendar timer to execute repeatedly, select the **Periodic** checkbox. Specify the interval at which you want the timer to expire. Periodic Calendar timer is a periodic timer based on the regular calendar. If the **Periodic** checkbox is not selected, the Calendar timer executes only once.

    4. If you want to set a reference value for the relative UDA values using the Java Script expression, define the expiration time as an expression as explained in the third step of the **Absolute** timer. Calendar timer will take the value returned by the expression as the reference value and add the UDA value to it to calculate the expiration time. For example, if you set 2 days in the Days field then uda.get("RequestDate") expires the Calendar timer after 2 days from the date returned by the UDA ("RequestDate"). The value of RequestDate is the long value shown in Date class.

Note

The maximum value of the Calendar timer (Day, Time) is the following: Calendar timer (Day, Time) is converted into the millisecond. This value is not to exceed the Long.MAX_VALUE.

- If you want to set a relative timer based on the business calendar:

    1. Select **Business** timer in the **Type** drop-down list of the **Timer Details** area.

    2. Specify after how many business days and at what business time the timer is to expire. For the business time, you have different options. You can specify an absolute time, a time relative to the current time, a time relative to opening time, or a time relative to closing time.

    3. To set the Business timer to execute repeatedly, select the **Periodic** checkbox. Specify the interval at which you want the timer to expire. Business timer is a periodic timer based on the business calendar. If the **Periodic** checkbox is not selected, the Business timer executes only once.

- If you want to set an absolute or relative timer specified with an expression:

    1. Select **Advanced** timer in the **Type** drop-down list of the **Timer Details** area and type the expression in the **Set Expression** field. For information on date codes that you can use in the expression, refer to section 6.22.3 Time and Day Codes for Advanced Due Dates and Timers.

    2. To set the Advanced timer to execute repeatedly, select **Advanced** and select the **Periodic** checkbox. Specify the interval at which you want the timer to expire. If the **Periodic** checkbox is not selected, the Advanced timer executes only once.

Note

If you set the expression in Advanced timer and if this expression is expressible as Business timer then this expression is displayed as the Business timer.

Note

Periodic timers are always relative to a given event. The first operation of a periodic timer is relative to the time when the process instance or the node becomes active. Subsequent operations are relative to the last operation.

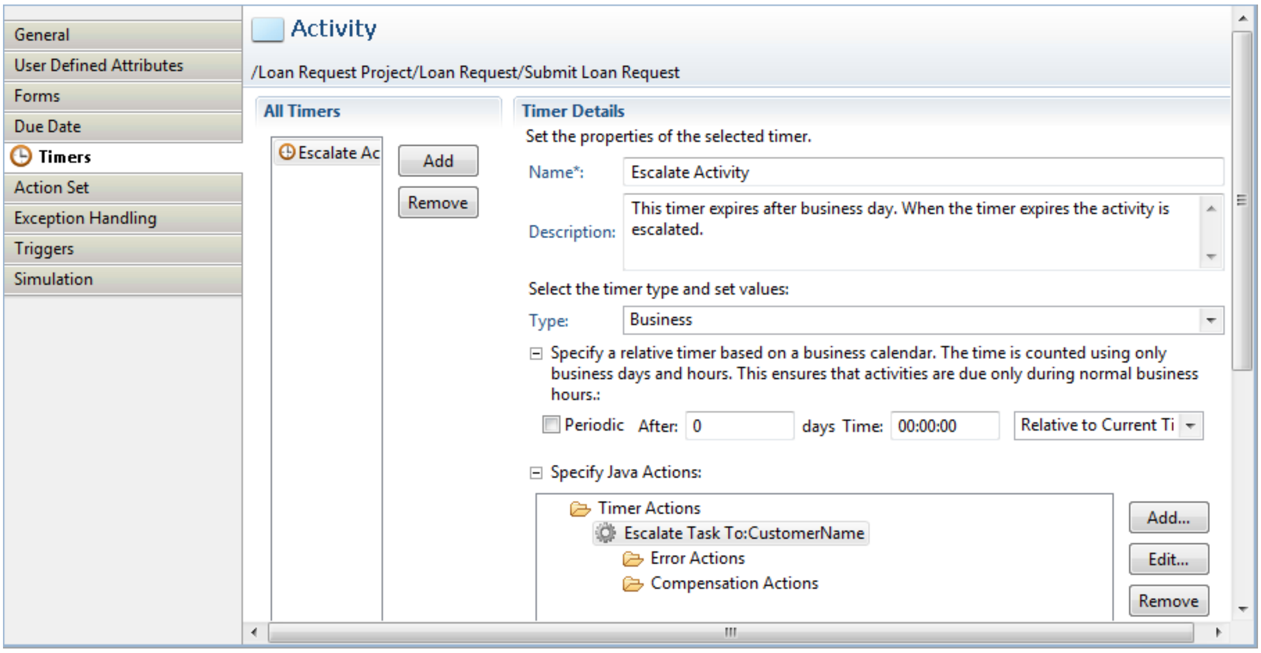
8. Click **Add** and select the Java Actions to be executed when the timer is expired.

    You can add regular Java Actions, Error Actions, and Compensation Actions to the **Action** list.

    For more information on Java Actions, refer to section 11.1 Using Java Actions.

The following figure shows a timer that expires after one business day. When the timer expires, the activity is escalated.

Figure 6.33 Defining a Timer



/Loan Request Project/Loan Request/Submit Loan Request

Set the properties of the selected timer.

Name*: Escalate Activity

Description: This timer expires after business day. When the timer expires the activity is escalated.

Select the timer type and set values:

Type: Business

⊟ Specify a relative timer based on a business calendar. The time is counted using only business days and hours. This ensures that activities are due only during normal business hours.:

□ Periodic After: 0 days Time: 00:00:00 Relative to Current Ti

⊟ Specify Java Actions:

- Timer Actions
  - Escalate Task To:CustomerName
  - Error Actions
  - Compensation Actions

For each timer that you define, one or multiple User Defined Attributes (UDAs) are created. The UDA

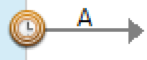
names identify the timer and the timer action.

For example, a typical UDA name might be __atmr_publication_time. The prefix __atmr identifies this as a timer, the middle term publication contains the name of the timer, and the suffix time identifies the function of the UDA.

These UDAs can be modified on a form, through a Java Action, or through a JavaScript.

When a Make Choice Java Action has been defined for a timer and an arrow name has been defined as the choice item for this Java Action, the following symbol is added to this arrow connecting the Activity Node or Voting Activity Node or Compound Activity Node with another node:

Figure 6.34 Timer symbol



6.22.3 Time and Day Codes for Advanced Due Dates and Timers

Formulas can be set up in the Detail section for due dates and timers.

Example UC(Japan);BD(1);OT(00)

For the detail UC, Refer to 6.22.6 Assigning Business Calendars to Due Dates or Timers

Time Codes

Code	Meaning	Example
AT	Sets the absolute time of the day.	AT(16:30:00): 4:30pm on that day.
CT	Sets the business time relative to the closing time of the day. Allowed values: 00 or negative hours. Typically you will use negative hours with closing time in order to calculate a relative time before closing time.	CT(00): Closing time. CT(-02:00:00): 2 hrs before the closing time.

Code	Meaning	Example
OT	Sets the business time relative to the opening time of the day. Allowed values: 00 or positive hours.	OT(00): At the opening time. OT(02:00:00): 2 hrs after the opening time.
BT	Sets the business time relative to the current time of the day.	BT(04:30:00): After 4 & 1/2 business hours from the current time. BT(-02:00): 2 business hours earlier. BT(00): Use this to search forward to the next business time, without changing the time if the current time is not a business time. You may need this if different business days have different hours. BT(-00): Use this to search backward to the last previous business time. Has no effect if the current time is already during business time.

Day Codes

Code	Meaning	Example
BD	Sets a business day.	BD(4): Four business days from today. BD(0): Same day if it is a business day, else the next day. BD(-0): Same day if it is a business day, else the previous day.
RD	Sets a relative day from the current day.	RD(7): After one week. RD(-1): One day earlier.
WD	Sets a day of the week. Allowed values: 1 to 7.	WD(1): Sunday of that week. WD(7): Saturday of that week. WD(8): Sunday of next week
WN	Sets the next weekday after today. Allowed values: 1 to 7.	WN(1): The next Sunday after today. WN(7): The next Saturday after today.
RM	Sets a relative month in the future. If the month does not have enough days to be the same day, the day will be the last day of the month.	RM(3): After 3 months
DM	Sets an exact day of the month. Allowed values: Any number except 0.	DM(1): The first day of the month. DM(-1): The last day of the month.
BM	Sets an exact business day of the month. Allowed values: Any number except 0.	BM(1): The first business day of the month. BM(-1): The last business day of the month.
DY	Sets a day of the year. Allowed values: Any number except 0.	DY(1): The first day of the year. DY(-1): The last day of the year.
BY	Sets a business day of the year. Allowed values: Any number except 0.	BY(1): The first business day of the year. BY(-1): The last business day of the year.

6.22.4 Creating Your Own Business Calendars

When defining due dates and timers, you can use a business calendar instead of a regular calendar. A business calendar defines the business hours and days of an organization. Using a business calendar ensures that timers expire during business hours only. The same applies for due dates; business calendars ensure that activities are due during business hours only.

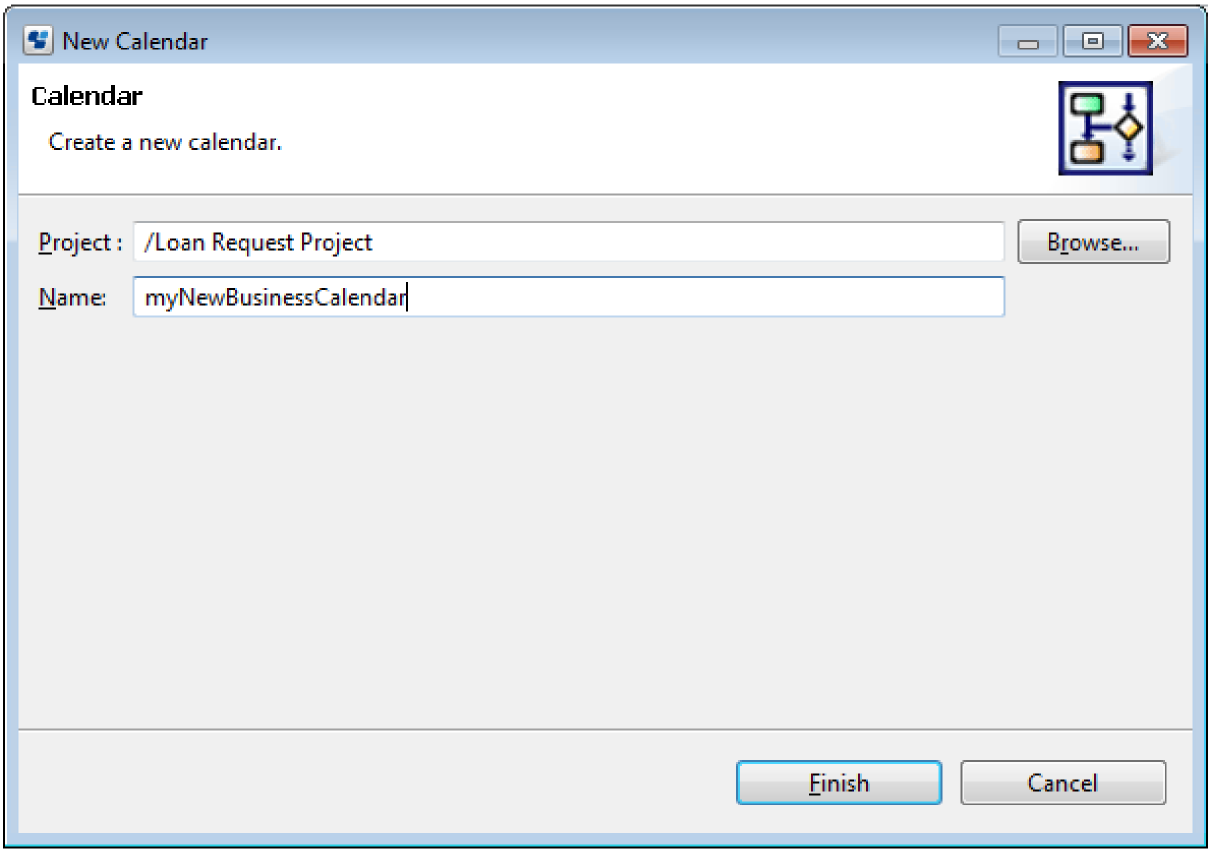
Systemwalker Runbook Automation Studio provides a fully functional default business calendar. You can modify the default business calendar or you can create your own business calendars. You may create as many business calendars as necessary to meet the needs of your organization. For example, if your organization has divisions in multiple states that have different business hours and public holidays, you create a business calendar for each state.

To create a business calendar:

1. In your Workflow Application project, right click the Calendar folder and select **New** >> **Calendar**.

   The **New Calendar** dialog is displayed. The **Project** field automatically shows the name of the Workflow Application project for which you want to create the business calendar.

   Figure 6.35 Displaying the New Calendar dialog

   Project : /Loan Request Project

   Name: myNewBusinessCalendar

2. If you want to select a different project, click **Browse**.

   The **Folder Selection** dialog is displayed. Select a project directory for the new calendar file. The default name of your calendar file is <your name>.cal. The .cal extension indicates that this is a business calendar. Calendars are like properties or .ini files that specify business days and hours.

3. Select the project where the new calendar file is to be saved, and click **OK**.

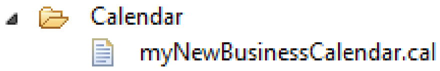
   If the file name already exists, type in a new name to avoid a file name conflict.

4. In the **Name** field, type in a name for the new calendar file, and click **Finish**.

   The new calendar (.cal) file is automatically stored in the **Calendar** folder of the selected project.

The following figure shows the location of the new myNewBusinessCalendar.cal file:

Figure 6.36 Location of a new calendar file



The new file automatically opens in the text editor you have specified for opening .cal files. The default calendar file is stored in the C:\Fujitsu\Systemwalker\SWRBA_Studio\ibpm\Data\calendar directory. The file looks as follows:

```
EVERYDAY=8:30,18:00;
SAT=;
SUN=;
2009/01/01=;
2010/01/01=;
2011/01/01=;
2012/01/01=;
2013/01/01=;
2014/01/01=;
2015/01/01=;
2016/01/01=;
CALENDAR_BEGIN=2009/01/01;
CALENDAR_END=2018/12/31;
```

5. Define your business calendar.

   You can use the default calendar file as an example. For a detailed explanation of the business calendar format, refer to the "6.22.4.1 Business Calendar Format".

   Note

   You can only save your changes using the **Save** option from the file menu. You cannot select the **Save As** option for saving new resource files.

   Once you have created a business calendar, you can assign it to process definitions or to particular timers.

6.22.4.1 Business Calendar Format

The format for business calendars is shown below.

For business calendars, use the following parameters to specify business hours.

- EVERYDAY
  Specifies the default business hours (for days and dates for which no special settings are specified).

- <*day*>
  Specifies settings for particular days of the week for which the business hours are not normally the same as the default business hours (such as excluding weekends from business hours, for example).

- <*date*>
  Specifies settings for particular dates throughout the year for which the business hours are not normally the same as the default business hours (such as excluding holidays from business hours, for example).

The priority for these parameters is as follows:

date > *day* > EVERYDAY

The following example shows how business hours can be set up.

```
EVERYDAY=9:00,17:00;
FRI=9:00,14:00;
2007/06/08=09:30,13:00;
```

In this example, on Friday June 8, 2007 business hours run from 9:30 AM until 1:00 PM. On all other Fridays, business hours are from 9:00 AM until 2:00 PM, and on all other days business hours are from 9:00 AM until 5:00 PM.

The following section provides a more detailed explanation of each of the parameters that can be used in the calendar file.

EVERYDAY

This parameter is mandatory. It defines the default business hours applied to all days (except for days where specific settings have been specified using the <day> or <date> parameters).

Example: EVERYDAY=9:00,17:00;

<day>

This parameter defines the default business hours applied to particular days. These settings take priority over the settings specified with the EVERYDAY parameter.

Example: The business hours for a particular day of the week (SUN, MON, TUE, WED, THU, FRI, or SAT) can be specified as below.

FRI=9:00,16:00;

If no hours are specified, that day will be treated as a non-business day.

This parameter is normally used to exclude Saturday and Sunday from the weekly business calendar. To do this, set the "null" value for Saturday and Sunday. The following example shows how to exclude Saturday and Sunday from the weekly business calendar.

SAT=; SUN=;

<date>

This parameter defines the business hours for particular dates. Settings specified with this parameter take priority over the settings specified with the EVERYDAY parameter and the <*day*> parameter.

Format: yyyy/mm/dd (year/month/day)

This parameter is normally used to exclude holidays from the annual business calendar.

Example: 2007/01/01=;

This parameter can also be used to specify special business hours, such as longer lunch breaks because of special occasions.

Example: 2007/12/01=9:00,12:00;15:30,17:00;

DST

This parameter is used to make adjustments for daylight savings time (summer time).

Format: <*date*>=DST(<*number of hours to adjust*>)

For <*number of hours to adjust*>, a value between 0 and 4 can be specified.

The following example shows how to move business hours forward in spring and then return to normal in autumn.

2006/04/20=DST(1);

2006/10/19=DST(0);

It is not possible to combine "DST(<*number of hours to adjust*>)" with a time specification. For example, specifications such as "2006/10/20=DST(1);9:00,12:00;13:00,17:30" are not correct.

CALENDAR_END

This parameter is mandatory. It defines the last date for which the calendar is valid. Once the date specified by this parameter has passed, the calendar can no longer be used. If the end date for a timer has been specified as a date later than the end date for the calendar, an array out of range error will occur, and the timer will not function properly.

Default value: CALENDAR_END=2010/12/31;

CALENDAR_BEGIN

This parameter is mandatory. IT defines the date when the calendar is to be issued. The calendar cannot be used before the date specified by this parameter. If the end date for a timer has been specified as a date earlier than the start date for the calendar, an array out of range error will occur, and the timer will not function properly.

Default value: CALENDAR_BEGIN=2003/01/01;

Note

The maximum period that a calendar can be valid is 10 years. This means that the dates specified for the start date (CALENDAR_BEGIN) and end date (CALENDAR_END) for the business calendar cannot be separated by more than 10 years.

TIMEZONE

This parameter is mandatory. It defines the time zone for the client's location relative to Greenwich Mean Time (GMT).

Example: Japanese Standard Time (JST) is nine hours ahead of GMT, so specify "TIMEZONE=+9:00;" for this parameter.

6.22.5 Assigning Business Calendars to Process Definitions

You can assign different business calendars for every process definition.

Prerequisites:

- An independent business calendar has been created.

To assign a business calendar to a process definition:

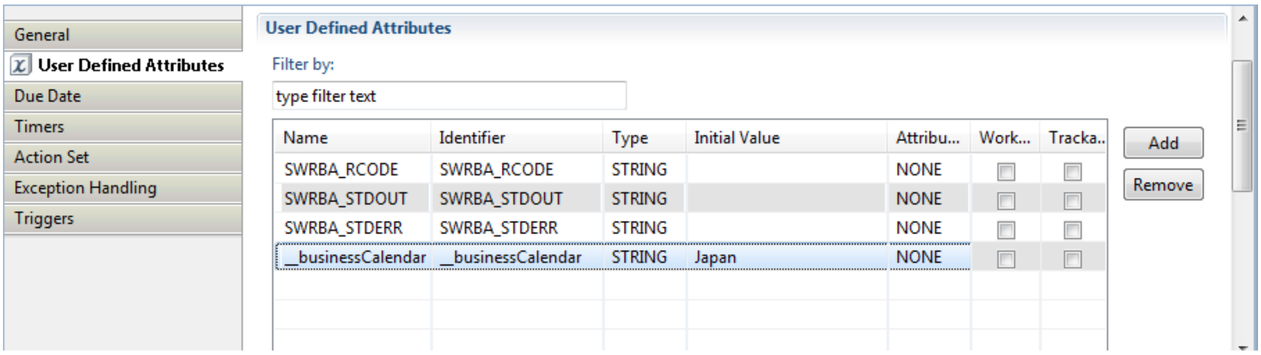
1. Click the empty space in the Process Definition Editor to display the Properties view for the process definition.

2. Select the **User Defined Attributes** tab.

3. Add the User Defined Attribute (UDA) __businessCalendar of type STRING to the process definition. Specify the name of your business calendar without the .cal extension as its value.

   In the following example, a business calendar called German.cal is assigned to the process definition:

Figure 6.37 Assigning a Business Calendar



Name	Identifier	Type	Initial Value	Attribu...	Work...	Tracka..
SWRBA_RCODE	SWRBA_RCODE	STRING		NONE	□	□
SWRBA_STDOUT	SWRBA_STDOUT	STRING		NONE	□	□
SWRBA_STDERR	SWRBA_STDERR	STRING		NONE	□	□
__businessCalendar	__businessCalendar	STRING	Japan	NONE	□	□

6.22.6 Assigning Business Calendars to Due Dates or Timers

Prerequisite: An independent business calendar has been created.

You can assign a business calendar to a particular due date or to a particular timer. The due date or timer is then calculated based on this business calendar.

To assign a business calendar to a timer or due date:

1. Define an advanced timer or an advanced due date. Refer to sections 6.22.1 Defining Due Dates or 6.22.2 Defining Timers for instructions.

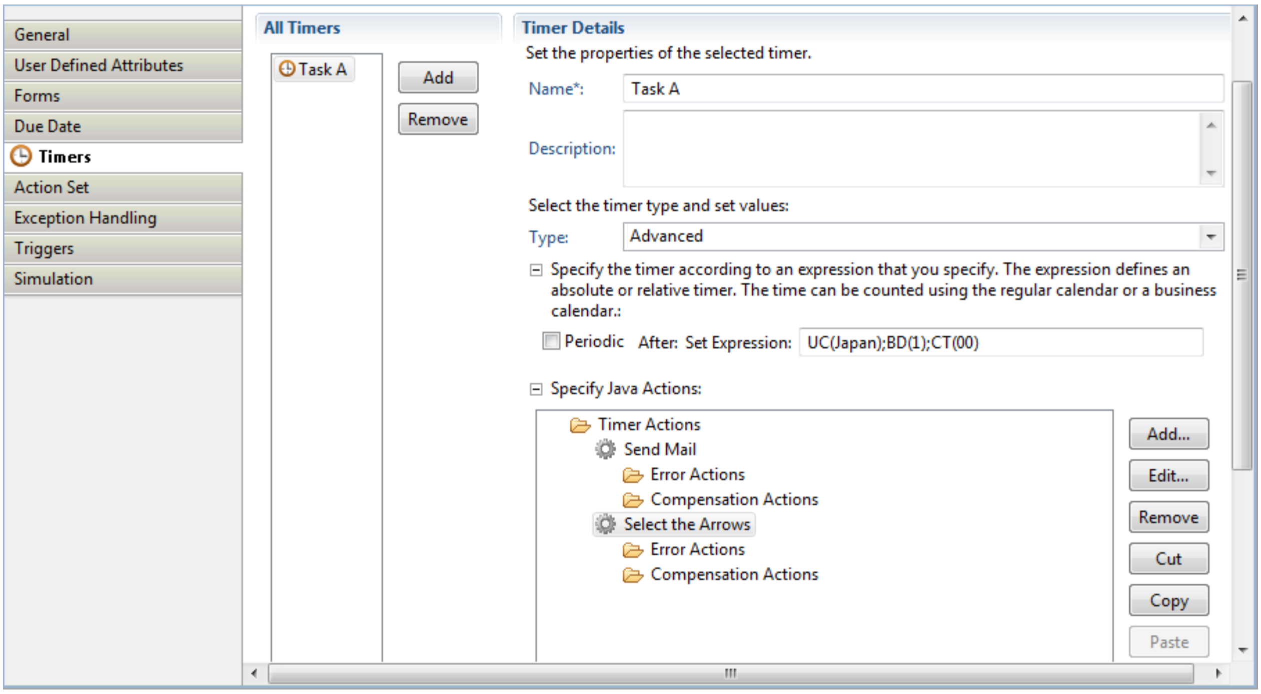
2. In the **Set Expression** field, enter the following expression:

   UC(<business calendar>.cal);

   For <business calendar>, specify the name of your business calendar without the .cal extension.

   In the following example, a business calendar called german japan is assigned to a timer:

Figure 6.38 Assigning a Business Calendar to a Timer



Set the properties of the selected timer.

Name*: Task A

Description:

Select the timer type and set values:

Type: Advanced

Specify the timer according to an expression that you specify. The expression defines an absolute or relative timer. The time can be counted using the regular calendar or a business calendar.:

□ Periodic After: Set Expression: UC(Japan);BD(1);CT(00)

Specify Java Actions:

6.23 Defining Voting Rules

This section explains the definition of the voting rules.

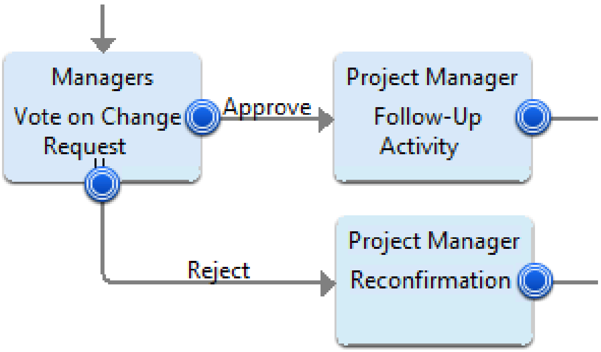
Prerequisites:

- You have added a Voting Activity Node to your process definition.
- The Voting Activity Node has outgoing arrows.

A Voting Activity Node allows users to work on an activity in collaboration with one another. All users can make their own choice (or vote). All of their votes are polled. The winning vote, represented by one of the outgoing arrows, is determined by voting rules.

The following figure shows an example, where Managers can give their vote on a change request.

Figure 6.39 Voting on a Change Request



You define a voting rule for every outgoing arrow. The following rule types are available:

- **Majority Rule**

  If this rule is assigned to a choice, a majority of votes for that choice make it the winning choice.

  For example, if the majority rule were assigned to a choice called "Approve", a majority of votes for the "Approve" choice would make that the winning choice. The process instance would proceed along the "Approve" arrow to the next activity.

- **Percentage Rule**

  If this rule is assigned to a choice, the specified percentage of votes for that choice makes it the winning choice.

  For example, if a 50% rule is assigned to a choice called "Approve", 50% of the total number of votes for the "Approve" choice would make that the winning choice.

- **"Number of" Rule**

  If this rule is assigned to a choice, the specified number of votes for that choice makes it the winning

  choice.

  For example, if a "1" is assigned to a choice called "Reject", one vote for "Reject" would make

  that the winning choice.

When defining voting rules, you also choose a default rule. The default rule is chosen if none of the rules apply.

To define voting rules:

1. Select the Voting Activity Node to display the **Properties** view for it.
2. Select the **Voting Rules** tab.

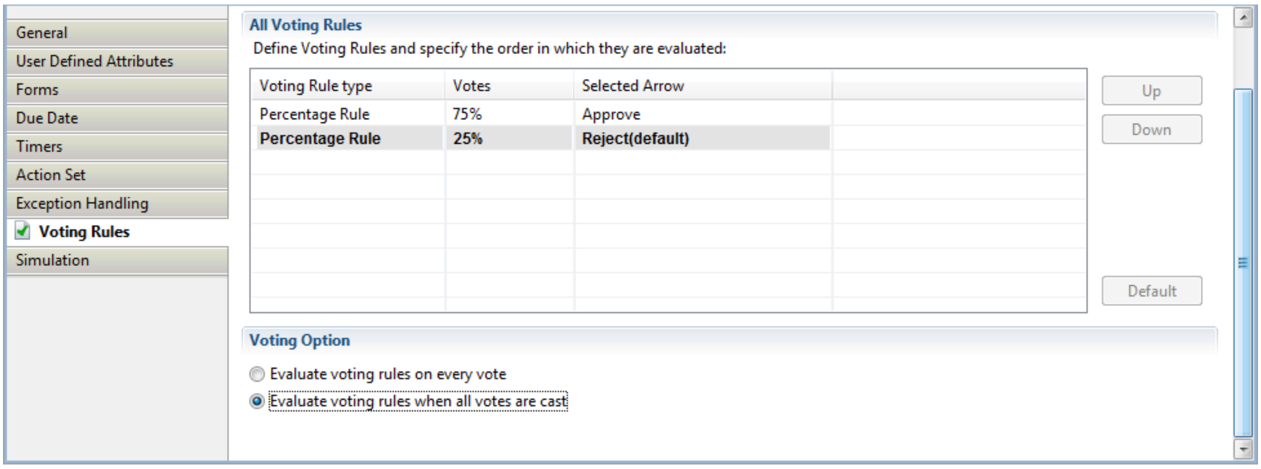
   The **All Voting Rules** area displays all of the Voting Activity Node's outgoing arrows.
3. For each arrow, define a voting rule.
4. Do one of the following:
   - If you want to complete the activity as soon as a voting rule is satisfied, select **Evaluate voting rules on every vote**.
   - If you want to ensure that everyone gets a chance to vote, select **Evaluate voting rules when all votes are cast**.
5. If you want to define a default voting rule, select the voting rule and click **Default**. If it has already been selected as the default, then the **Default** button is disabled.
6. You can rearrange the order in which the voting rules are evaluated by highlighting a rule and clicking **Up** or **Down**.

   All outgoing arrows are checked in the sequence in which they appear in this view. The first matching rule is chosen. If no valid rule is found, the arrow defined as default is chosen - even if the rule does not apply.

Examples

The following figure shows an example for percentage rules. A decision is approved if at least 75 % of the managers vote for it. The setting **Evaluate voting rules when all votes are cast** makes sure that all managers give their vote.

Figure 6.40 Percentage Rules



All Voting Rules

Define Voting Rules and specify the order in which they are evaluated:

Voting Rule type	Votes	Selected Arrow
Percentage Rule	75%	Approve
Percentage Rule	**25%**	**Reject(default)**

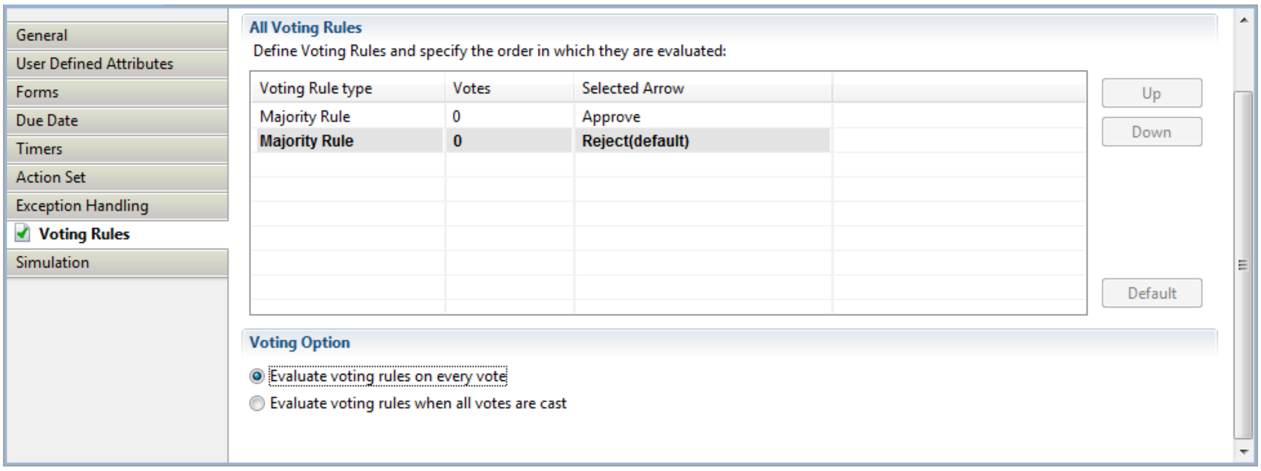
Voting Option

○ Evaluate voting rules on every vote

● Evaluate voting rules when all votes are cast

The next example shows the usage of majority rules. A decision is approved or rejected, if the majority of users vote for or against the decision. If an equal number of users vote for and against the decision, the default voting rule is used. In this case, the default is to reject the decision.

Figure 6.41 Majority Rules



All Voting Rules

Define Voting Rules and specify the order in which they are evaluated:

Voting Rule type	Votes	Selected Arrow
Majority Rule	0	Approve
Majority Rule	**0**	**Reject(default)**

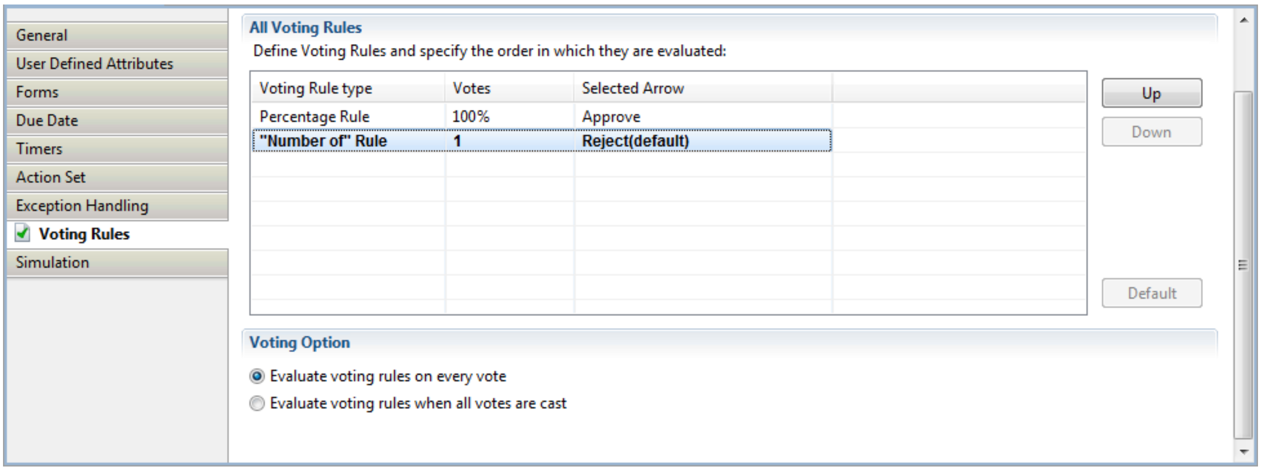
Voting Option

● Evaluate voting rules on every vote

○ Evaluate voting rules when all votes are cast

In the following example, a decision is only approved if all users vote for it. As soon as one user votes against it, the decision is rejected, and the process proceeds to the next activity.

Figure 6.42 Number of Rule



All Voting Rules

Define Voting Rules and specify the order in which they are evaluated:

Voting Rule type	Votes	Selected Arrow
Percentage Rule	100%	Approve
"Number of" Rule	**1**	**Reject(default)**

Voting Option

● Evaluate voting rules on every vote

○ Evaluate voting rules when all votes are cast

6.24 Defining Conditions

This section explains the definition of the conditions.

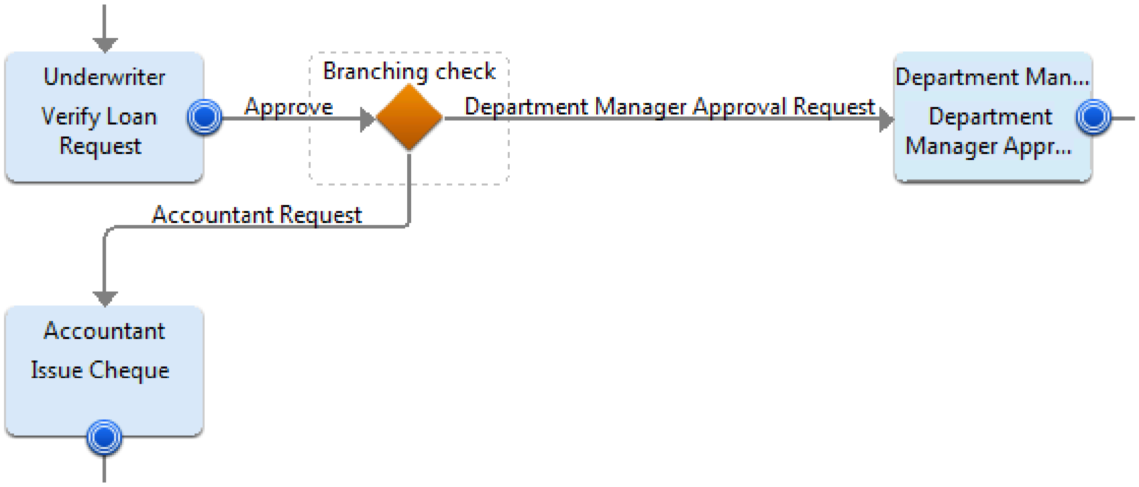
Prerequisites:

- You have added a Conditional Node to your process definition.
- The Conditional Node has outgoing arrows.
- You have specified User Defined Attributes (UDAs) that you want to evaluate.

A Conditional Node represents a step where the process proceeds in one of many possible directions. The outgoing arrows of the node represent the directions in which the process can proceed. The decision which direction to take depends on the value of a UDA.

The following figure shows parts of a purchasing process. Team members can order goods directly if the price is below a certain limit. Otherwise, the approval of the project manager is required.

Figure 6.43 Using Conditional Nodes



When defining conditions, you can also define a default arrow. The default arrow is chosen if none of the conditions apply. It is also chosen, if you do not enter values for the Conditional Node to evaluate.

To define conditions:

1. Select the Conditional Node to display the **Properties** view for it.
2. Select the **Decision** tab.
3. In the **UDA to Evaluate** area, select the UDA data type from the **UDA name** drop-down list.

   If you select a UDA data type as XML, the **Properties** view displays an additional **XPath** field.

   Note

   If you select a non-XML UDA type, the XPath field will be disabled.

4. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the **Edit** button that is displayed next to the **XPath** drop-down list. **XPath Editor** dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

   Note

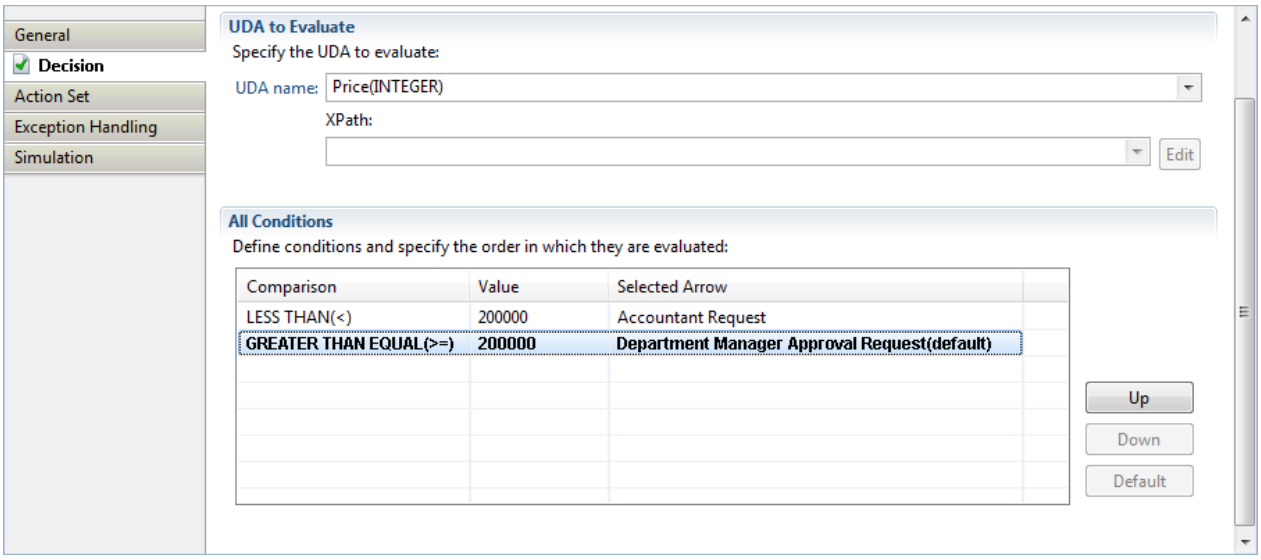
   The XPath expressions related to the selected XML UDA are displayed in the XPathdrop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

Note

XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

5. For each arrow that originates from the Conditional Node, specify the criteria to compare with the value of the UDA.

Figure 6.44 Defining Conditions



UDA to Evaluate

Specify the UDA to evaluate:

UDA name: Price(INTEGER)

XPath:

All Conditions

Define conditions and specify the order in which they are evaluated:

Comparison	Value	Selected Arrow
LESS THAN(<)	200000	Accountant Request
GREATER THAN EQUAL(>=)	200000	Department Manager Approval Request(default)

6. If you want to define a default condition, select the condition in the **All Conditions** area and click **Default**. If it has already been set as the default condition, the **Default** button is disabled.

7. You can rearrange the order in which the conditions are evaluated by highlighting the condition and clicking **Up** or **Down**.

   All outgoing arrows are checked in the sequence in which they appear in this view. The first matching arrow is chosen. If no valid arrow is found, the arrow defined as default is chosen - even if the condition does not apply.

If you want to specify advanced conditions, use a Complex Conditional Node instead of a Conditional Node.

6.25 Defining Complex Conditions

Prerequisites:

- You have added a Complex Conditional Node to your process definition.
- The Complex Conditional Node has outgoing arrows.
- You have specified User Defined Attributes (UDAs) that you want to evaluate.

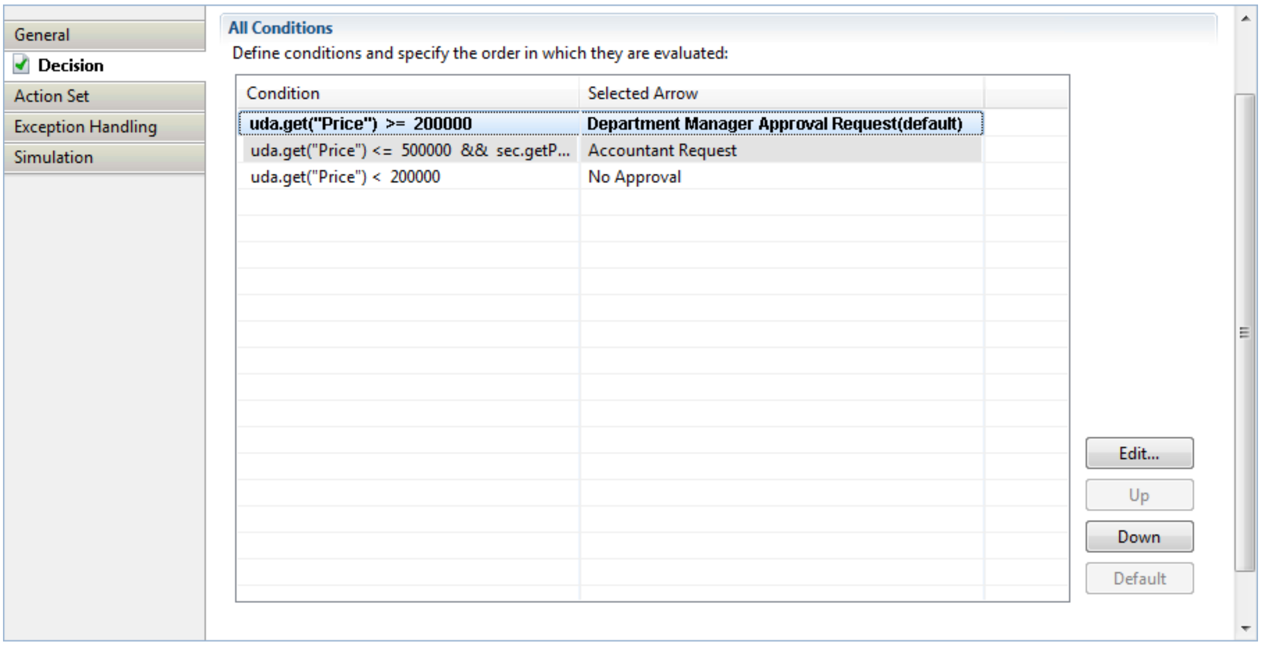
When using Conditional Nodes, you can only specify simple conditions; that is, you can only specify if a UDA is equal, less or greater than a given value. Complex Conditional Nodes give you more flexibility as they allow you to specify conditions using JavaScript expressions.

To define complex conditions:

1. Select the Complex Conditional Node to display the **Properties** view for it.

2. Select the **Decisions** tab.

Figure 6.45 Defining Complex Conditions



All Conditions

Define conditions and specify the order in which they are evaluated:

Condition	Selected Arrow
uda.get("Price") >= 200000	Department Manager Approval Request(default)
uda.get("Price") <= 500000 && sec.getP...	Accountant Request
uda.get("Price") < 200000	No Approval

3. For each arrow that originates from the Complex Conditional Node, specify a JavaScript expression. You can type the JavaScript expression directly or build it using the Expression Builder. For details, refer to section 11.14 Defining JavaScript Expressions.

   If the JavaScript expression for an arrow evaluates to true, the arrow is chosen.

4. If you want to define a default condition, select the condition in the **All Conditions** area and click **Default**. If it has already been set as the default condition, the **Default** button is disabled.

5. You can rearrange the order in which the conditions are evaluated by highlighting the condition and clicking the **Up** or **Down** button.

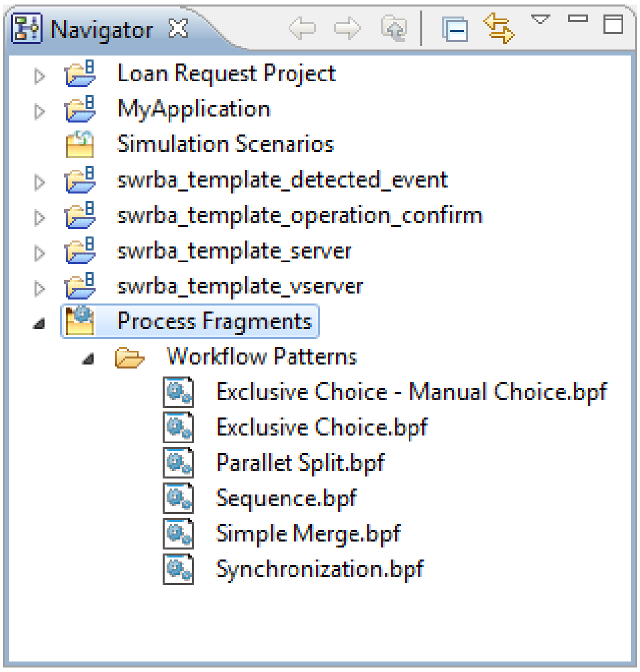
   All outgoing arrows are checked in the sequence in which they appear in this view. The first matching arrow is chosen. If no valid arrow is found, the arrow defined as default is chosen - even if the condition does not apply.

Chapter 7 Using Process Fragments

Process fragments are pre-defined, reusable fragments of a process definition that can be added to process definitions. You can use process fragments to quickly define processes.

Process fragments are displayed as part of the **Navigator** view. When a new process fragment is created it appears as a Business Process Fragment (.bpf) file in the **Process Fragments** project folder in the **Navigator** view. When you open a process fragment it is displayed in the Process Definition editor. You can add/remove nodes and arrows in this editor

Figure 7.1 Process Fragments Navigator View



Workflow Patterns folder is a default folder within the **Process Fragments** project folder. This folder contains 6 pre-built patterns to use in process definitions. You can define other folders within the **Process Fragments** project folder to contain the process fragments added by you.

Refer to the following sections for more information on process fragments

- 7.1 Creating Process Fragments
- 7.2 Adding Process Fragments to Process Definitions

7.1 Creating Process Fragments

To add new process fragments:

1. On an active Process Definition Editor, select the node and/or arrows that you want to save as reusable process fragments.

   Note

   - If you have not selected any nodes or arrows, the nodes and the arrows copied by the previous operation are used.

2. Right-click the Process Fragments folder and select **New** >> **Process Fragment** from the menu or select **New** >> **Process Fragment** from the **File** menu.
   The **New Process Fragment** dialog box is displayed.

3. Enter the folder name where you want to save the process fragment. You can also click Browse and navigate to the location where you want to save the process fragment.

4. Enter a name for the process fragment in the **Name** field.

5. Enter a description in the **Description** field.

6. Click **Finish**.

The process fragments is displayed in the Process Definition Editor with the elements that you selected in step 1 . You can edit the process fragment in this editor and save it.

Note

Unlike Process Definitions, process fragments need not have a Start node.

The Business Process Fragment (.bpf) file is displayed in the **Navigator** view in the Process Fragments project folder.

7.2 Adding Process Fragments to Process Definitions

Pre-requisites: Ensure that the process fragment that you want to add is available in the Process Fragments folder.

To add process fragments to Process Definitions:

1. Open the Process Definition that you want to add the process fragments to, in the Process Definition Editor.

2. Click the appropriate process fragment in the **Navigator** view.

3. Drag and drop the process fragment to the Process Definition editor.

   The nodes and arrows that are present in the process fragment are added to the Process Definition.

   The upper left point of the process fragment as defined while creating the process fragment is placed in the editor at the location dropped.

The following points should be remembered while adding a process fragment to a Process Definition.

- If the process fragment has a Start Node, a confirmation dialog is displayed when you add it to the Process Definition. The confirmation dialog confirms that you want to replace the Start Node with the one defined in the process fragment. If you replace the Start Node with the one in the process fragment, outgoing arrows defined in the target process definition re-attach the Start Node from the process fragment.

- A swimlane has the style to place the swimlane title on the left-hand side or on the top of a swimlane. If the process definition to which you want to add a process fragment has selected the style different from the process fragment for the swimlane title, the style of swimlane defined in the process fragment is changed to the same style as that process definition.

- When you add a process fragment to a Process Definition, UDAs used for nodes defined by process fragment are copied and pasted to the process definition, except those UDAs which have the same name as UDAs in the target Process Definition. System-generated UDAs with same name as defined in the target Process Definition are renamed and pasted to the target Process Definition.

Chapter 8 Using Forms

Most business processes require information-based decisions and actions. Users can use forms to access and display information.

A form is a structured field-based HTML file that acts as an interface for data exchanges between Systemwalker Runbook Automation and structured data repositories.

Forms are used for the following purposes:

- Displaying or reporting data saved in user defined attributes (UDAs).
- Enabling users to add or change data saved in UDAs.

Forms can be associated with Start nodes, Activity nodes, Voting Activity nodes and Compound Activity nodes. A form can be associated with multiple nodes and, conversely, one or more forms can be associated with a node. If multiple forms are associated with a node, then the Web console creates a tab for each.

QuickForms can be used with Systemwalker Runbook Automation. These are forms used for displaying and updating UDA values, and are defined using the Ajax Page Editor. The editor can be used to create any layout of forms parts (known as UI parts, which include TextInput, Select, RadioButton, and so on), to associate these UI parts with UDAs, and to display and refresh UDA values.

For information on QuickForm UI parts, refer to Reference for QuickForm UI Widgets in the *Systemwalker Runbook Automation Reference Guide*.

8.1 Creating a QuickForm

To create a QuickForm in the Ajax Page Editor, follow the steps below:

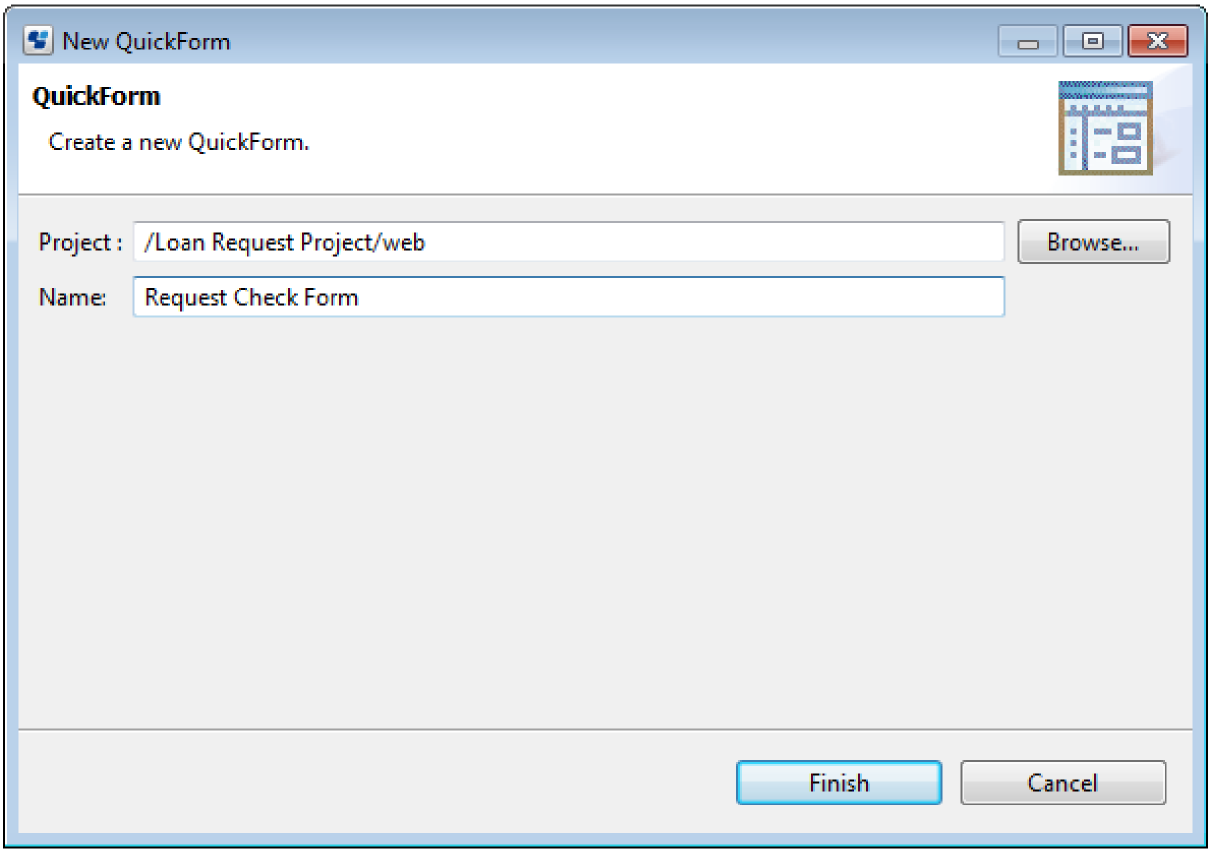
1. Open the forms editor using one of the methods below:

    - In the **Navigator** view, select the Workflow Application project that creates a QuickForm, then from the **File** menu, select **New** >> **QuickForm**.

      -or-

- From the process definition editor, open the process definitions for the Workflow Application project that creates QuickForms and click the associated Start node, Activity node, Voting Activity node, or Compound Activity node, then select **QuickForm** >> **New** from the popup menu.

Figure 8.1 QuickForm creation



QuickForm

Create a new QuickForm.

Project : /Loan Request Project/web

Name: Request Check Form

2. The web folder name of the selected Workflow Application project is displayed in the **Project** field.

3. In the **Name** field, enter the QuickForm name, then click **Finish**.
   A new QuickForm (.jsp file) is created in the web folder.

Note

The following special characters cannot be used in QuickForm (.jsp file) names and in QuickForm subfolder names:
$ @ & < > ? ; # : = , " ' ~ + %

Note

Do not use names starting with "web" for QuickForms (.jsp files) or for subfolders under Workflow Applications or Web folders that include a QuickForm.

8.2 Overview of Ajax Page Editor

The Ajax Page Editor is a tool that allows editing QuickForms, and has the features below:

- Allows editing the QuickForm while checking the execution image.
    - The window layout is displayed using the window image that will actually be displayed.
    - The position and size of parts can be edited in the window image that will actually be displayed.

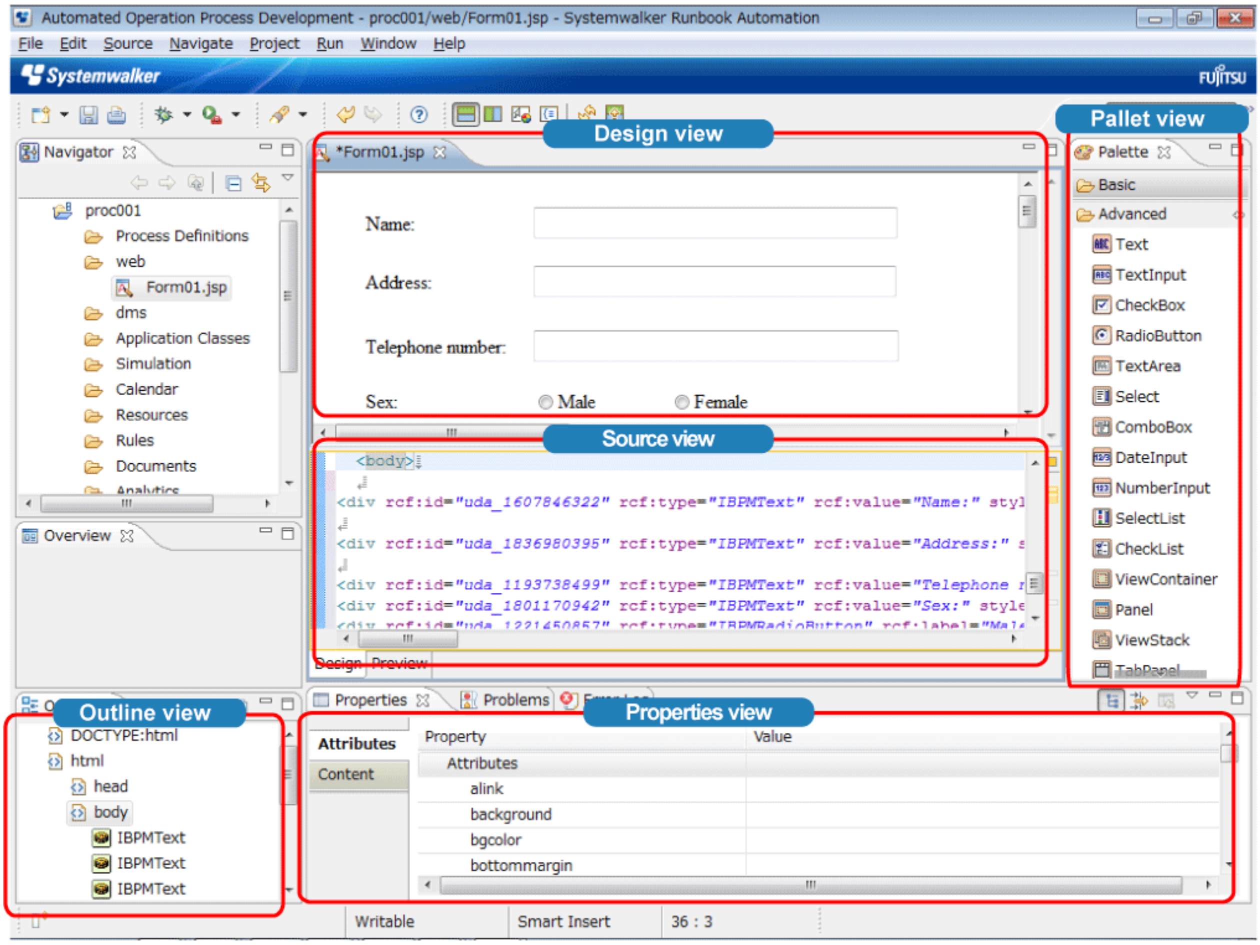
- Windows are displayed in accordance with attribute and CSS settings.
- As with UI parts, HTML tags can be edited in the window image that will actually be displayed.

- Allows text to be edited in conjunction with the visual editing function.

8.2.1 Overview of Views

An overview of Ajax Page Editor views is given below.

Figure 8.2 Ajax Page Editor views



Name:

Address:

Telephone number:

Sex: ○ Male ○ Female

```
<body>
<div rcf:id="uda_1607846322" rcf:type="IBPMText" rcf:value="Name:" styl
<div rcf:id="uda_1836980395" rcf:type="IBPMText" rcf:value="Address:" s
<div rcf:id="uda_1193738499" rcf:type="IBPMText" rcf:value="Telephone r
<div rcf:id="uda_1801170942" rcf:type="IBPMText" rcf:value="Sex:" style
<div rcf:id="uda_1221450857" rcf:type="IBPMRadioButton" rcf:label="Male
```

Property	Value
Attributes	
alink	
background	
bgcolor	
bottommargin	

- Design view
  Enables WYSIWYG editing of a QuickForm constructed using UI parts, and has the following features:
  - Absolute coordinates can be used.
  - The mouse is used to select parts from the palette and to position them.
  - The mouse can be used to move and resize selected parts.
  - The clipboard can be used to perform editing operations on UI parts and HTML tags.
  - Editing results are coupled with the source view, properties view and the outline view.

- Source view
  Used to edit the source code of a QuickForm constructed using UI parts, and has the following feature:
  - Changes are reflected in the design view, properties view, and outline view

- Palette view
  Displays HTML tags and UI parts in the palette, and is used to position them in the design view.

- Outline view
  Displays the tag structure of the window being edited.

- Properties view
  Displays the properties of UI parts selected in the design view, tags and UI parts that have carets in the source view, and tags and UI parts selected in the outline view.
  It contains the tabs below:

  - **Attributes**: Displays the attributes of the selected tag or part.

  - **Contents**: Displays the contents of the selected tag or part.

- **Preview**
  Contains the tabs below.

  - **Design**: Displays the design and source views.

  - **Preview**: Displays a preview.

  The preview enables use of Internet Explorer IE components to check runtime window displays and event processes run by browsers.

Editable files

QuickForm files that meet the following conditions can be edited using the Ajax Page Editor:

- Compatible with XHTML 1.0

- Contains DOCTYPE declaration below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
```

- Uses Ajax framework operation definition (rcf_config.js).

- Uses Ajax framework initialization processing (rcf.js).

The paths for the above files (rcf_config.js and rcf.js) must be defined as follows in QuickForms under the Workflow Application project web folder:

```
<script type="text/javascript" src="../rcf_config.js"></script>
<script type="text/javascript" src="../acf/file/rcf/rcf.js"></script>
```

Similarly, the paths for the files above (rcf_config.js and rcf.js) must be defined as follows in QuickForms under web folder subfolders (web/sub1):

```
<script type="text/javascript" src="../../rcf_config.js"></script>
<script type="text/javascript" src="../../acf/file/rcf/rcf.js"></script>
```

Note

The paths above are updated automatically when a QuickForm is opened. However, if **Save As** saved a QuickForm in another folder during editing, the paths above are not updated automatically. In that case, use the source view to update the paths.

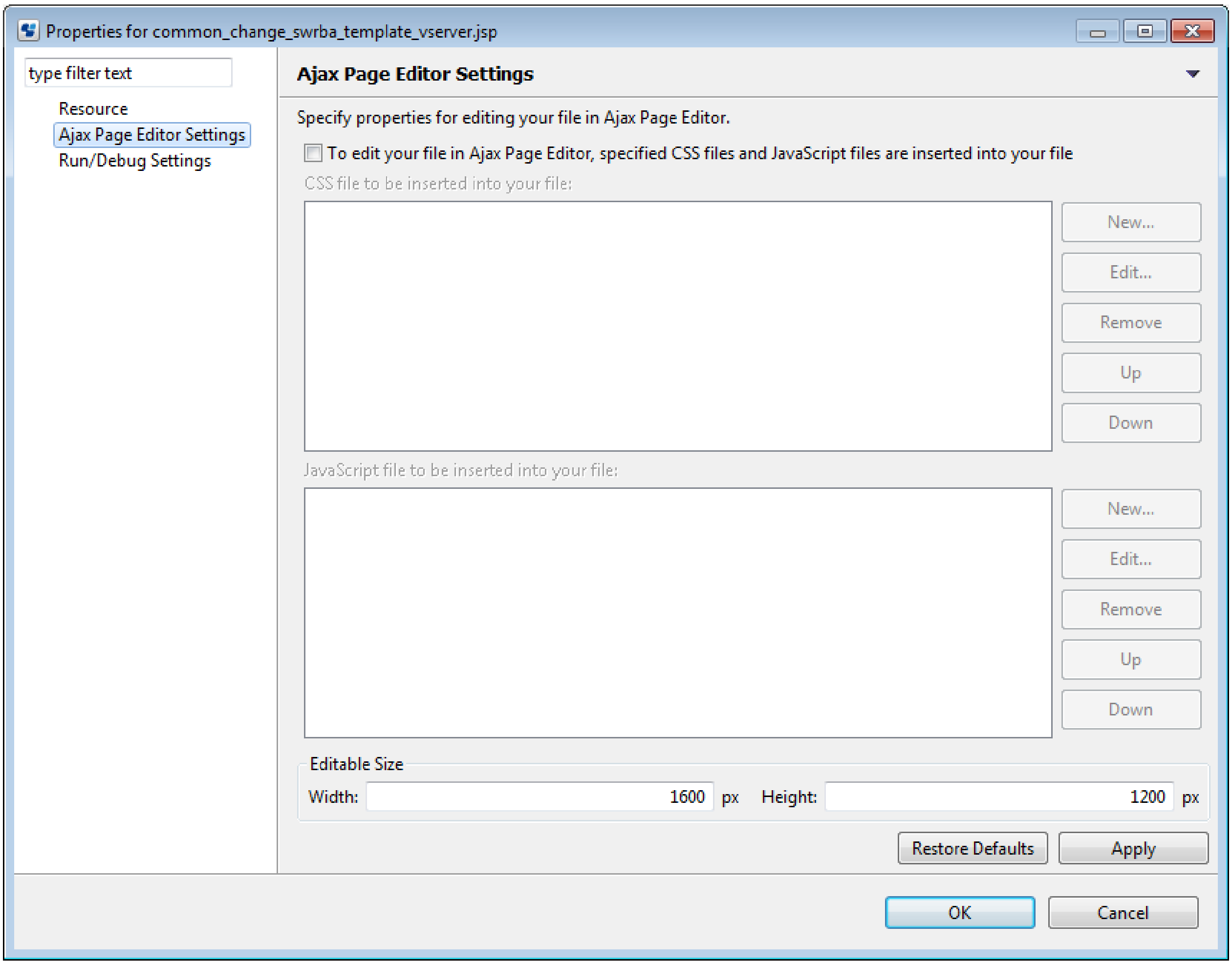
Note

In principle, edit only the <body> tag when editing a QuickForm - do not change other tags unless there is a special need.

Ajax Page Editor settings

To set separate Ajax Page Editor operation options for each file being edited, use the **Ajax Page Editor Settings** page. To display it, execute **Properties** from the jsp file popup menu in the navigator view, then in the **Properties** window select **Ajax Page Editor Setting**.

The **Ajax Page Editor Settings** page is shown below.



Properties for common_change_swrba_template_vserver.jsp

Ajax Page Editor Settings

Specify properties for editing your file in Ajax Page Editor.

□ To edit your file in Ajax Page Editor, specified CSS files and JavaScript files are inserted into your file

CSS file to be inserted into your file:

JavaScript file to be inserted into your file:

Editable Size

Width: 1600 px Height: 1200 px

The table below describes the **Ajax Page Editor settings** page items.

Item	Description
To edit your file in Ajax Page Editor, specified CSS files and JavaScript files are inserted into your file	Do not select this item.
CSS file to be inserted into your file	Cannot be used.
JavaScript file to be inserted into your file	Cannot be used.
Editable Size	For the selected file, specify in the width and height (in pixels) that can be edited using the Ajax Page Editor. UI parts within the specified size range can be edited from the Ajax Page Editor design view.
Restore Defaults	Restores the default settings, which is that nothing is specified, so all specified files are deleted.
Apply	Applies the specified settings.

8.2.2 Design View

The design view displays an image that is the same as when the HTML tags and window parts are executed. Displayed parts can be selected, moved, resized, or deleted in part units. In addition, parts selected in the palette view can be positioned in the design view.

Note

If there is an error in the source code being edited, the design view cannot be used for editing operations. In this case, use the source view to correct the source code.

The table below shows the commands in the design view popup menu.

Item	Description
Refresh	Refreshes the design view display. If an error prevents edits from the design view, then refresh is required after the source code is corrected.
Cut	Cuts the selected part and copies it to the clipboard.
Copy	Copies the selected part to the clipboard.
Paste	Pastes the clipboard contents. This command is enabled for parts cut or copied using the design view.
Delete	Deletes the selected part.
Align	Aligns parts.
Define model binding	Cannot be used.
Define event processing	Cannot be used.
Define functional parts binding	Cannot be used.
Automatic reflection during source view editing	Sets whether edits in the source view should be automatically reflected in the design view. A check mark is displayed at the start of this item if it is set.
Properties	Displays the properties view

Information

Display/edit prohibited state of the design view

If source view edits are not configured to be automatically reflected in the design view, then the design view window becomes grey and changes to the display/edit prohibited state.
The design view changes from this state in the following situations:

- When the focus is moved to it.
- When **Refresh** is clicked in the **Edit** menu, or when ⟳ (Refresh Ajax Page Editor) is clicked in the toolbar.
- When ⇧ (automatic reflection to design view during source code editing) is clicked in the toolbar.

Note that, even when the design view is in the display/edit prohibited state, a window with the latest status can be displayed by switching to the **Preview** tab.

The following two operations can be performed from the design view:

- Parts-common operations
- Part-specific operations (editing operations for container parts and calendar parts)

Parts-common operations

Parts-common operations are described below.

- Position

  Use the method below:

  - From the palette view, position the part in any position within the design view area:

    1. Select a part in the palette view.
    2. Move the mouse cursor to the desired position in the design view and click.
       The part border remains displayed until the part position is decided, enabling the user to check the size of the part.

    When a UI part is positioned in the design view from the palette view, the part ID (rcf:id attribute) is appended automatically.
    To deselect a part in the palette view, press the ESC key.

  For information on the palette view, refer to 8.2.4 Palette View.

  For information on how to position parts within a container part, refer to Container part editing operations.

- Select

  Use one of the methods below:

  - Click on the selectable area of the design view part.
  - Click the part in the outline view.
  - Place the caret on the part tag in the source view.

- Move

  Use one of the methods below:

  - Select the part and drag it.
  - Select the part and press the arrow keys.

- Cut

  Use one of the methods below:

  - Select the part and select **Cut** from the popup menu.
  - Select the part and press Ctrl+X.

- Copy

  Use one of the methods below:

  - Select the part and select **Copy** from the popup menu.
  - Select the part and press Ctrl+C.

- Paste

  Use one of the methods below:

  - Select **Paste** from the design view popup menu.
  - Press Ctrl+V.

  The part is pasted below right of the cut or copied part.
  If the part pasted has an rcf:id set, a new value is set as the rcf:id.

- Delete

  Use one of the methods below:

  - Select **Delete** from the popup menu of the selected part.
  - Select the part and press DELETE.

  Note that, if the selected part has child elements, then they are also deleted.

- Resize

  Use the method below:

    - Drag the resize marks.
      When a part is selected, resize marks are displayed at each corner and side of the rectangular area.

  When a container part is resized, the positions of its child elements are not changed. Other parts included in the part area after resizing do not become its child elements.

- Undo

  Use one of the methods below:

    - Select **Undo** from the **Edit** menu.

    - Press Ctrl+Z.

  The following operations can be undone: position, move, cut, paste, delete, resize, and justify.

- Redo

  Use one of the methods below:

    - Select **Redo** from the **Edit** menu.

    - Press Ctrl+Y.

- Justify

  Use the method below:

    - Select two or more parts, then select a **Justify** option from the popup menu.

  The following types of justifications are available:

    - Align left

    - Align center horizontally

    - Align right

    - Align top

    - Align center vertically

    - Align bottom

Container part editing operations

Parts held in child elements of parts that can operate in the design view are called container parts. Container parts include the following:

- Container parts of window parts (except for FragmentContainers)

- HTML form tags

Operations specific to container parts are described below.

- Moving container parts.
  This also moves the parts (child nodes) on the container part.

- Positioning and moving parts to within a container part.
  To add a part within a container part, position or move the part you want to add as a child element to within the container part area.

For information on container parts, refer to **Container Parts** in the *Systemwalker Runbook Automation Reference Guide*.

Calendar part editing operations

The PopupCalendar and CalendarButton are used as a pair of parts. When the CalendarButton is positioned, the PopupCalendar is also positioned automatically.

8.2.3 Source View

Use the source view to edit HTML/JSP files as plain text - the design view is used to define basic definitions and the source view for additional definitions.

The table below shows the commands in the source view popup menu.

Item	Description
Undo text	Undoes the previous operation.
Return file to previously saved state	Restores previous version of the file.
Store	Saves a file for which work is in progress.
Cut	Cuts the selected data.
Copy	Copies the selected data.
Paste	Pastes the clipboard contents. This command is enabled when data has been saved to the clipboard in text format.
Format	Formats the editor contents.
Format active elements	Formats active element contents.
Properties	Displays the properties view.
Settings	Cannot be used.

8.2.4 Palette View

The palette view is used for operations that add UI parts to the design view. It contains the following categories:

- Basic
- Advanced

[Basic] category

The table below shows the parts displayed in the basic category.

Window part (display name)	HTML tag	Description
Button	input	An input element provided with type=submit displays an input option or button used to tell a user agent to execute a form.
HorizontalRule	hr	An hr element is a divider between text sections. Normally this is a graphic similar to a full-width ruled line.
Image	img	An img element uses a hyperlink to reference an image or icon.
Image Button	input	An input element provided with type=image specifies an image resource to be displayed and specifies the X and Y coordinates of the pixels selected from the image.
Link	a	An a element enables users to navigate document contents.

[Advanced] category

The table below shows the parts displayed in the advanced category.

Window part (display name)	Description
Text	Displays text, and corresponds to the HTML <span>*text*</span>.
TextInput	Allows input and editing of a single line of text, and corresponds to the HTML <input type="text" value="*value*"> and <input type="password" value="*value*">.
CheckBox	Expresses either the on or the off status, and corresponds to the HTML <input type="checkbox" value="*value*">.

Window part (display name)	Description
RadioButton	Radio button part. This part corresponds to the HTML <input type="radio" value="*value*">.
TextArea	Allows input and editing of a single line or multiple lines of text, and corresponds to the HTML <textarea>.
Select	Selection list that enables single or multiple selections, and corresponds to the HTML <select> <option>.
ComboBox	Combination of an input field and a selection list. This part is for selection of an item and its display.
DateInput	Allows input and editing of date and time data, which is one type of TextInput.
NumberInput	Allows input and editing of numeric values, which is one type of TextInput.
SelectList	Selection list that enables single or multiple selections.
CheckList	Selection list containing check boxes.
ViewContainer	General container part that can contain HTML elements as child elements, which can be handled as grouped parts. ViewContainer itself does not have a display part.
Panel	Container part with a title bar comprised of a title part and body part. It can have HTML elements as child elements, and the contents defined in the child elements become the contents of the body part.
ViewStack	Container part used to switch between displays in the same position.
TabPanel	Container part used if tabs are used to switch between displays.
FragmentContainer	Container part used to display window information from an external source at any time after the page is displayed.
Calendar	Performs calendar display and date selection.
CalendarButton	Displays the PopupCalendar (used in conjunction with the PopupCalendar).

Palette view popup menu

The table below shows the commands in the palette view popup menu.

Item	Description
Hide (*1)	Hides a palette category.
Layout	Changes the layout of parts.
Use large icons	Uses large icons to display parts in the palette.
Customize	Displays the **Customize Palette** dialog box for customizing the palette categories.
Set	Displays the Palette Settings dialog box for setting the display format of parts in the palette.
Pinned (*1)	Causes a category to be pinned, in which case it is not closed.

*1: Only displayed when a category is right-clicked.

8.2.5 Outline View

Displays a tree-format list of the parts in the window. Parts in the window can also be selected from this list.

The table below shows the commands in the outline view popup menu.

Item	Description
Delete	Deletes the selected node
Define model binding	Cannot be used.
Define event process	Cannot be used.
Define functional part binding	Cannot be used.
Properties	Displays the properties of the selected node

8.2.6 Properties View

Displays and allows changes to the properties of a tag or part selected in the design view, source view or outline view.

It contains the tabs below:

- **Attributes**: Displays the attributes of the selected tag or part.
- **Contents**: Displays the contents of the selected tag or part.

Displaying and editing attributes

The **Attributes** tab displays and allows changes to the attribute values of the selected tag or part.

The table below shows the items displayed by the **Attributes** tab of the properties view.

Item	Description
Properties	Properties are displayed hierarchically under the following categories: - Common Property: Properties common to all window parts - Control Property: Properties specific to UI parts - Style Property: Properties specific to style - Event Listener: Properties specific to be able to use event listener - Validation Property: Properties such as whether input is mandatory, etc.
Value	Displays property values

Note

When a value is deleted at the properties view, the property is not deleted - it just has its value set to " ".
Therefore, when a property having a value other than the default (" ") is deleted at the properties view, the value becomes invalid.

This causes a design view error and prevents editing using the design view.
In this case, take one of the following actions:

a. Delete the empty property (only if it is not mandatory).

   - Use the source view to delete the property.
   - From the properties view popup menu, select **Restore Defaults**.

b. Specify a valid value for the property

However, when property is mandatory, to delete property, error has occurred. In this case, action above b.

Displaying and editing contents

Clicking the **Contents** tab displays the contents of the selected tag or part. The contents can be edited in the contents editing area.

The HTML tags below (empty elements) cannot be edited using the contents editing area:

- area
- base
- br
- col
- hr
- img
- input
- link
- meta
- param

HTML tags can also bet set as non-editable (with the exception of the tags listed above).
To set a HTML tag as non-editable, follow the steps below:

1. Select **Settings** from the **Window** menu.
2. In the settings items in the **Settings** dialog box, select **Ajax Page Editor >> Properties view**.
3. Under **Contents tab**, at **Specify tags that have no content** enter the HTML tag.
   Separate multiple tags with a comma (,).

- Newline character

  To enter a newline character in the contents editing area, press one of the following:

  - Ctrl+Enter
  - Shift+Enter

  Users can select which of the above key combinations to use to enter a newline character (the default is Ctrl+Enter).
  To set the newline character input keys, follow the steps below:

  1. Select **Settings** from the **Window** menu.
  2. In the settings items in the **Settings** dialog box, select **Ajax Page Editor >> Properties view**.
  3. Under **Contents tab**, at **Set the return key** select the newline character input keys.

- Conversion of special characters

  When escapes are set, the special characters shown below are converted.
  However, if values are already set in the contents, special characters are not converted.

Original character	Character after conversion
"	&quot;
&	&amp;
<	&lt;
>	&gt;

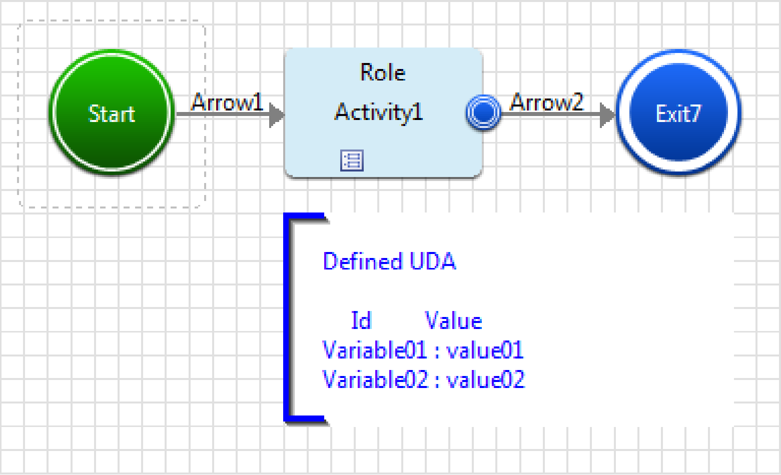
  To set escapes, follow the steps below:

  1. Select **Settings** from the **Window** menu.
  2. In the settings items in the **Settings** dialog box, select **Ajax Page Editor >> Properties view**.
  3. Under **Contents tab,** select **Convert escape characters**.

8.3 QuickForm Introduction

This section describes how to create a QuickForm so that the user can start a process instance. The figure below shows the process definitions that create a QuickForm.

Figure 8.3 Process definitions for creating a QuickForm



Step 1: Create a QuickForm.

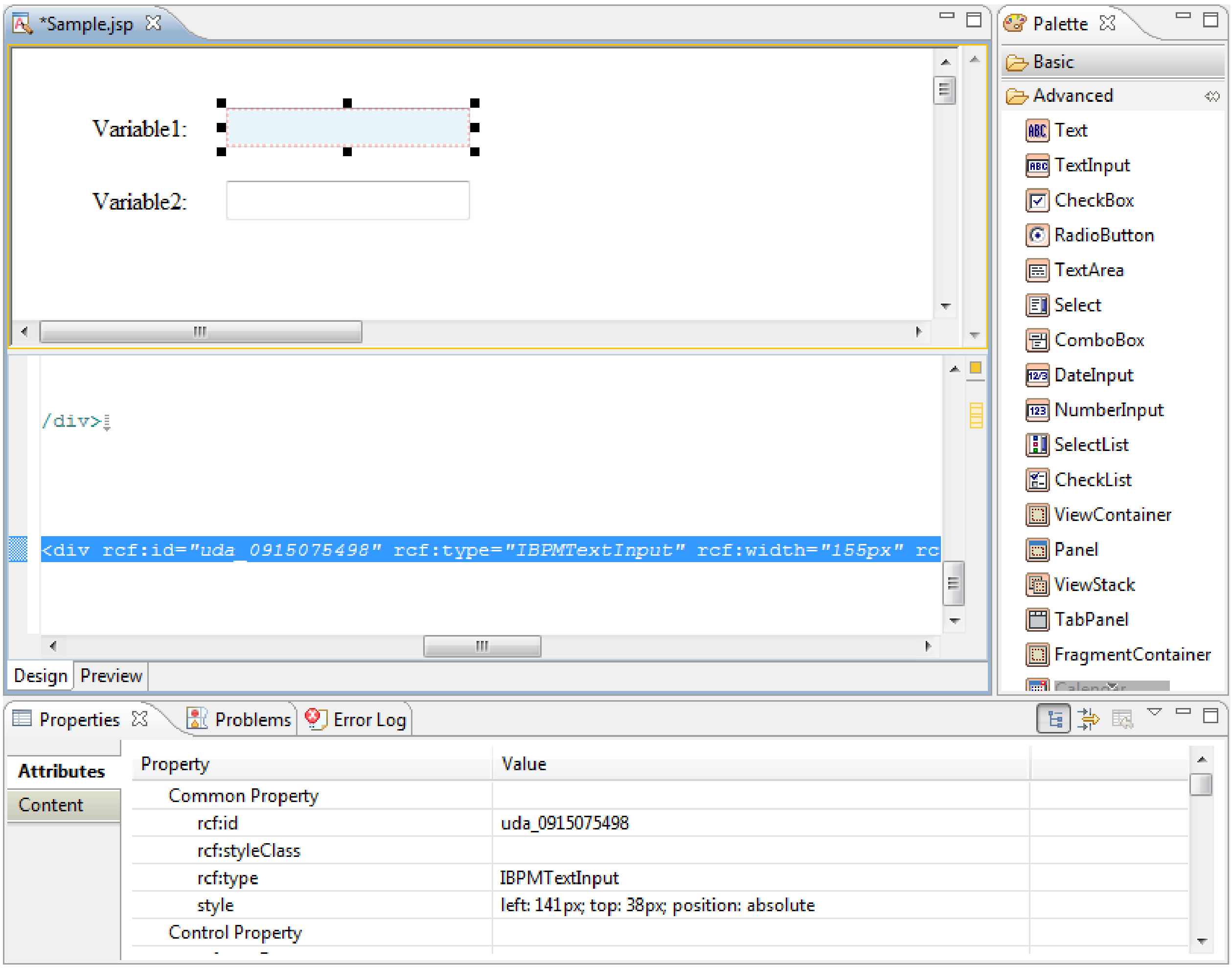
1. From the process definition editor, open the process definitions for the Workflow Application project and click the Activity node to be associated.
2. Select **QuickForm** >> **New** from the popup menu.
3. Enter the filename (for details, refer to 8.1 Creating a QuickForm).

The QuickForm (.jsp file) is created in the Workflow Application project.

Step 2: Define UI parts that display UDA values

Define the UI parts that display the UDA values.

Figure 8.4 Method for defining the UI parts that display the UDA values



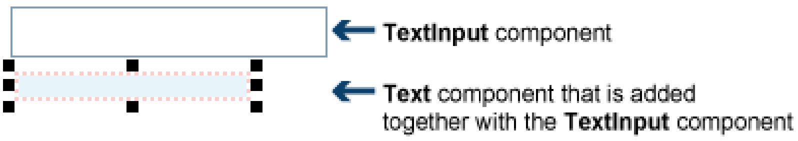
```
/div>
<div rcf:id="uda_0915075498" rcf:type="IBPMTextInput" rcf:width="155px" rc
```

Property	Value
Common Property	
rcf:id	uda_0915075498
rcf:styleClass	
rcf:type	IBPMTextInput
style	left: 141px; top: 38px; position: absolute
Control Property	

1. Select the **Text** part in the **Advanced** category in the palette view of the Ajax Page Editor and position it in the design view (the **Text** part is used as a label to identify the UDA).

2. Select the **TextInput** part in the **Advanced** category in the palette view and position it in the design view (the **TextInput** part is used to display the UDA values).
   When the **TextInput** part is added, the **Text** part is added with it. If an incorrect operation (such as the user not entering a value when mandatory input is defined for this **TextInput** part, associating an unsupported UDA type with this part, or similar) is executed at the Console for this **TextInput** part, an error message is displayed in this **Text** part.

   Figure 8.5 'Text' part added together with the 'TextInput' part

   
   TextInput component
Text component that is added together with the TextInput component

3. Repeat Steps a and b to add another **Text** part and **TextInput** part pair.

4. Select the first **Text** part added, then in the properties view enter "Variable01:" in the **rcf:value** field.

5. Select the second **Text** part added, then in the properties view enter "Variable02:" in the **rcf:value** field.

6. Select the first **TextInput** added, then in the properties view enter "uda_Variable01:" in the **rcf:id** field (the synopsis for this value is "uda_<*UDA identifier*>").
   If the **Text** part that displays error messages is required, then select the **Text** part that displays error messages and in the properties view enter "error_Variable01:" in the **rcf:id** field (the synopsis for this value is "error_<*UDA identifier*>").

7. Select the second **TextInput** added, then in the properties view enter "uda_Variable02:" in the **rcf:id** field. If the **Text** part that displays error messages is not required, delete this **Text** part.

Information

- When a **TextInput** part is placed in the design view, the **Text** part used to display error messages is added with it (refer to Step 2.). The border (surrounding the **TextInput** part and the **Text** part) is also added.
  If the **Text** part used to display error messages is not required, the procedure for deleting this **Text** part and border and retaining only the **TextInput** part is shown below:

  1. Select the unnecessary **Text** part and delete it.

     The **Text** part is deleted, and the **TextInput** part and the border (surrounding the **TextInput** part) remains.

  2. Delete the border. If a **TextInput** part is within the border area, the **TextInput** part will also be deleted. Use the procedure below to avoid deleting the **TextInput** part:

     1) Select the **TextInput** part and move outside the area surrounded by the border.

     2) Select and delete the border that surrounded the **TextInput** part.

     3) Move the **TextInput** part to its original position.

- If you want to delete a **TextInput** part and you select and delete the **TextInput** part, the **Text** part that displays error messages and the border (surrounding the **TextInput** part and the **Text** part) are not deleted with it. If you want to delete all of these, select the border and delete it. This deletes the border and also the **TextInput** part and **Text** part that are within the border area.

Note

- In clients (QuickForms), the values entered in basic category UI parts are not validated - they are saved to the server without validation.

- If multiple QuickForms are defined for a node, a tab is displayed at the Console for each. This **Save** button is defined in each of these tabs.

- If the same UDA is used in multiple QuickForms associated with a particular node, then the same value must be set at the Console for all parts associated with the UDA.

  For this reason, it is recommended to avoid associating each part with the same UDA.

Step 3: Save the QuickForm.

The jsp file will be saved in the Workflow Application project.

Step 4: Associate the QuickForm with a process definition.

1. Use the process definition editor to open the process definitions.
2. In the **Navigator** view, select the created QuickForm.
3. Drag and drop the QuickForm to the Start node of the process definitions.

   For details, refer to 8.4.2 Associating a QuickForm with a Node.

8.4 QuickForm Management

This section describes how to manage QuickForms.

8.4.1 Updating a QuickForm

A QuickForm can be changed to, for example, add a user defined attribute (UDA) to a form or to change a UI part type.

To update a QuickForm, follow the steps below:

1. Open the QuickForm in the Ajax Page Editor using one of the methods below:

   - In the **Navigator** view, double-click the QuickForm or right-click it and select **Open** from the popup menu.

   - In the process definition editor, right-click the node associated with the QuickForm and select **QuickForm** >> **Edit** from the popup menu.

2. Change the QuickForm using the Ajax Page Editor.

   For details, refer to 8.2 Overview of Ajax Page Editor.

3. Click the **Save** button in the workbench window to save the changes.

8.4.2 Associating a QuickForm with a Node

When a user starts to create a QuickForm for a specific node using the process definition editor, the form is automatically associated with that node. Existing QuickForms can be reused and associated with additional nodes.

To associate an existing QuickForm with a node, follow the steps below:

1. Open the process definitions.

2. In the **Navigator** view, click the QuickForm you want to associate with the Start node, Activity node, Voting Activity node, or Compound Activity node.

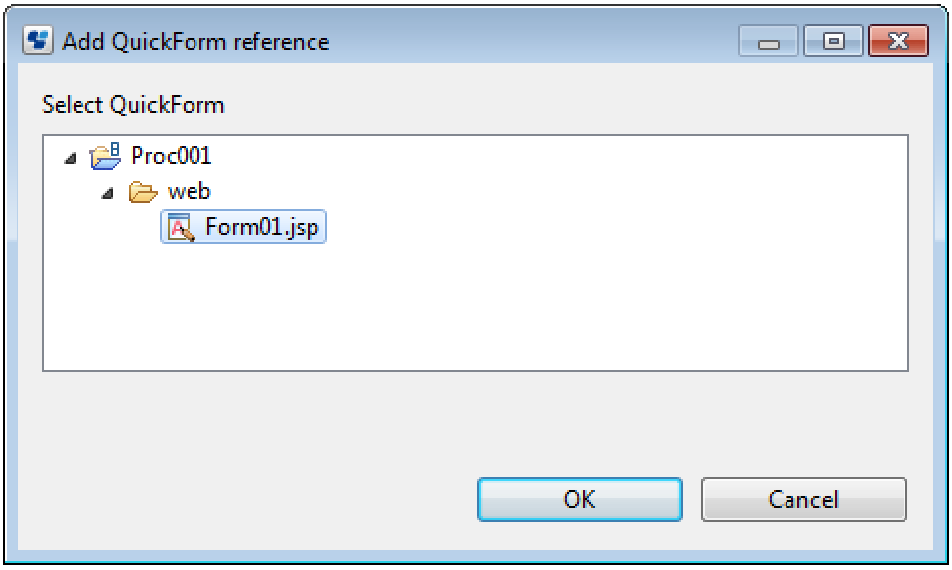
3. Drag the form to those nodes (the form name and path can be checked under the **Forms** tab of the **Properties** view).

To reuse an existing QuickForm, follow the steps below:

1. Open the process definitions.

2. Right-click the node you want to associate with the QuickForm and select **QuickForm** >> **Add Node** from the popup menu.
   The dialog box below will be displayed:

   Figure 8.6 Add QuickForm to Node



3. From the QuickForm list, select the form to be associated with the specific node, then click **OK**.

Note

The **Browse** button under the **Forms** tab of the **Properties** view of the process definition Start node, Activity node, Voting Activity node, or Compound Activity node is provided in order to associate forms other than QuickForms with a node.
Use the operations shown above to associate a QuickForm with a node.

8.4.3 Canceling the Association of a QuickForm with a Node

The association of a QuickForm with a node can be cancelled. The form itself is not deleted even if the association is deleted.

To cancel the association of a QuickForm with a node, use one of the methods below:

- Using the **Properties** view:
    1. Select the activity.
    2. From the **Properties** view, select the **Forms** tab.
    3. Select the form for which you want to cancel the association, then click **Delete**.
- Using the popup menu:
    1. In the process definition editor, right-click each node.
    2. Select **QuickForm > Delete Node**from the popup menu.
    3. Select the form for which you want to cancel the association, then click **OK**.

8.4.4 Renaming a QuickForm

A QuickForm name can be changed, taking into account all references to it.

To rename a QuickForm, follow the steps below:

1. In the **Navigator** view, right-click the QuickForm, then select **Rename** from the popup menu.
2. In the **Name** field, enter the new name of the form.
3. Decide whether or not to update all assign to this QuickForm to match this new name.
4. Click **OK**.

8.4.5 Deleting a QuickForm

A QuickForm name can be deleted, taking into account all references to the form.

To delete a QuickForm, follow the steps below:

1. In the **Navigator** view, right-click the QuickForm, then select **Delete** from the popup menu.
2. Decide whether or not to also delete all assign to this QuickForm.
3. In the confirmation window, click **Yes**.

8.4.6 Importing a QuickForm

The Workflow Application project that is the target of the QuickForm import must already exist.

To import a QuickForm, follow the steps below:

1. In the **Navigator** view, right-click the web folder for importing the form, then select **Import** from the popup menu.
2. Move to the location where the QuickForm that you want to import is stored.
3. Select the form, then click **End**.

For details, refer to 3.1.4 Importing Folders and Files.

8.4.7 Exporting a QuickForm

A QuickForm can be exported from Systemwalker Runbook Automation Studio to a file system. Exported files can be imported to other systems.

Note

An entire Workflow Application project, including its process definitions, forms and attachments can be exported in one operation, and applications can be deployed on the Admin Server later. For details, refer to 3.1.13 Exporting Workflow Application Projects.

To export a QuickForm, follow the steps below:

1. In the **Navigator** view, right-click the QuickForm, then select **Export**.
2. Move to the location where the exported form is stored.
3. Click **Save**.

Chapter 9 Modeling Subprocesses and Chained-Processes

Systemwalker Runbook Automation Studio enables you to break complex tasks into easier-to-handle units.

Using subprocesses, you can seamlessly link the work of different departments with different processes for the purpose of trading information and coordinating collaborative tasks. A subprocess can run on the same workflow server as the parent process.

A chained-process is a process that operates independently once it has been activated by its parent process. The parent process continues with its own flow logic without waiting for the chained-process to complete.

This chapter explains how to model subprocesses and chained-processes.

9.1 Modeling Subprocesses

Subprocesses are used to break complex tasks into a hierarchy of easier-to-handle units. They are most appropriate for seamlessly linking the work of different departments with different processes for the purpose of trading information and coordinating collaborative tasks.

A subprocess is represented by a Subprocess Node in the parent process definition. When a Subprocess Node is reached, control is passed to the subprocess. The parent process waits for the subprocess to complete and the results to come back.

To model a subprocess:

1. Create the parent process definition and model the process flow.
2. Create the subprocess definition and model the process flow.
3. Add a Subprocess Node to the parent process definition.
4. Connect the parent to the subprocess definition.

   For instructions, refer to section 9.1.1 Connecting Parent and Subprocess Definitions.

   When the process definitions are connected, you can easily navigate from the parent to the subprocess definition. For instructions, refer to section 9.1.2 Navigating to Subprocess Definitions.
5. In both parent and subprocess definitions, specify the User Defined Attributes (UDAs) that will be passed back and forth.

   Note

   You can use different names for the same UDA in the process definitions involved. However, the data type must be identical. Otherwise, you cannot map the UDAs.
6. Define data mappings for the UDAs that need to be passed back and forth.

   For instructions, refer to section 9.1.3 Defining Data Mappings for Subprocesses.

   Note

   Be careful when designing process definitions that have recursive subprocesses. Check all process definitions involved and ensure that there are no infinite recursions.

9.1.1 Connecting Parent and Subprocess Definitions

Once you have created the parent and subprocess definition, you can connect them.

To connect process definitions using drag and drop:

To connect the parent process definition and the subprocess, drag the subprocess definition and drop it on the Subprocess Node of the parent process definition.

To connect process definitions using **Browse**:

1. Ensure that you have added a Subprocess Node to the parent process definition.
2. Select the Subprocess Node to display the **Properties** view for the node.
3. Select the **Data Mapping** tab.
4. Click **Browse**.
5. In the **Select Subprocess Definition** dialog, click **Get List**.
6. Select the subprocess definition and click **OK**.

9.1.2 Navigating to Subprocess Definitions

Prerequisites:

- You have created a subprocess definition.
- In the parent process definition, you have added a Subprocess Node.
- You have connected the Subprocess Node to the subprocess definition.

You can navigate to the subprocess definition from the parent process definition.

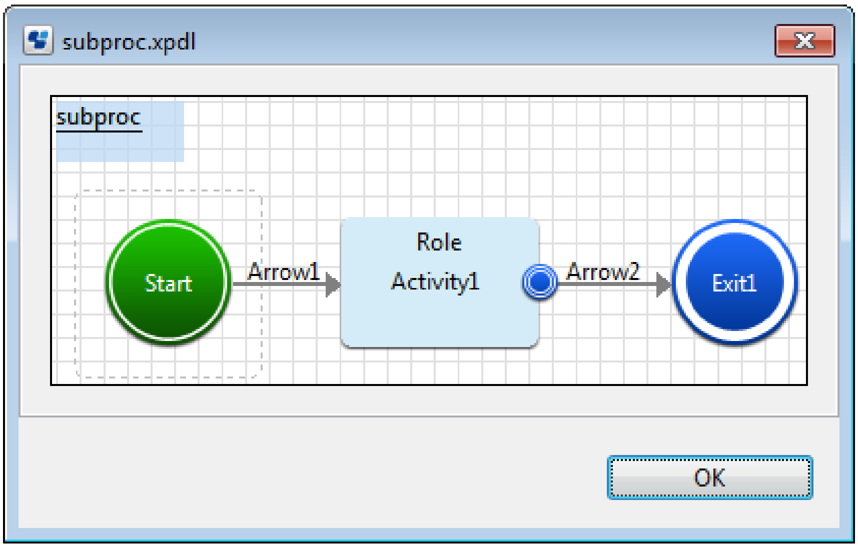
To navigate to a subprocess definition:

1. In the parent process definition, select the Subprocess Node.
2. Right click and select **Go to Subprocess** from the pop-up menu.

A new Process Definition editor is opened displaying the subprocess definition.

If you do not want to edit the subprocess definition, but simply want to view it, you can display it in a separate window: In the parent process definition, click the plus sign (+) of the Subprocess Node. The subprocess definition will be opened in a separate window, for example:

Figure 9.1 Subprocess Definition



9.1.3 Defining Data Mappings for Subprocesses

When data is to flow between parent and subprocess, you map the UDA of the parent process definition to the corresponding UDA of the subprocess definition. Also, you specify in which directions values are to be passed between parent and subprocess definition.

Prerequisites:

- You have specified User Defined Attributes (UDAs) that will be passed back and forth in the parent process definition and in the subprocess definition.
- The UDAs you want to pass have the same data type in the parent process definition and subprocess definition.

- You have connected the parent process definition and the subprocess definition. You have connected the parent process definition and the subprocess definition.

Auto data mapping for parent process definition and subprocess definition

When the pre-requisites mentioned above are fulfilled and you connect the parent process definition and the subprocess definition, data mapping between them is automatically completed. The UDAs of the parent process definition are mapped to the UDAs of the subprocess that have the same data types as that of the parent process definition. To view the automatically mapped UDAs, click the subprocess node of the parent process definition and select the **Data Mapping** tab. The mapped UDAs are displayed in **All Data Mappings** area along with their data flow directions.

Editing data mapping

To delete unwanted mappings, select them in **All Data Mappings** area and click the **Delete** button.

You need to manually edit the UDA mappings in the following scenarios:

- If the mappings that were automatically created after connecting the parent and subprocess definition were incorrect and need to be changed
- If you add or edit UDAs in the parent process definition and the subprocess definition after connecting them

To manually edit UDA mappings

1. Select the **Data Mapping** tab of the subprocess node in the parent process definition.
2. To map a UDA:
   a. From the **Mapping Type** drop-down list, select the data type of the UDA you want to map. This drop-down list displays all data types that occur in the parent process definition. The UDA in the parent process definition and UDA in the subprocess definition drop-down lists are populated with the UDAs of the same mapping type as selected in the Mapping Type drop-down list.

   b. Select the UDA in the parent process definition and the corresponding UDA of the same type as that of the parent process definition UDA in the subprocess definition.

   c. Click **Add**.

   In the **All Data Mappings** area, the mapped UDAs are displayed.
3. If the UDA type is XML then click in the corresponding **XPath** column field. A drop-down list is displayed. Select from the drop-down list, the XPath expression that specifies where the data can be found in the XML string of the Source UDA. This field is active only if you have selected a UDA of type XML. The XPath expressions related to the selected UDA are displayed in the **XPath** drop-down list.
4. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the ellipsis (...) button that is displayed next to the **XPath** drop-down list. **XPath Editor** dialog box is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click OK to use the edited XPath expression.

Note

The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note

The XPath expressions related to the selected XML UDA are displayed in the XPath drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

Note

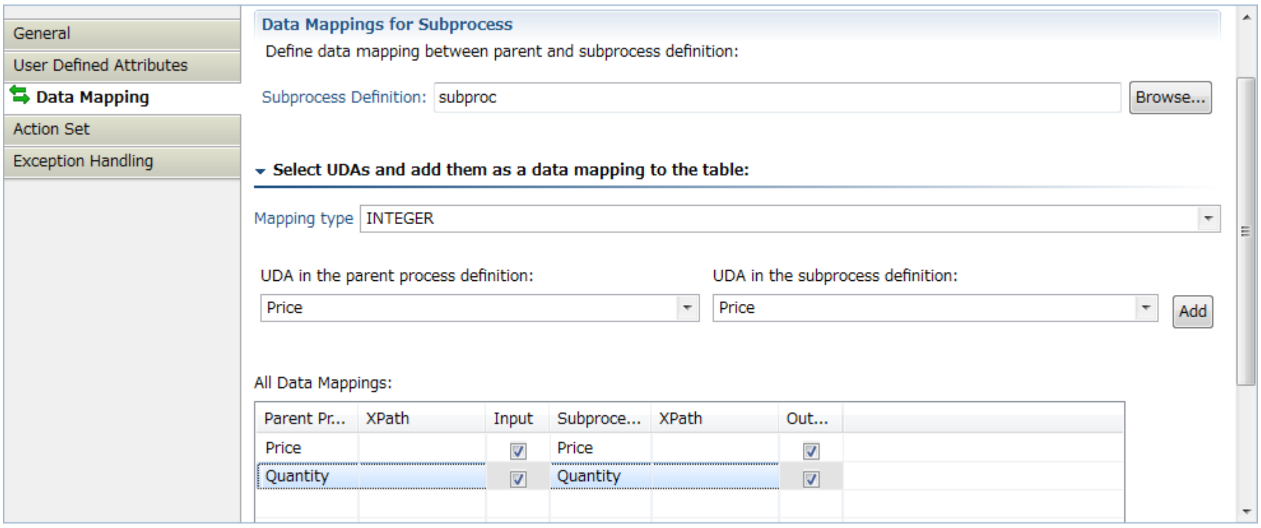
XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

5. In the **All Data Mappings** area, specify the data flow between process definitions:

   a. When the data value passes from the parent to the subprocess, click **Input**.

   b. When the data value passes from the subprocess to the parent process, click **Outpu**t.

Figure 9.2 Defining Data Mappings



Define data mapping between parent and subprocess definition:

Subprocess Definition: subproc

Select UDAs and add them as a data mapping to the table:

Mapping type INTEGER

UDA in the parent process definition: Price

UDA in the subprocess definition: Price

All Data Mappings:

Parent Pr...	XPath	Input	Subproce...	XPath	Out...
Price		☑	Price		☑
Quantity		☑	Quantity		☑

9.2 Modeling Chained-Processes

A chained-process operates independently once it has been activated by its parent process. The parent process continues with its own flow logic without waiting for the chained-process to complete.

The interaction between a parent process and a chained-process goes in one direction only. Neither process control nor values of User Defined Attributes (UDAs) are passed back to the parent process.

To model a chained-process:

1. Create the parent process definition and model the process flow.

2. Create the Chained-Process definition and model the process flow.

3. Add a Chained-Process Node to the parent process definition.

4. Connect the parent to the chained-process definition.

   For instructions, refer to section 9.2.1 Connecting Parent and Chained-Process Definitions.

   When the process definitions are connected, you can easily navigate from the parent to the chained-process definition. For instructions, refer to section 9.2.2 Navigating to Chained-Process Definitions.

5. In both parent and chained-process definitions, specify the User Defined Attributes (UDAs) that will be passed from the parent to the chained-process.

   Note

   You can use different names for the same UDA in the process definitions involved. However, the data type must be identical. Otherwise, you cannot map the UDAs.

6. Define data mappings for the UDAs that need to be passed to the chained-process.

   For instructions, refer to section 9.2.3 Defining Data Mappings for Chained-Processes.

9.2.1 Connecting Parent and Chained-Process Definitions

Once you have created the parent and chained-process definition, you can connect them.

To connect process definitions using drag and drop:

To connect the parent process definition and the chained-process, drag the chained-process definition and drop it on the Chained-Process Node of the parent process definition.

To connect process definitions using Browse:

1. Ensure that you have added a Chained-Process Node to the parent process definition.
2. Select the Chained-Process Node to display the **Properties** view for the node.
3. Select the **Data Mapping** tab.
4. Click **Browse**.
5. In the **Select Subprocess Definition** dialog, click **Get List**.
6. Select the chained-process definition and click **OK**.

9.2.2 Navigating to Chained-Process Definitions

You can navigate from a parent process definition to a chained-process definition.

Prerequisites:

- You have created a chained-process definition.
- You have added a Chained-Process Node to the parent process definition.
- You have connected the Chained-Process Node to the chained-process definition.

To navigate to a chained-process definition:

1. In the parent process definition, select the Chained-Process Node.
2. Right click and select **Go to Subprocess** from the pop-up menu.

   A new Process Definition editor is opened displaying the chained-process definition.

If you do not want to edit the chained-process definition, but simply want to view it, you can display it in a separate window: In the parent process definition, click the plus sign (+) of the Chained-Process Node. The chained-process definition will be opened in a separate window.

9.2.3 Defining Data Mappings for Chained-Processes

This section explains the procedure for defining data mapping for chained processes.

Prerequisites:

- You have specified User Defined Attributes (UDAs) that will be passed from the parent to the chained-process definition in the parent and in the chained-process definition.
- The UDAs you want to pass have the same data type in the parent and the chained-process definition.
- You have connected the parent and the chained-process definition.

When data is to flow between parent and chained-process, you map the UDA of the parent process definition to the corresponding UDA of the chained-process definition.

Auto data mapping for parent process definition and chained-process definition

When the pre-requisites mentioned above are fulfilled and you connect the parent process definition and the chained-process definition, data mapping between them is automatically completed. The UDAs of the parent process definition are mapped to the UDAs of the chained-process that have the same data types as that of the parent process definition. To view the automatically mapped UDAs, click the chained-process node of the parent process definition and select the Data Mapping tab. The mapped UDAs are displayed in All Data Mappings area along with their data flow directions.

Editing data mapping

To delete unwanted mappings, select them in **All Data Mappings** area and click the **Delete** button. You need to manually edit the UDA mappings in the following scenarios:

- If the mappings that were automatically created after connecting the parent and chained-process definition were incorrect and need to be changed

- If you add or edit UDAs in the parent process definition and the chained-process definition after connecting them

To manually edit UDA mappings

1. Select the **Data Mapping** tab of the chained-process node in the parent process definition.

2. To map a UDA:

    a. From the Mapping Type drop-down list, select the data type of the UDA you want to map. This drop-down list displays all data types that occur in the parent process definition. The UDA in the parent process definition and UDA in the chained-process definition drop-down lists are populated with the UDAs of the same mapping type as selected in the Mapping Type drop-down list.

    b. Select the UDA in the parent process definition and the corresponding UDA of the same type as that of the parent process definition UDA in the chained-process definition.

    c. Click **Add**.

    d. If the UDA type is XML then click in the corresponding **XPath** column field. A drop-down list is displayed. Select from the drop-down list, the XPath expression that specifies where the data can be found in the XML string of the UDA. This field is active only if you have selected a UDA of type XML. The XPath expressions related to the selected UDA are displayed in the **XPath** drop-down list.

    e. Optional: If you want to edit the XPath expression that you selected in the XPath drop-down list, click the ellipsis (...) button that is displayed next to the **XPath** drop-down list. XPath Editor dialog box is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click OK to use the edited XPath expression.

Note

The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note

The XPath expressions related to the selected XML UDA are displayed in the **XPath** drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

Note

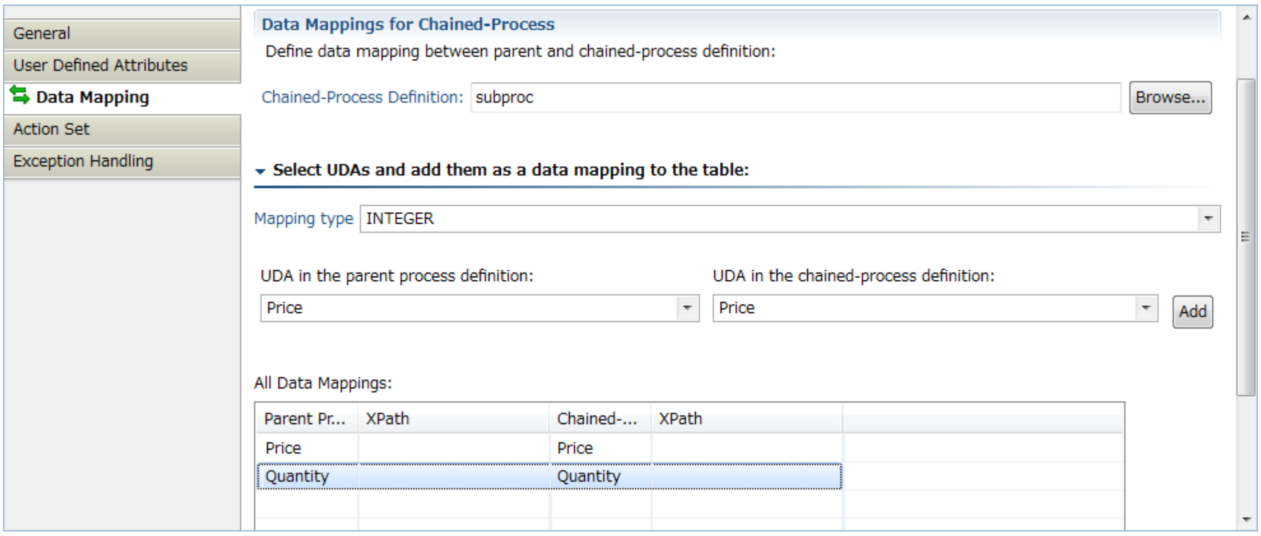
XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

The data mapping is added to the **All Data Mappings** area.

Note

This indicates that the data value passes from the parent to the chained-process. As the chained-process operates independently from its parent once it has become active, no data values are passed back to the parent.

Figure 9.3 Defining Data Mappings



General

User Defined Attributes

Data Mapping

Action Set

Exception Handling

Data Mappings for Chained-Process

Define data mapping between parent and chained-process definition:

Chained-Process Definition: subproc

Browse...

Select UDAs and add them as a data mapping to the table:

Mapping type INTEGER

UDA in the parent process definition: Price

UDA in the chained-process definition: Price

Add

All Data Mappings:

Parent Pr...	XPath	Chained-...	XPath
Price		Price	
Quantity		Quantity	

Chapter 10 Decision Tables

Decision Tables allow you to design advanced rules for decision making without programming.

Decision Tables use a simple and powerful table based approach for managing rules dynamically.

To add Decision Tables, see 10.2.2 Creating a New Decision Table.

10.1 Summary of the Decision Tables Procedure

You must perform the steps for creating and configuring a Decision Table, in a particular order, if you want a useful Decision Table. It is assumed that you have a particular Process Definition for which you need this advanced rules functionality. The following is a summary of the steps for creating, configuring, and using a Decision Table:

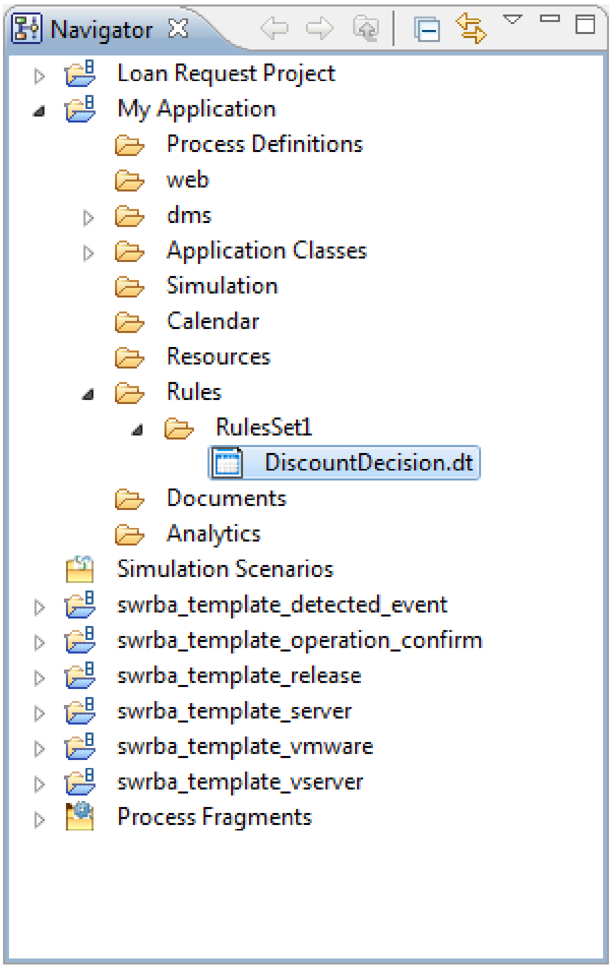
1. Open an existing application or create a new application. Refer to 3.1.1 Creating Workflow Application Projects for instructions.
2. Add **Rules Sets** to the **Rules** folder. Refer to the section 10.2.1 Creating a New Rule Set.
3. Add Decision Table Files to the **Rules Set** folder. Refer to the section 10.2.2 Creating a New Decision Table.
4. Add conditions to your Decision Table. These are the inputs to your Decision Table. You will map these conditions to User Defined Attributes (UDAs) in a later step. Refer to section 10.3.1 Adding or Editing Conditions for instructions.
5. Add results to your Decision Table. These are the outputs to your Decision Table. You will map these results to UDAs in a later step. Refer to section 10.3.2 Adding or Editing Results for instructions.
6. Add decisions to your Decision Table. These are the actual rules. These rules are evaluated to determine the results of your Decision Table. Refer to section 10.3.3 Adding or Editing Decisions for instructions.
7. Save your changes by clicking **Save** in the tool bar.
   The **Save** button appears as soon as you have modified any information in a Decision Table.
8. The system validates the Decision Table while saving the changes made to it. Refer to 10.4 Validating Decision Tables for Errors and Warnings for more information about validating the Decision Tables for errors and warnings.
   After validating the Decision Table, errors and warnings in the Decision Table, if any, are displayed in the Problems View tab.
9. Test the Decision Table that you just created. Refer to section 10.5 Validating Decision Table Rules for instructions.
10. Assign the Decision Table to a Java Action and map the conditions and results added in an earlier step to the UDAs that you want to use as input to the Decision Table. Refer to section 10.6 Using the Decision Table Action for instructions.

10.2 Creating a Decision Table

In the Studio application, every Application Project has a **Rules** directory that contains **Rules Sets**.

Each Rules Set stores Decision Tables (.dt files). **Rules Set** is a sub-directory within the **Rules** directory.x

Figure 10.1 Decision Table Menu



Perform the following steps to create a Decision Table File

- Create a new application. Refer to 3.1.1 Creating Workflow Application Projects
- Create a **Rules Set** folder. Refer to 10.2.1 Creating a New Rule Set
- Create a Decision Table inside the **Rules Set** folder. Refer to 10.2.2 Creating a New Decision Table

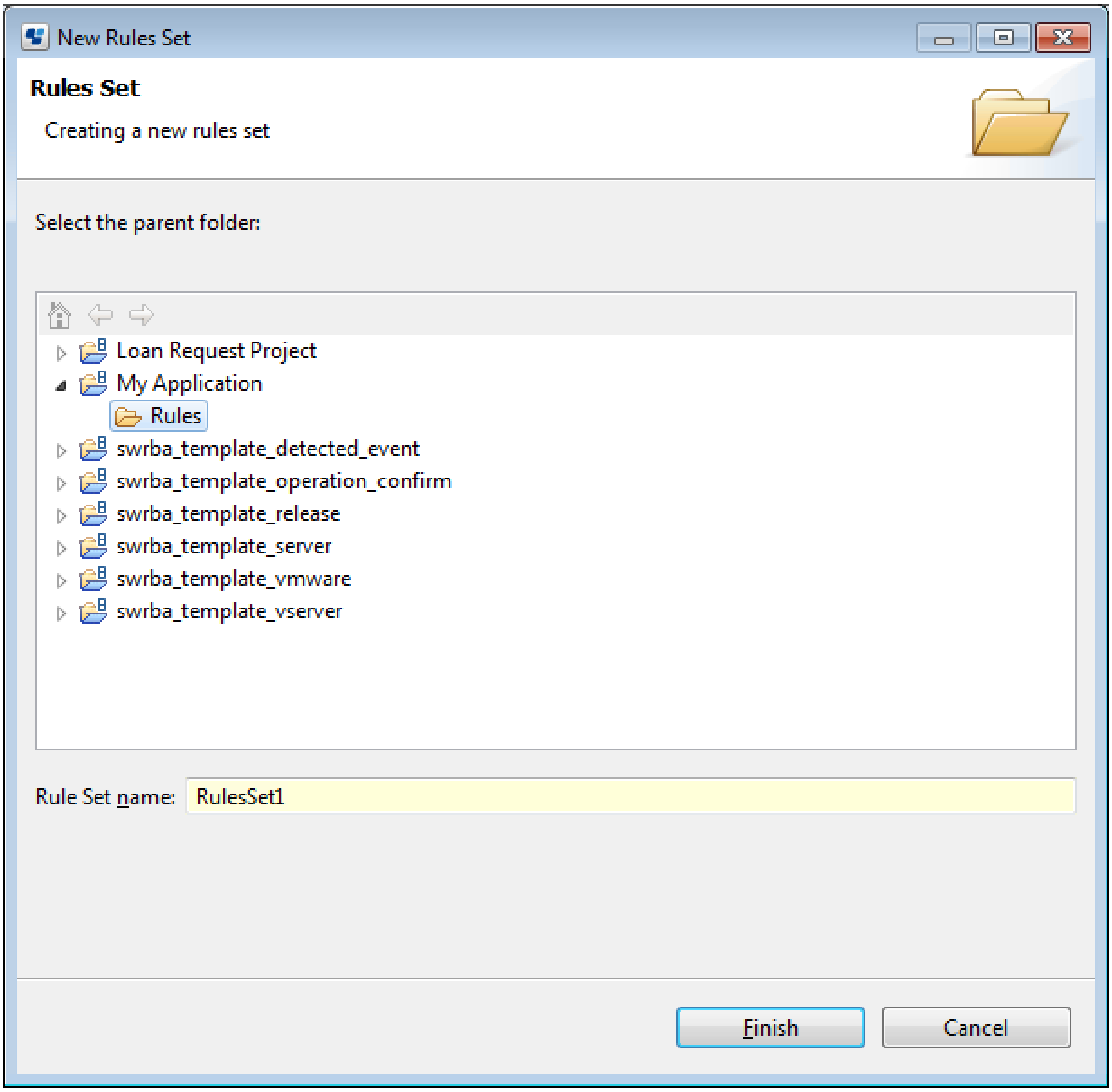
10.2.1 Creating a New Rule Set

To create a new Rule Set:

1. Right click on the Rules folder. Select **New** >> **Rules** >> **Rules Set**. You can also go to File and select **New** >> **Rules** >> **Rules Set**, from the **Windows** menu.

   The **New Rules Set** wizard is displayed.

   Figure 10.2 New Rules Set Wizard

   New Rules Set

   Rules Set

   Creating a new rules set

   Select the parent folder:

   - Loan Request Project
   - My Application
     - Rules
   - swrba_template_detected_event
   - swrba_template_operation_confirm
   - swrba_template_release
   - swrba_template_server
   - swrba_template_vmware
   - swrba_template_vserver

   Rule Set name: RulesSet1

2. Enter the Rule Set name in the **Rule Set name** field and click **Finish**.

   A new **Rules Set** folder is created under the **Rules** folder.

Note

You cannot create a Rules Set folder directly in an application or in another Rules Set folder. Rules Set folders can only be created in the Rules folder.

10.2.2 Creating a New Decision Table

This section explains the procedure for creating a Decision Table file.

Prerequisite

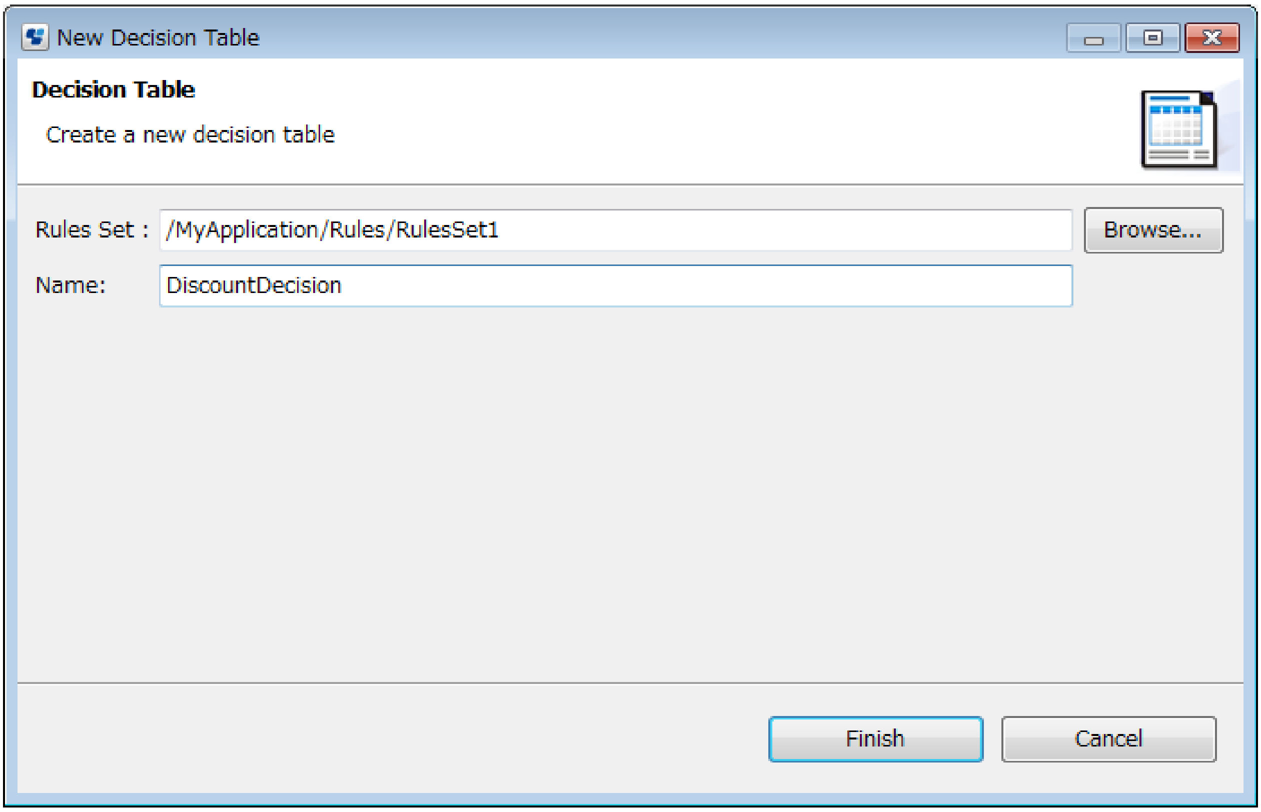
- You have created a Rules Set folder in the Rules folder of the application.

To Create a Decision Table:

1. Right click on the **Rules Set** folder. Select **New** >> **Rules** >> **Decision Table**. You can also go to **File** and select **New** >> **Rules** >> **Decision Table** , from the **Windows** menu.

   The **New Decision Table** Wizard is displayed.

   Figure 10.3 New Decision Wizard

   New Decision Table

   Decision Table

   Create a new decision table

   Rules Set : /MyApplication/Rules/RulesSet1

   Name: DiscountDecision

2. Enter the Decision Table name in the **Name** field and click **Finish**.

   A new rules file is created in **Rules Set** folder and the **Decision Table Editor** is displayed.

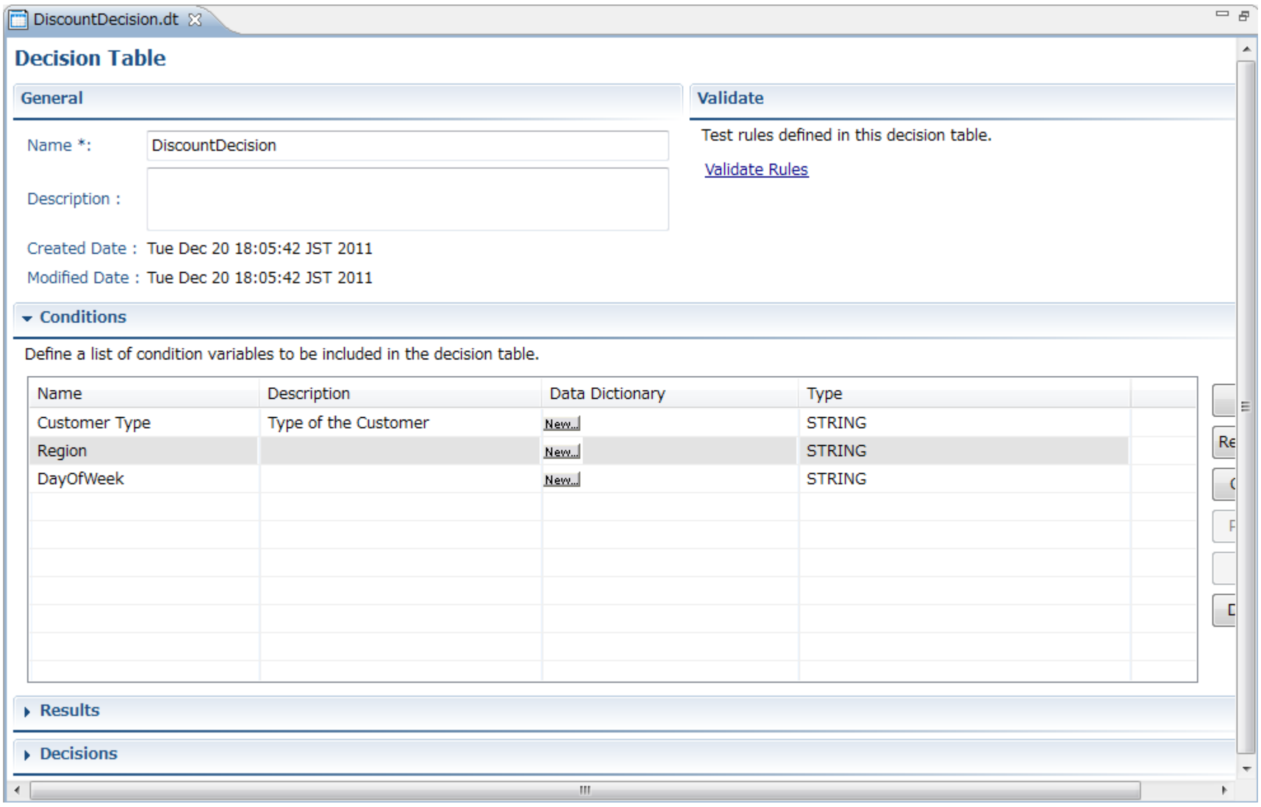
3. Enter a name for your Decision Table, if you have not already entered in the previous step, in the **Name** field and a description in the **Description** field (a description is optional).

Note

- The decision table should be created with .dt extension.
- A Rules Set folder is associated to the application that it is part of, and a .dt file is associated to the Rules Set folder it belongs to. Thus, the Rules Set names should be unique in an Application and the .dt file names should be unique in a Rules Set folder.

The example in the following figure creates a Decision Table called DiscountDecision.dt.

Figure 10.4 Creating a Decision Table



Decision Table

General

Name *: DiscountDecision

Description :

Created Date : Tue Dec 20 18:05:42 JST 2011

Modified Date : Tue Dec 20 18:05:42 JST 2011

Validate

Test rules defined in this decision table.

Validate Rules

Conditions

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
Customer Type	Type of the Customer	New...	STRING
Region		New...	STRING
DayOfWeek		New...	STRING

Results

Decisions

4. Click **Save** in the toolbar.

   Your new Decision Table is saved. To know more about adding and editing Decision Tables, refer to 10.3 Editing a Decision Table.

5. To remove a Decision Table file, right click on the file in the Navigator view of Studio, and select **Delete**.

   You can also import and export Decision Table files and BAR files. See Exporting Decision Table Files and Importing Decision Table Files and Creating Workflow Application Projects0 sections for more information.

Exporting Decision Table Files

You can export saved Decision Tables to a file system from Systemwalker Runbook Automation Studio by performing the following procedure:

1. Right click on the Decision Table file in the **Navigator** view. , Select **Export**. You can also go to **File** and select **Export**.

2. Move the exported Decision Table file to the storage location.

3. Save the file to the local file system.

Importing Decision Table Files

You can import a Decision Table file to Systemwalker Runbook Automation Studio from a file system by performing the following procedure:

1. Right click the Decision Table file or the Rule Set folder, in the **Navigator** view, and select **Import**. You can also go to **File** and select **Import**.
   The **Import Decision Tables** dialog is displayed.

2. Navigate to the Decision Table file (*.dt) that you want to import.

3. Select the file and click **Open**.

4. If a Decision Table file with the same name already exists in RuleSet1, a dialog box that prompts you to enter a file name appears. Enter a file name and click **OK**.

10.3 Editing a Decision Table

To add a decision to the Decision Table, add condition and result variables and map them. These actions have been explained in the subsequent sections.

- 10.3.1 Adding or Editing Conditions
- 10.3.2 Adding or Editing Results
- 10.3.3 Adding or Editing Decisions

10.3.1 Adding or Editing Conditions

This section explains the procedure for editing conditions.

Prerequisite:

- You have created a Decision Table.

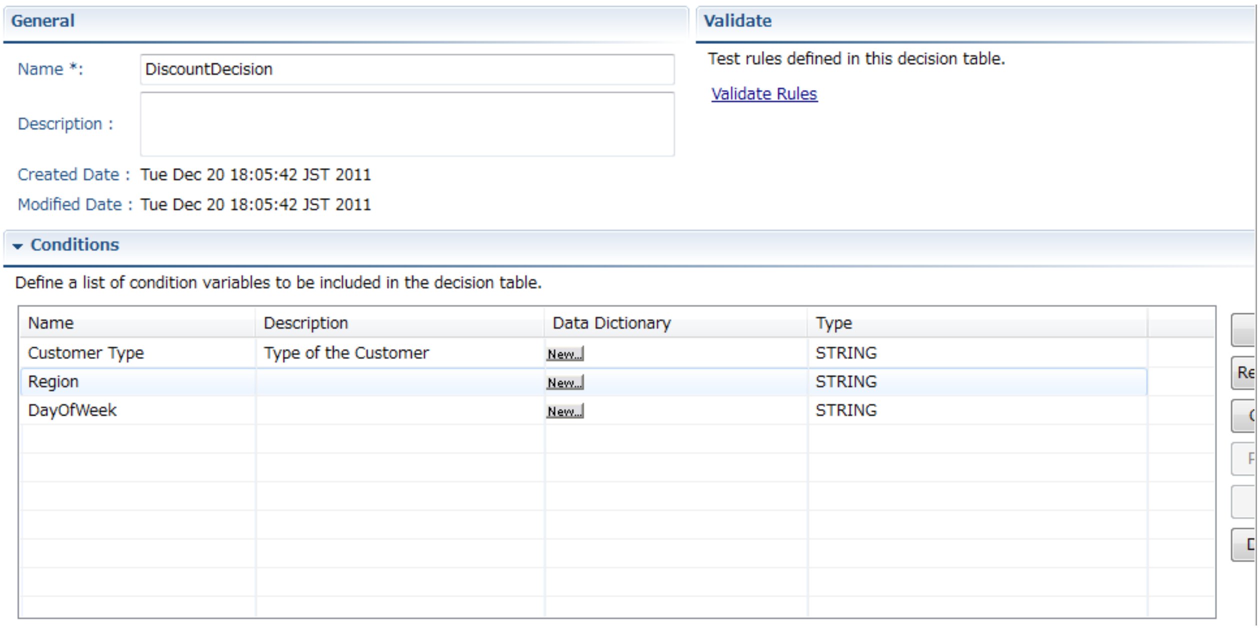
To add a condition to the Conditions list:

1. Double-click the Decision Table file from **Rules** folder of the application. The Decision Table is located in the corresponding **Rule Set** folder.
2. To add a new condition to the Decision Table, in the **Condition** section of the **Decision Table Editor**, click **Add**.
   A new condition with the default name, Condition0 is added to the Condition table.
3. Click on the default name of the condition to change it.
4. Enter a description (optional) for the condition in the corresponding **Description** column.
5. Enter information about substitute mapping in the **Data Dictionary** column. See *Data Dictionary* for information on how to add substitute values for user inputs.
6. Select a data type for the condition from the **Type** column. Click the down arrow in the corresponding row, to display type field options.

7. Click **Save** from the tool bar to save the condition.

   The figure below demonstrates the addition of conditions called CustomerType, Region, and DayOfWeek of data type STRING.

Figure 10.5 Adding a Condition



General

Name *: DiscountDecision

Description :

Created Date : Tue Dec 20 18:05:42 JST 2011

Modified Date : Tue Dec 20 18:05:42 JST 2011

Validate

Test rules defined in this decision table.

Validate Rules

Conditions

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
Customer Type	Type of the Customer	New...	STRING
Region		New...	STRING
DayOfWeek		New...	STRING

8. To remove a condition, select it in the **Conditions** list and click **Remove**.

9. To copy the condition, select it in the **Conditions** list and click **Copy**.

10. To paste the condition, copy the condition as described in the above step. Select the row above which you want to paste the condition and click **Paste**. If you do not specify a row, the copied condition is pasted below the last condition.

11. You can also move the conditions up and down using the **Up** and **Down** buttons. To do this, select the condition that you want to move and click **Up** or **Down** according to your requirement.

Note

You can copy, move up, move down, and remove multiple conditions by holding down the Shift or Ctrl keys and selecting multiple rows.

You can now add results to your Decision Table. See 10.3.2 Adding or Editing Results section for more information.

Data Dictionary

Decision Tables also support the use of a custom dictionary for value mapping. You can define synonyms of values.

To use value mapping:

1. In the **Conditions** section of the **Decision Table Editor**, for a given condition, click the corresponding Data Dictionary cell.

   The Data Dictionary dialog is displayed. The condition information is displayed in the **Condition Details** section of the dialog.

2. Enter the UDA value in the **Input Value** field.

3. Enter the value that you want to substitute for your UDA Value in the **Substitute Value** field.

4. Click **Add Mapping**.

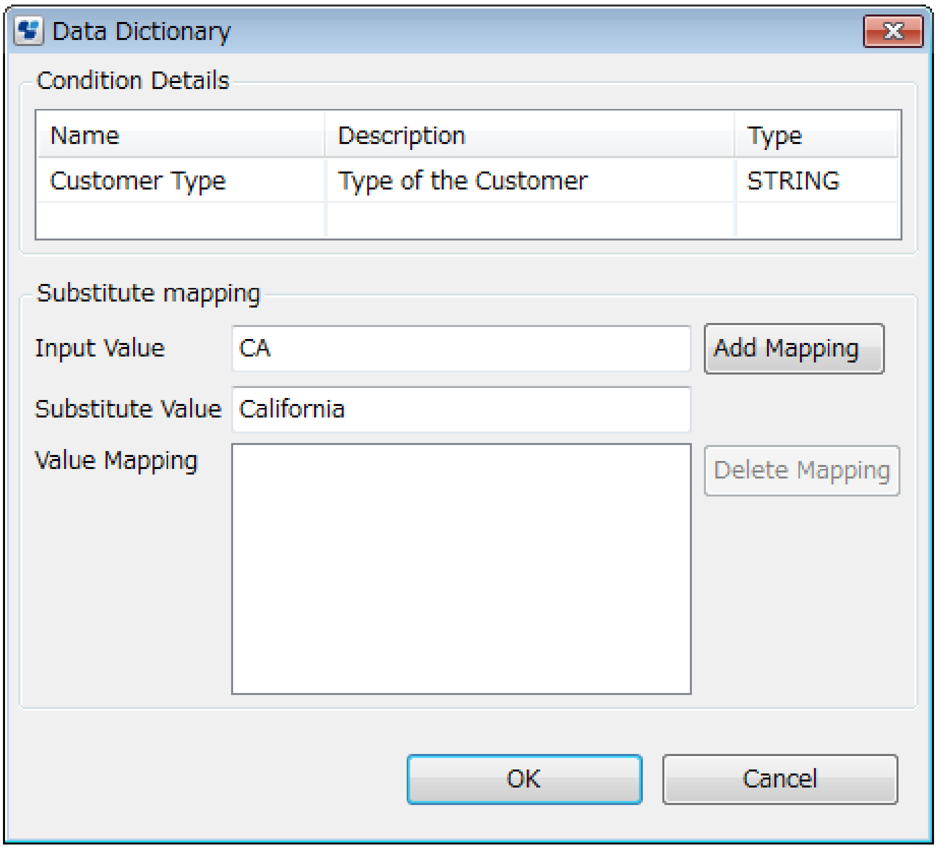
The mapping appears in the **Value Mapping** list.

The substitute value that you entered is used in your condition in place of the Input Value that you enter.

Example: Substituting California for CA

If you entered CA as your input value and California as your substitute value, then clicked **Add Mapping** (see figure below), California will be the value of the condition when CA is the value of the UDA.

Figure 10.6 Substitute Mapping CA to California



Data Dictionary

Condition Details

Name	Description	Type
Customer Type	Type of the Customer	STRING

Substitute mapping

Input Value: CA

Substitute Value: California

Value Mapping:

10.3.2 Adding or Editing Results

This section explains the procedure for editing results.

Prerequisites:

- You have created a Decision Table.

  See 10.2.2 Creating a New Decision Table for information on creating a Decision Table file and added conditions to your Decision Table.

A result is generated by a Decision Table if the criterion specified in one of its decisions is satisfied with respect to its conditions.

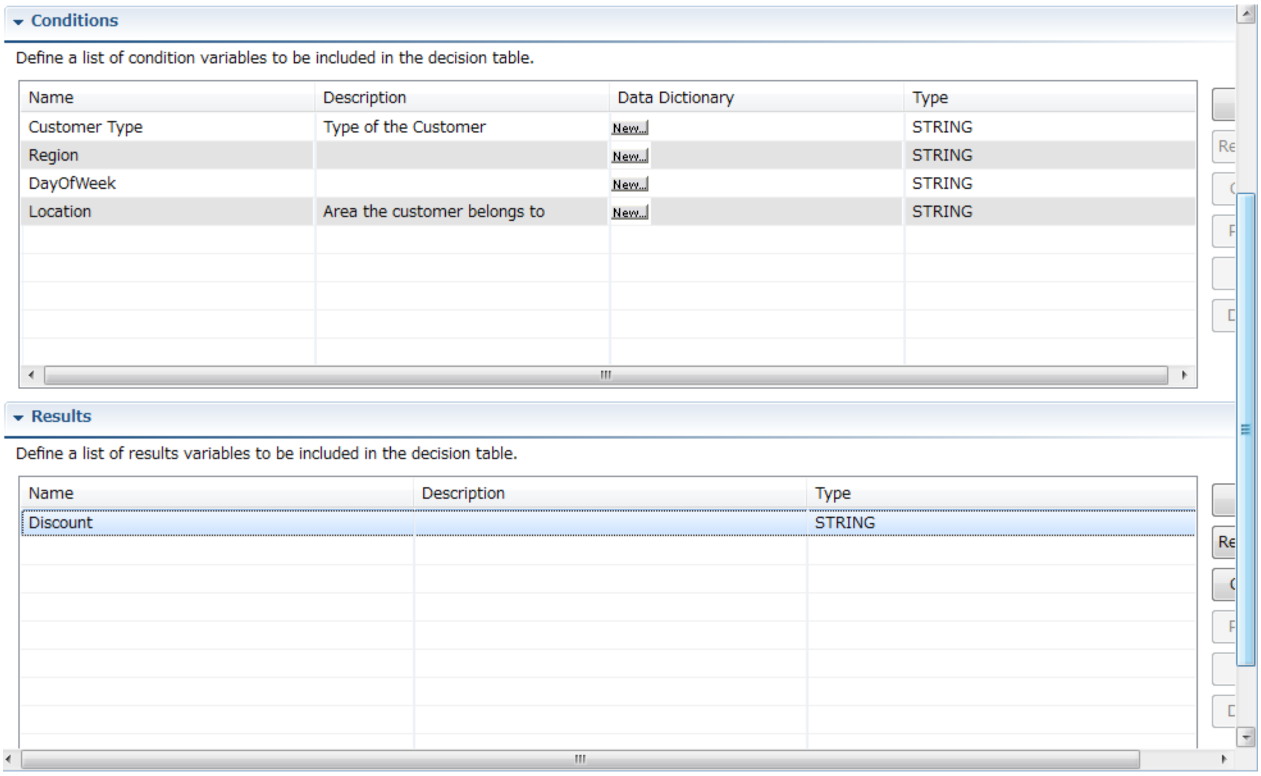
To add a result to the Results List:

1. Double click the Decision Table file from **Rules** folder of the application. The Decision Table file is located in the corresponding **Rules Set** folder.
2. To add a new result to the Decision Table, in the **Results** section of the **Decision Table Editor**, click **Add**.
   A new result with the default name, Result0 is added to the Result table.
3. Click on the default name of the result to change it.
4. Enter a description (optional) for the result in the corresponding **Description** column.

5. Select a data type for the result from the **Type** column. Click the down arrow in the corresponding row, to display type field options.

   The figure below demonstrates the addition of a result called Discount.

Figure 10.7 Adding a Result



Conditions

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
Customer Type	Type of the Customer	New...	STRING
Region		New...	STRING
DayOfWeek		New...	STRING
Location	Area the customer belongs to	New...	STRING

Results

Define a list of results variables to be included in the decision table.

Name	Description	Type
Discount		STRING

6. To remove a result, select it in the **Results** list and click **Remove**.

7. To copy the result, select it in the **Results** list and click **Copy**.

8. To paste the result, copy the result as described in the above step. Select the row above which you want to paste the result and click **Paste**. If you do not specify a row, the copied result is pasted below the last result.

9. You can also move the results up and down using the **Up** and **Down** buttons. To do this, select the result that you want to move and click **Up** or **Down** according to your requirement.

Note

You can copy, move up, move down, and remove multiple results by holding down the Shift or Ctrl keys and selecting multiple rows.

10.3.3 Adding or Editing Decisions

Please note that before executing this procedure, a Decision Table file must have been created, and the necessary conditions and results must have been defined.

This section explains the procedure for editing a Decision Table.

1. Double click the Decision Table file from **Rules** folder of the application. The Decision Table file is located in the corresponding **Rules Set** folder.

2. To add a new decision to the Decision Table, in the **Decision** section of the **Decision Table Editor**, click **Add**.

   A new row will be added with incremental serial number in the **No.** column of the Decisions table. If this is the first decision you are adding to the Decision Table the numeral 1, is displayed in the **No.** column.

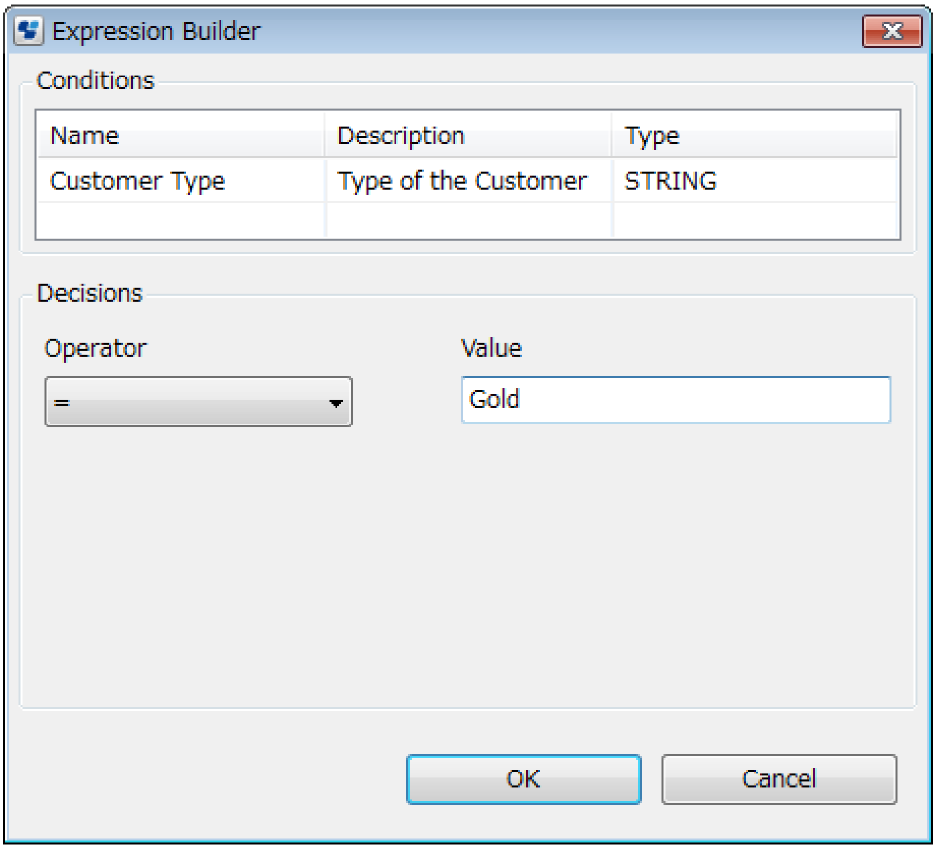
3. The conditions and results that you added previously, are displayed as separate columns in the **Decisions** section.

4. Click the cell corresponding to a condition. A text area followed by the **Expression Builder** button is displayed. Click the button.

   The **Expression Builder** dialog is displayed.

5. Select the operation that you want to apply for the condition from the **Operator** drop-down.

   Figure 10.8 Expression Builder



Conditions

Name	Description	Type
Customer Type	Type of the Customer	STRING

Decisions

Operator: =

Value: Gold

The Expression Builder can be used to generate an expression for the condition, using the following comparison operators.

- <= (less than or equal to)
- < (less than)
- = (equal)
- != (not equal)
- > (greater)
- >= (greater than or equal to)
- in
- between
- like
- notlike

Note

- You can also modify the condition and result values directly in the Decision table. To do this, click on the cell in the condition or results column and enter the expression in the text area. An error is displayed in case the expression is entered in a wrong format.

- For the operators =,!=,<,>,<=,>= , the format is operator followed by the value. For example, =xyz. For operators in,like,not like, between the format is operator(value1,value2...). For example, in(a,b,c).

The following table lists different operators supported for specific data types.

Operator	Boolean	Integer	Float	Long	BigDecimal	Date	String
<	No	Yes	Yes	Yes	Yes	Yes	No
>=	No	Yes	Yes	Yes	Yes	Yes	No
<=	No	Yes	Yes	Yes	Yes	Yes	No
>=	No	Yes	Yes	Yes	Yes	Yes	No
!=	Yes	Yes	Yes	Yes	Yes	Yes	Yes
=	Yes	Yes	Yes	Yes	Yes	Yes	Yes
in	No	Yes	Yes	Yes	Yes	Yes	Yes
between	No	Yes	Yes	Yes	Yes	Yes	No
like	No	No	No	No	No	No	Yes
notlike	No	No	No	No	No	No	Yes

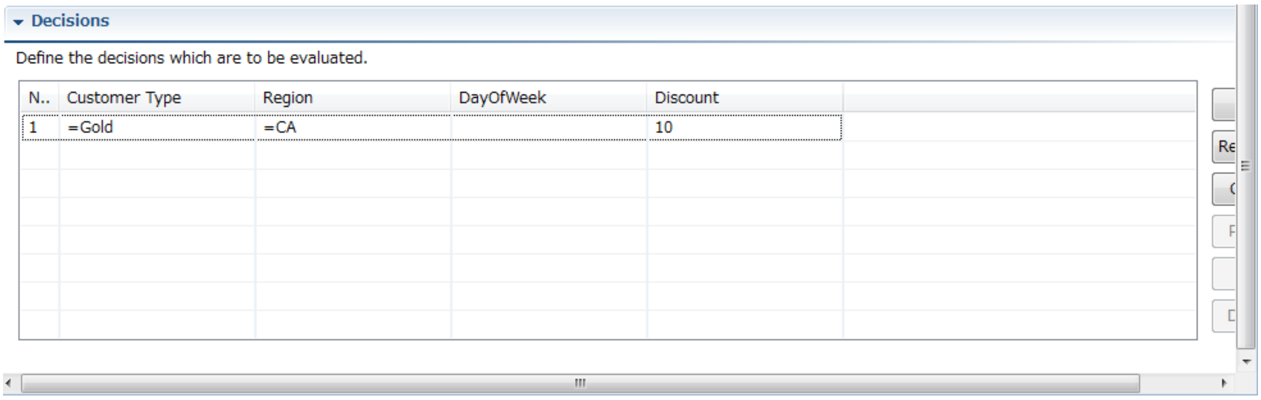
6. Click **OK**.

   The expression is displayed in the Decision Table Editor, corresponding to the condition.

7. In the Discount column, enter the value for the decision, if a given condition is true.

   For example, the decision in the below screen shows if CustomerType = Gold, Region = CA, then Discount = 10.

Figure 10.9 Adding a Decision



Decisions

Define the decisions which are to be evaluated.

N..	Customer Type	Region	DayOfWeek	Discount
1	=Gold	=CA		10

When specifying multiple conditions, you can leave the value of one of the conditions blank. A blank value is a wildcard that matches any value. The decision displayed in the figure above executes with any DayOfWeek

Note

Blank values can be provided as input values for any condition, even for conditions of type BOOLEAN. Blank input values are treated as wildcards.x

At run time, This product evaluates all of the defined decisions in the order shown on the Decision Table Details page. When defining decisions, ensure that the order makes sense. If multiple decisions apply, the result of the decision that is evaluated first is finally used. As an example, say you define the following discount rules:

- Decision 1: For a sales amount > $50000, a discount rate of 20% applies.
- Decision 2: For a sales amount > $20000, a discount rate of 10% applies.

If a process instance has a sales amount of $70000, both decisions match. But as soon as the first discount rule is evaluated, the discount rate is set to 20%. The second Decision is ignored. If you want to add a decision with values of all conditions as blank, and if it is the first decision to be evaluated then that decision will be used. The other decisions will be ignored.

To validate decision rules, see 10.5 Validating Decision Table Rules .

To know about mapping UDAs, conditions and results to a Process Definition using JavaAction, see 10.6 Using the Decision Table Action .

10.4 Validating Decision Tables for Errors and Warnings

Decision Table files can be validated by using any of the following methods:

- Using the Validate icon: Click the **Validate** icon to validate the Decision table. This icon is located near the **New** icon below the **File** menu. The errors and warnings in the Decision Table are displayed in the **Problems View** tab below the **Decision Table** editor.
- Using the Validate Rules link: Click this link to validate the Decision Table. Refer to 10.5 Validating Decision Table Rules for more information about **Validate Rules** link. The **Test Decision Tables** dialog box is displayed in which the Decision Table can be validated.
- Saving the Decision Table: Click **Save** or **Save As** from **File** menu. In this scenario, validation of the Decision Table is done before saving it. If the Decision Table file is valid; it is saved. If it is not valid then an error message "Decision Table file is not valid. Would you like to save it anyway?" is displayed. If you click **Yes** on the error message dialog, the Decision Table is saved with errors and warnings and they are displayed in the **Problems View** tab. If you click **No**, the **Problems View** tab is restored to its earlier state, that is, the state before saving the Decision Table and the file is not saved. If you click **Details**, the errors and warnings in the Decision Table are displayed below the error message.
- **Closing the Decision Table**: Click **Close** to close the Decision Table. If you modify the Decision Table and do not save the changes before closing it, the **Save Resource** dialog is displayed with a message "<Decision Table Name> has been modified. Save changes?". If you click **Yes** on the dialog, an error message "Decision Table file is not valid. Would you like to save it anyway?" is displayed if the Decision Table is not valid. All the three options on this dialog have been explained in **Saving the Decision Table** above. If you click **No** on the **Save Resource** dialog, the file is not saved and closed immediately. If you click **Cancel** on it, the Decision Table is neither saved nor closed.

The errors and warnings are displayed in the **Problems View** tab. Double clicking on the error row selects the area in the Decision Table editor where the error has occurred. The error can be corrected in this view. Refer to 2.2.10 Problems View for more information about the **Problems View**. The figure below displays the **Problems View** and the errors and warnings in the Decision Table.

10.5 Validating Decision Table Rules

This section explains the procedure for validating Decision Tables.

Prerequisite:

- You have created a Decision Table file and defined your conditions, results, and decisions in it.

To test a Decision Table:

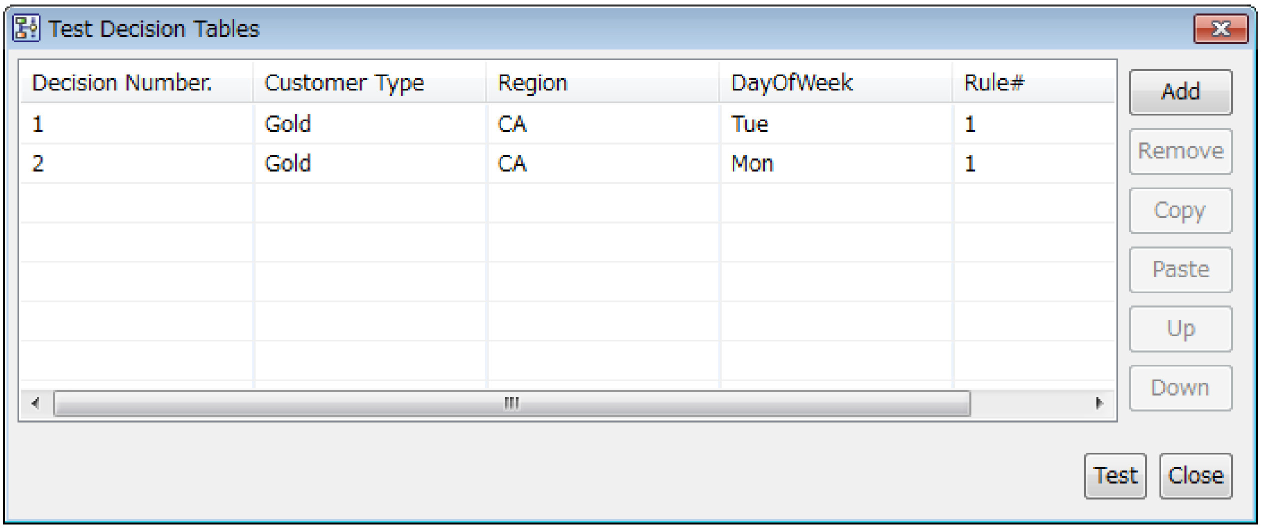
1. Ensure that the **Decision Table Editor** of the Decision Table that you want to test is displayed.
2. Click **Validate Rules**.

   The **Test Decision Tables** dialog box opens if no errors exist in the Decision Table. It opens if warnings exist or do not exist in the Decision Table.
3. Click **Add** and enter values in each of the input fields that correspond to the values that you expect from the UDAs, that are mapped through conditions.

4. Click **Test**.
   The line number of the decision to use is displayed in the **Rule#** column.

   The figure below displays the results of a test on the DiscountDecision Decision Table.

Figure 10.10 Results of Test on DiscountDecision Decision Table



Decision Number.	Customer Type	Region	DayOfWeek	Rule#
1	Gold	CA	Tue	1
2	Gold	CA	Mon	1

10.6 Using the Decision Table Action

A pre-built Java Action is included to facilitate the use of Decision Tables. These actions have been explained in the subsequent sections.

- 10.6.1 Selecting a Process Definition to Contain Your Decision Table
- 10.6.2 Assigning Decision Tables to Java Action

10.6.1 Selecting a Process Definition to Contain Your Decision Table

A Decision Table is useful only if it is run as a component of the Systemwalker Runbook Automation process instance. The definition for that process instance must contain the following components along with the Decision Table inside a Decision Table Action for a Decision Table to work correctly:

- Input User Defined Attributes (UDAs) corresponding to those defined as conditions in the Decision Table
- Output UDAs corresponding to those defined as results in the Decision Table

Note

If the necessary UDAs are not defined in your process definition, the Decision Table Action may work unpredictably. For example, the method that you want to use may be unavailable. A method will be available only if all the UDAs that it uses are defined. If the process definition UDAs are mismatched with the Decision Table conditions or results, error messages will appear.

10.6.2 Assigning Decision Tables to Java Action

This section explains the procedure for defining the Decision Table action.

Prerequisites:

- You have created a Decision Table and defined your conditions, results, and decisions in it.
- The process definition where the Decision Table action is defined contains required User Defined Attributes (UDAs).

To integrate a Decision Table:

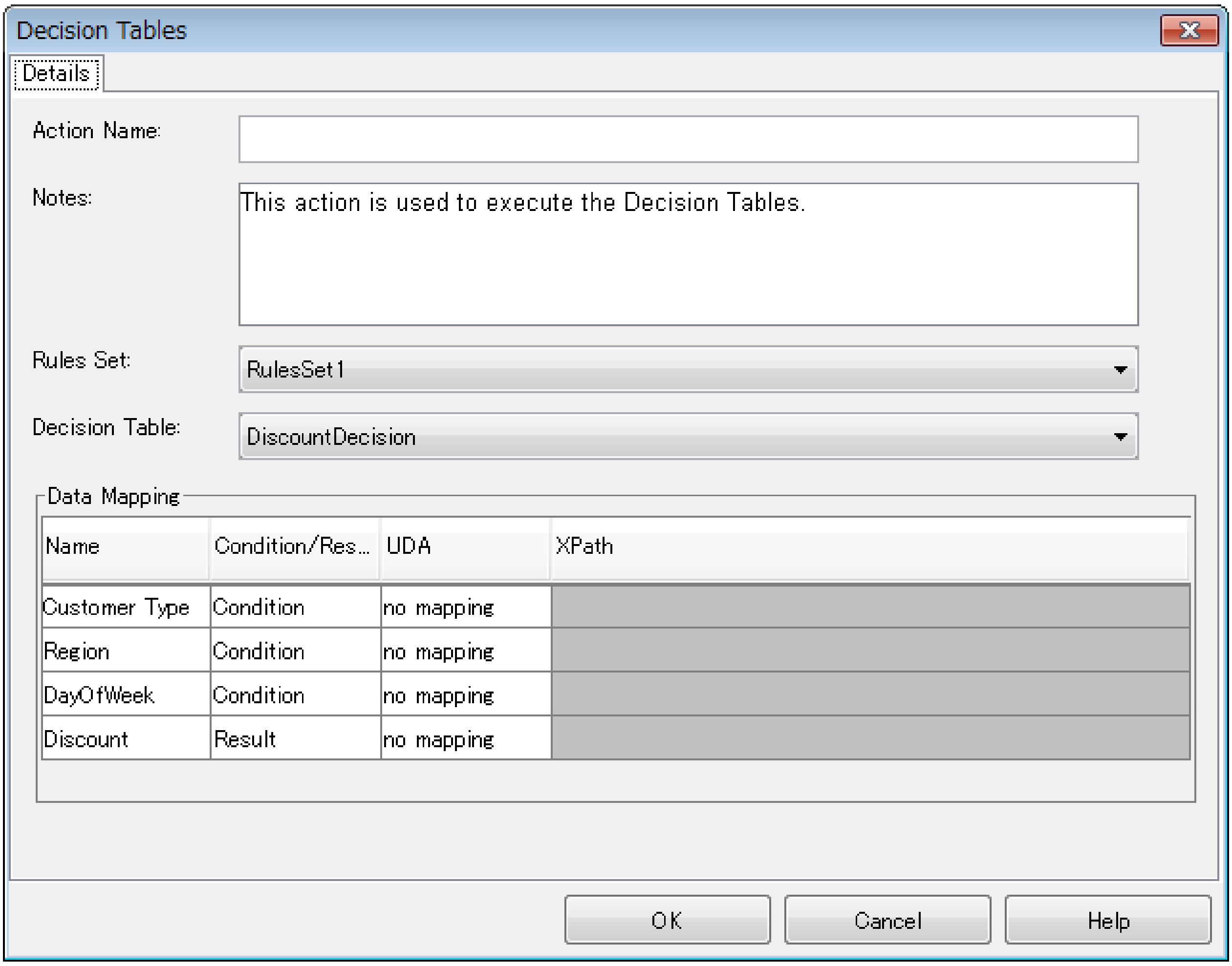
1. Open the process definition that will contain your Decision Table by double clicking the .XPDL from the **Process Definitions** folder.

2. To associate your Action with an activity, select the **Action Set** option from **Properties** view.
3. Click **Add** in the corresponding **Role Actions**, **Epilogue Actions**, or **Prologue Actions** tabs, where you want to add the Java Action. The **Action Type List** dialog is displayed.
4. Expand **Rules Actions** and double click **Decision Tables**.
   The **Action editor - Set Rules** dialog is displayed.
5. Type a descriptive name and notes for the Java Action in **Action Name** and **Notes** fields respectively.
6. Select the **Rules Set** to which your Decision Table belongs.

   The **Rules Set** drop-down list displays all the available **Rules Sets** for the application.
7. Select the Decision Table that you want to use.

   The **Decision Table** drop-down list displays all the Decision Tables in the selected **Rules Set**.

Figure 10.11 Integrating Decision Tables



Decision Tables

Details

Action Name:

Notes: This action is used to execute the Decision Tables.

Rules Set: RulesSet1

Decision Table: DiscountDecision

Data Mapping

Name	Condition/Res...	UDA	XPath
Customer Type	Condition	no mapping	
Region	Condition	no mapping	
DayOfWeek	Condition	no mapping	
Discount	Result	no mapping	

8. In the **Data Mapping** table, click in the **UDA** column and select UDAs that you want to assign to the conditions and results, from the drop-down list.

Note

- By default, no mapping is displayed in the **UDA** column. When you click in it, drop-down list of the UDAs is displayed. XML UDAs are displayed in the drop-down list in **UDA** column irrespective of the type of the **Condition** or **Result** provided the process definition has XML UDAs. For example, if a Condition is of String type then the UDA drop-down list for that Condition will also display the XML UDAs even if the type of the Condition is String.

- When you select a UDA in the **UDA** column of the **Data Mapping** table, you need to click inside any other field of the same table in order to confirm the selection of the UDA for assigning it to the respective condition and result.

- Mapping UDAs to Conditions is mandatory but mapping UDAs to Results is optional. If you do not map UDAs to Conditions and click **OK**, an error message is displayed. If you do no map UDAs to Results and click **OK**, a warning message is displayed. You can click **OK** on the warning message and still assign Decision Tables to the Java Action.

9. If the UDA is of type XML then, click in the field in the **XPath** column and select the XPath expression for the UDA from the drop-down list.

Note

- An error message is displayed if XPath is not provided for UDA of type XML.

- The XPath expressions related to the selected XML UDA are displayed in the **XPath** drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

10. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the ellipsis (**...**) button that is displayed next to the drop-down list. **XPath Editor** dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

Note

- The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

- **XPath Editor** only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

11. Click **OK** to close the **Action editor** dialog.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

Chapter 11 Advanced Process Modeling

This chapter explains how to use Java Actions and triggers. It also provides information about how to define JavaScript expressions.

11.1 Using Java Actions

Using Java Actions, you can customize process execution and integrate with external systems.

Systemwalker Runbook Automation Studio provides the following types of built-in Java Actions:

- Notification Actions
- Server Actions
- Rules Actions
- XML Actions

It also provides the following functions:

- A mechanism to integrate methods of your own Java classes as Generic Actions
- A No-Operation Action that is typically used for specific error situations

You can assign Java Actions to the process definition itself, to nodes, timers, and due dates. Start Nodes and Exit Nodes cannot have Java Actions assigned.

When assigning a Java Action, you always assign it to a so-called Action Set. Each Action Set is executed at a particular point of time during process execution or upon execution of specific commands.

11.1.1 Types of Java Actions

The following Java Action Sets can be distinguished:

- **Init Actions** and **Process Owner Actions** are executed upon process initialization. These Java Actions initialize User Defined Attribute data before the first activity is performed.
- **Commit Actions** are executed upon process completion. They can be used to clean up or analyze the data of an entire process instance.
- A **Prologue Action** is evaluated before an activity starts. This Java Action can therefore be used to set up or initialize values associated with a specific node before it does its work.
- An **Epilogue Action** is executed after a node finishes its task and before the process instance moves on to another node. This Java Action can therefore be used to clean up or analyze values associated with the node after the intended work is finished.
- A **Role Action** is evaluated after resolving a role and before assigning a task. This Java Action is therefore used to dynamically compute a list of assignees for a task in conjunction with a Role.
- **Timer Actions** are executed when a timer expires or a due date is reached.
- An **Error Action** can be used for handling specific error situations in the execution of a process. Error Actions can be defined for entire process definitions, for individual nodes and for other Java Actions, i.e. when you want to react on errors occurring during the execution of another Java Action.
- A **Compensation Action** can be defined for another Java Action that accesses a system outside of Systemwalker Runbook Automation, e.g. an external database. Compensation Actions are useful to ensure a consistent state of all systems involved in a transaction for cleaning up and rolling back transactions, e.g. to delete a newly added row in an external database.
- A special type of action can be activated as soon as an Administrator issues the command to abort, suspend or resume the processing of a process instance. Such actions are stored in either of the following Action Sets:
  - **onAbort Action**
  - **onResume Action**
  - **onSuspend Action**

These Java Actions are to be performed before the state of a process instance is changed. They can be defined for individual activities, i.e. for individual nodes in a process definition, or for an entire process definition.

11.1.2 Assigning Java Actions

Java Actions can be defined and assigned on three different levels:

- **Process definition level**: Init Actions, Owner Actions, Commit Actions, Timer Actions, OnSuspend, OnResume, and OnAbort Actions, Error Actions.
- **Node level**: Role Actions, Prologue and Epilogue Actions, OnSuspend, OnResume, and OnAbort Actions, Error Actions, and Timer Actions. The node type determines which type of Java Action you can assign and at which point in time a Java Action is to be executed.
- **Java Action level**: Compensation and Error Actions.

When you define Java Actions both on Node level and on process definition level, first the Node level actions, second the process definition level actions will be executed.

You can create a Java Action once and then use it in different contexts. For example, you can use the same Java Action as Prologue Action, Error Action, and Compensation Action.

Note

To handle specific error situations, Systemwalker Runbook Automation Studio provides Error Actions and Compensation Actions. You can assign these actions to any other Java Action. For more information on Error and Compensation Actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

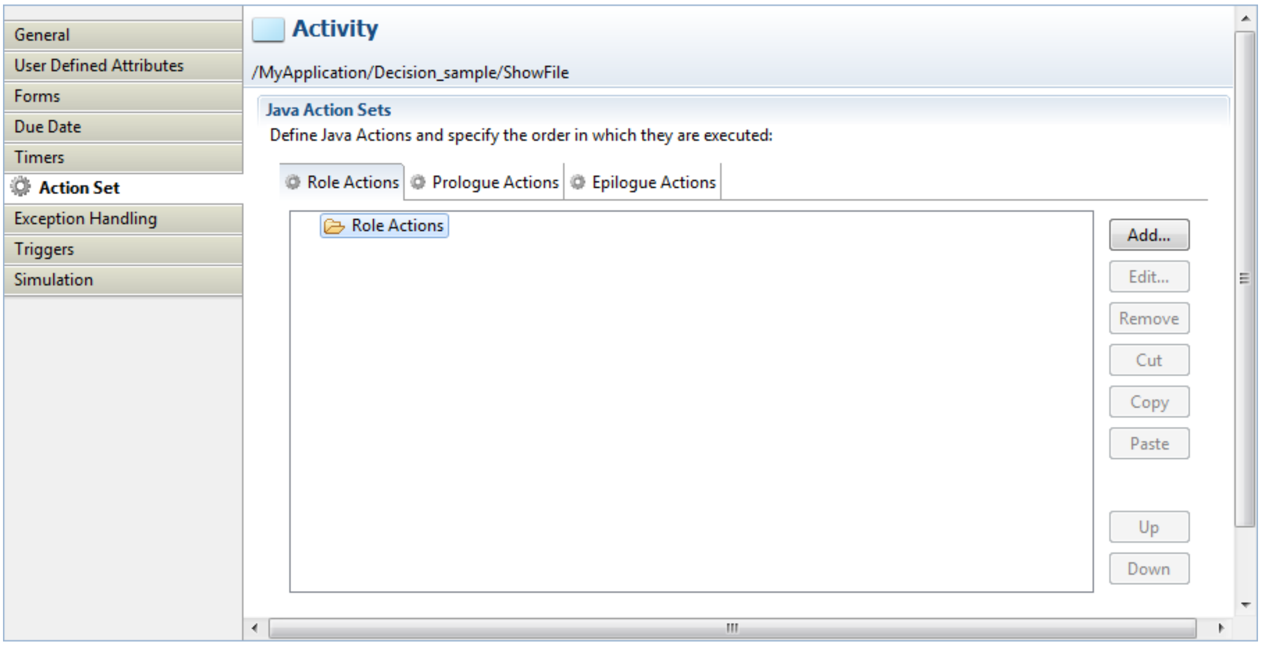
This section gives you an overview of the general steps required for assigning Java Actions.

To assign a Java Action to a process definition or a node:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the Properties view for the process definition or the node respectively.
2. Select the **Action Set** tab.

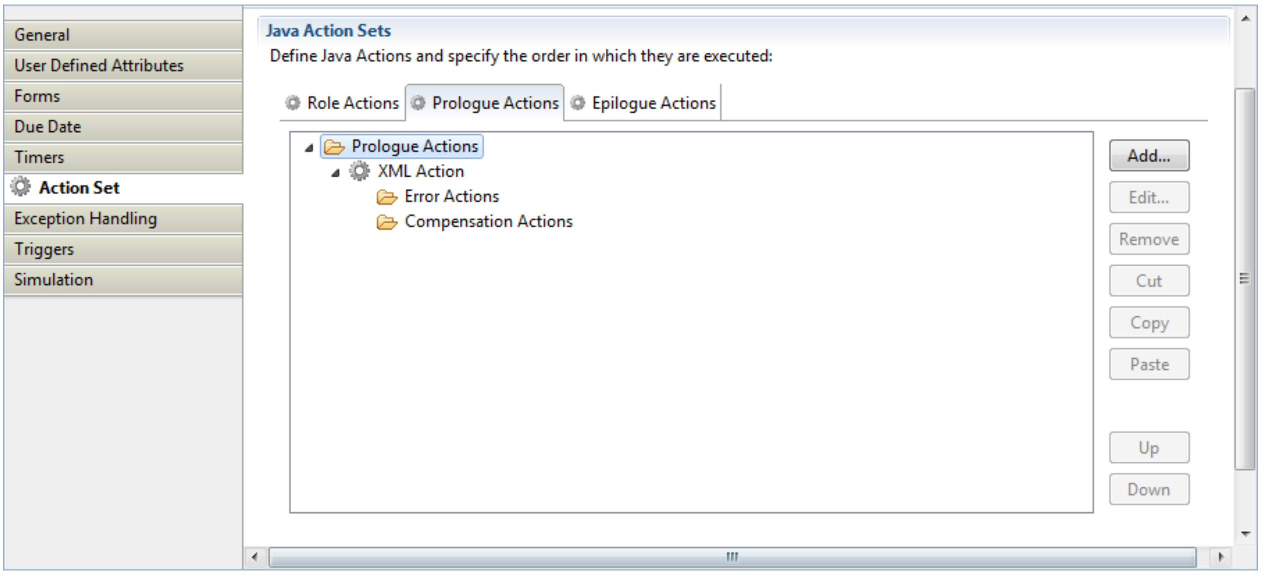
   The content of the **Action Set** tab depends on the item for which you open it. The following figure shows the **Action Set** tab for an Activity Node.

Figure 11.1 Action Set Tab for Activity Nodes



The following figure shows the **Action Set** tab for a process definition.

Figure 11.2 Action Set Tab for Process Definitions



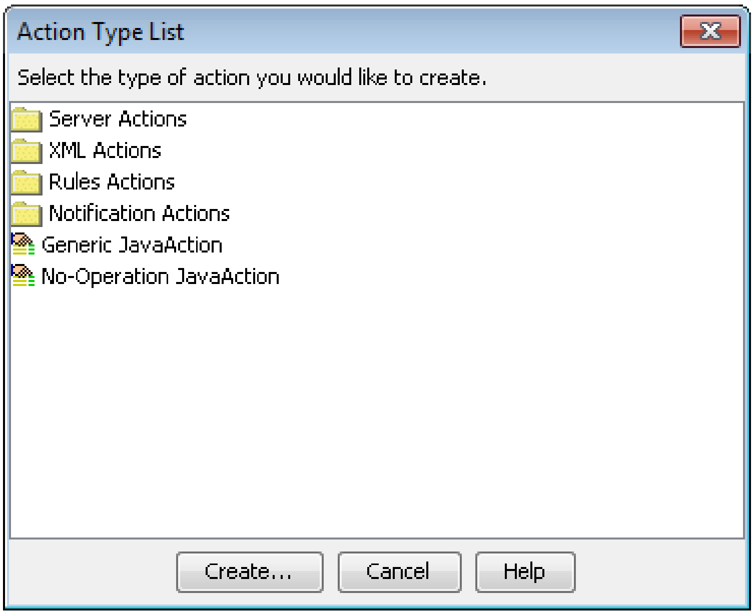
Note

You can also assign Java Actions to timers and due dates, using the **Timers** and **Due Dates** tabs. For more information, refer to sections 6.22.1 Defining Due Dates and 6.22.2 Defining Timers.

3. Specify which type of Java Action you want to add by selecting the corresponding tab, and click **Add**.

   The **Action Type List** dialog is displayed.

   Figure 11.3 Action Type List

4. In the **Action Type List** dialog, expand the folder where the Java Action to be added is located. Select the Java Action, and click **Create**.

   Perform operations required depending on the action you selected.

5. Click **OK**.

6. You can rearrange the order in which Java Actions are executed by highlighting the Java Action and clicking the **Up** or **Down** button.

Note

You cannot use the **Up** and **Down** buttons to move a Java Action to a different Action Set. To do this, use cut, copy, and paste (for more information, refer to section 11.1.4 Cutting, Copying and Pasting Java Actions).

11.1.3 Editing Java Actions

You can edit Java Actions that you have assigned to the process definition or to individual nodes.

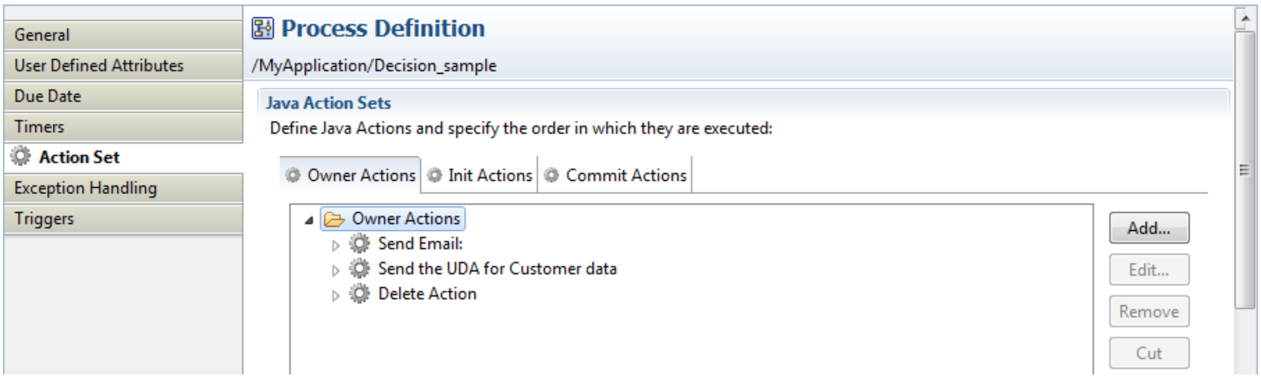
To edit a Java Action:

1. Click the empty space in the Process Definition editor or select the node that has the Java Action assigned, to display the Properties view for the process definition or the node respectively.

2. Select the **Action Set** tab.

   The **Properties** view shows all Java Actions that have been assigned to the Action Set. The following example shows a number of Java Actions assigned to the Owner Action Set.

Figure 11.4 Displaying the Owner Action Set



3. Select the Java Action that you want to change. Click **Edit** next to it.

   A dialog is displayed where you can make your changes. In the dialog box displayed, make changes required depending on the action you selected.

4. Click **OK** to close the dialogs.

11.1.4 Cutting, Copying and Pasting Java Actions

You can move or copy Java Actions to another position using cut, copy, and paste.

To use cut, copy and paste a Java Action:

1. Click the empty space in the Process Definition editor or select the node that has the Java Action assigned, to display the Properties view for the process definition or the node respectively.

2. Select the **Action Set** tab.

3. To cut a Java Action:

   - Select the Java Action and click **Cut**, OR:

   - Right click the Java Action and select **Cut** from the pop-up menu.

4. To copy a Java Action:

   - Select the Java Action and click **Copy**, OR:

   - Right click the Java Action and select **Copy** from the pop-up menu.

5. To paste the Java Action, right click the desired Action frame and select **Paste** from the pop-up menu.

   If you select a folder, the Paste function will paste inside this folder. If you select a Java Action, the Paste function will paste after the selected Java Action.

Pasting Regular, Error and Compensation Actions

You can copy Java Actions from one Action frame and paste them into another frame. However, it is not always useful to paste a certain action into another action (for example, pasting an Error Action into a Compensation Action). As an example, exception qualifiers would be meaningless on a Compensation Action. Consequently, the **Paste** function automatically looks to see if there are any exception settings when pasting to a location that does not use them; any information which will be useless at the pasted location will be lost.

The following table shows what happens if you paste regular Java Actions, Error Actions, and Compensation Actions.

Position to paste Java Action -> Java Action to be copied	Regular Java Action	Error Action	Compensation Action
Regular Java Action	A regular Java Action is pasted together with the Error and Error and Compensation Actions assigned to it.	An Error Action is pasted. The default Error Action settings (rollback of transaction, error state) are used. All Error or Compensation Action settings in the copied regular Java Action are lost.	The Error Action is pasted. The default Error Action settings (rollback of transaction, error state) are applied. All Error or Compensation Action settings in the copied regular Java Action are lost.
Error Action	A regular Java Action without Error and Compensation Actions is pasted. All error handling settings defined in the **Error Handling** tab will be lost (refer to section 11.1.7 Dealing With Errors in Java Actions. All other settings of the copied Error Action are successfully pasted.	An Error Action is pasted.	A Compensation Action is pasted. All error handling settings defined in the **Error Handling** tab will be lost (refer to section 11.1.7 Dealing With Errors in Java Actions. All other settings of the copied Error Action are successfully pasted.
Compensation Action	A regular Java Action without Error and Compensation Actions is pasted.	An Error Action is pasted. The default Error Action settings (rollback of transaction, error state) are used.	A Compensation Action is pasted.

11.1.5 Removing Java Actions

If you no longer want to use a Java Action, you can remove it from the process definition or from a node.

To remove a Java Action:

1. Click the empty space in the Process Definition editor or select the node that has the Java Action assigned, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab.
3. Select the Java Action that you want to remove. Click **Remove** next to it.
4. Click **OK**.

11.1.6 Specifying Transaction Settings

Java Actions are assigned to so-called Action Sets. Action Sets are associated with particular points of time within process execution:

- **Init Actions**: When the process instance starts
- **Process Instance Owner Actions**: When a process instance starts
- **Role Actions**: When Systemwalker Runbook Automation determines who to assign the activity to
- **Prologue Actions**: Before an activity is activated
- **Timer Actions**: When a timer expires or a due date is reached
- **Epilogue Actions**: When an activity is completed
- **Commit Actions**: When the process instance ends
- **OnSuspend Actions**: When a process instance is suspended
- **OnResume Actions**: When a process instance is resumed
- **OnAbort Actions**: When a process instance is aborted
- **Error Actions**: When a Java Action throws an exception.
- **Compensation Actions**: When an error occurs in a Java Action that accesses an external system.

As a default, each Action Set runs as a single transaction. If any action in the set fails, the database transaction for the entire set is rolled back. However, if Java Actions access resources that are not guarded by the transaction, such as an email server, then those effects cannot be rolled back.

To change transaction settings:

1. Select the node to display the Properties view for the node. The **General** tab contains the **Commit Transaction after completion** check box
2. If the Action Sets of the node are not to run as a single transaction, clear the **Commit Transaction after completion** check box.

Note

If you wish to make Java Actions run in separate transactions, simply move them to different sets in different nodes. Ensure that the **Commit Transaction after completion** check box is selected for that node.

11.1.7 Dealing With Errors in Java Actions

When an error occurs during the execution of a Java Action, an exception is thrown. Systemwalker Runbook Automation allows you to define your own error handling for erroneous Java Actions. In this way, you can prevent that a process instance goes into error state when an exception is thrown.

In addition, you can define actions for Java Actions which perform a "cleanup" before a transaction is rolled back and a process instance is set to the error state. This includes a rollback of all Java Actions in a Java Action Set, and allows you to perform general actions (e.g. sending notification emails in any error case), or executing some specific actions before setting the process instance to error state.

Systemwalker Runbook Automation provides the following options to handle errors in Java Actions:

- **Compensation Actions**: Compensation Actions are used to clean up the system and to ensure a consistent state of all systems involved in a transaction, e.g. external databases or mail servers. Compensation Actions are particularly useful whenever external systems are involved.

  If you do not define any error handling for a Java Action, the following happens: When an exception is thrown in this Java Action, the transaction will be rolled back. A rollback, however, is only possible for changes in the Application Server context. Any transactions in external systems cannot be rolled back, for example, if a row has been added to an external database. Therefore, it is sometimes necessary to manually clean up external systems to ensure a consistent state of all systems used in the transaction. Optionally, you can use Compensation Action Sets.

You can specify a Compensation Action for every Java Action in an Action Set. You can use a Compensation Action e.g. for removing a newly added row in a database or for sending out an additional email. If an exception occurs in a regular Java Action Set, all Compensation Actions defined for all Java Actions that have been successfully executed before the exception was thrown, are invoked in reverse order.

Note

You cannot embed a Compensation Action in another Compensation Action. If a Compensation Action throws an exception, the process instance will immediately go to error state, and the execution of remaining Compensation Actions will be aborted.

- **Error Actions**: On Java Action level, Error Action sets will become active when an exception is thrown in the related regular Java Action. Error Actions are bound together in Action Sets, similar to all other Action Sets. Error Actions can be specified for any action inside an Action Set. Note that you cannot define an Error Action that handles exceptions occurring in an Error Action.

  For every Error Action, you can specify additional error handling settings on an **Error Handling** tab. Note that this tab is only available when defining Error Actions for a Java Action. On process definition level and when defining Compensation Actions, defining error handling settings is not required.

  - **Behavior after Error**: Specifies the behavior of the process instance after executing the Error Action. You can specify that a process instance goes to error state or continues executing in case of an error.

  - **Exceptions to React to**: You can choose which kind of exceptions trigger the execution of the Error Action.

Note

In case an Error Action throws an exception, the transaction will be rolled back instantly and the process instance will go into error state. No Error Action can be defined for such a case.

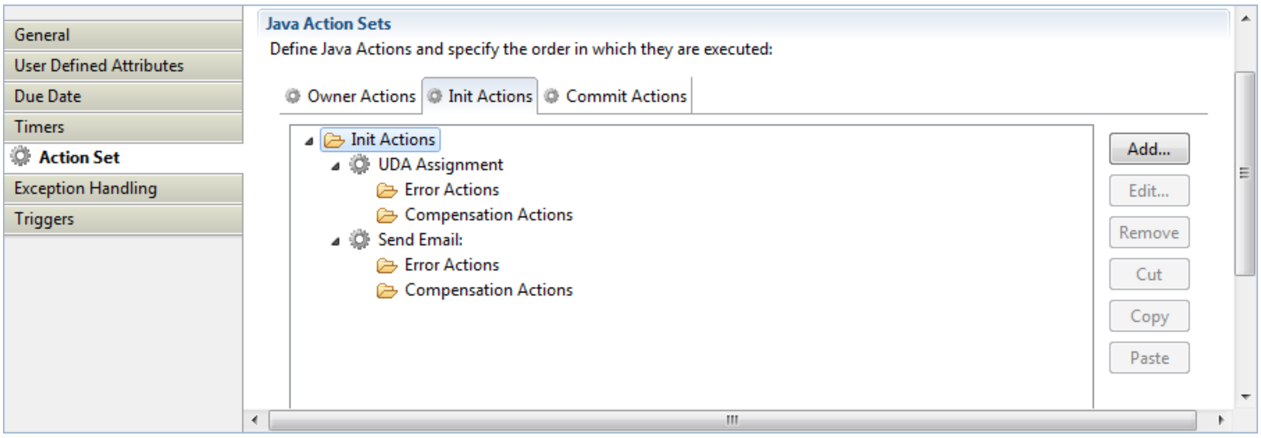
To define Error and Compensation Actions for regular Java Actions:

1. Click the empty space in the Process Definition editor or select the node to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab.

   The **Properties** view displays all Java Actions that are available for the selected process definition or node. If you have not yet defined any Error or Compensation Java Actions for a regular Java Action, empty folders for Error and Compensation Java Actions are shown. The following figure shows a Prologue Actions folder for an Activity Node, consisting of two Java Actions:

   Figure 11.5 Properties view with Prologue Actions



3. Select the regular Java Action for which you want to define an Error and/or Compensation Action.

4. Select the **Error Actions** or **Compensation Actions** folder or any Error or Compensation Action inside the folder, and click **Add**.

   The **Action Type List** dialog is opened where you can select a Java Action and add this action as a new Error or Compensation Action.

5. In the **Action Type List** dialog, expand the folder where the Java Action to be added is located. Select the Java Action, and click **Create**.

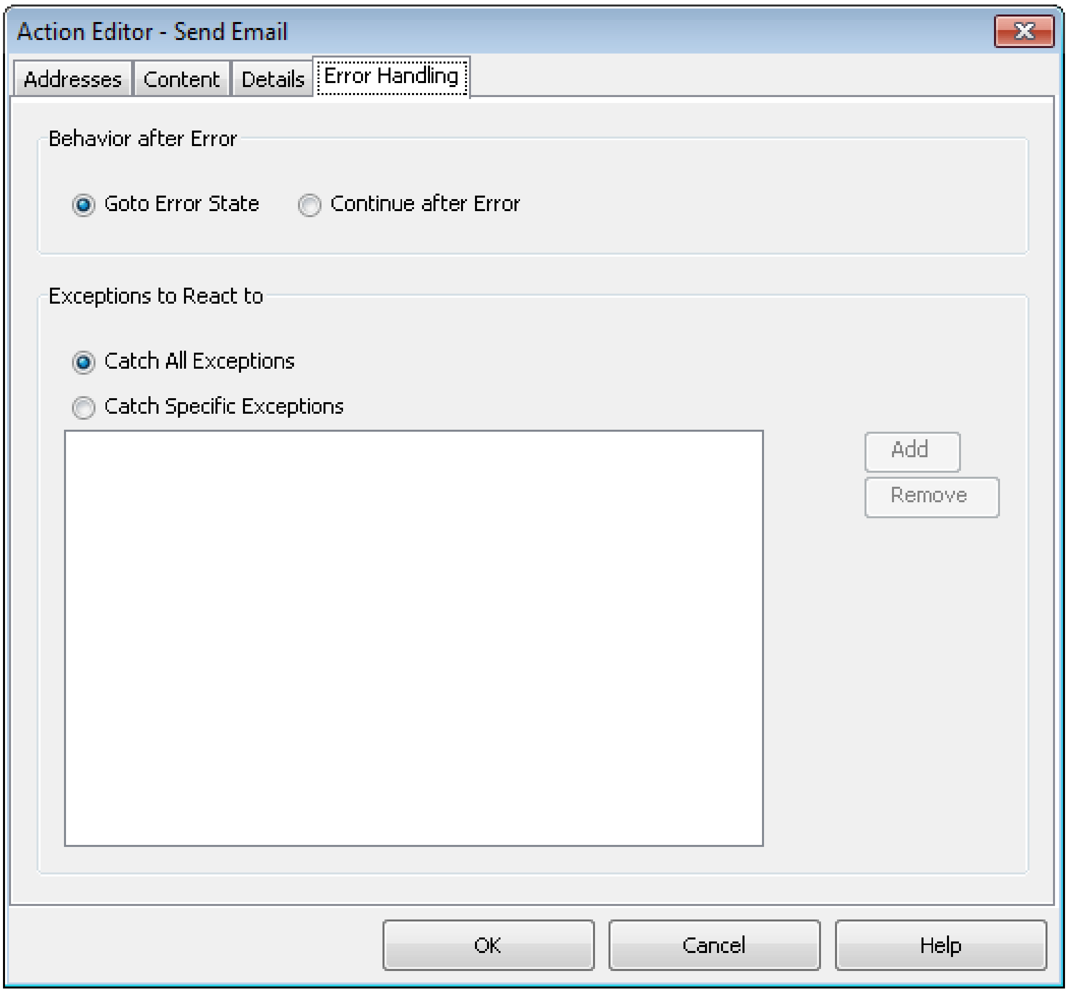
   A dialog is displayed where you fill in the details for the selected Java Action. Perform operations required depending the action you selected.

6. When defining an Error Action for a Java Action: In the Action Editor for the selected Java Action, click the **Error Handling** tab. Here you can specify the following error handling settings:

   Note

   This tab is only available when defining Error Actions on process definition level or a regular Java Action. On process definition level and when defining Compensation Actions, this tab is not available.

   Figure 11.6 Displaying the Error Handling Tab

   Action Editor - Send Email

   Behavior after Error: ● Goto Error State ○ Continue after Error

   Exceptions to React to: ● Catch All Exceptions ○ Catch Specific Exceptions

   - **Behavior after Error**: Specifies the behavior of the process instance after executing the Error Action. Select the **Goto Error State** radio button if you want your process instance to go to error state. In case of an error, the Compensation Actions specified for the Java Actions and the Error Actions on process level are executed, the process instance is rolled back, and the instance is put to error state.

     Select the **Continue after Error** radio button to continue executing your process instance. In case of an error, the exception gets caught, the Error Actions specified for the failed Java Action are executed, and the process instance continues. The default setting is **Goto Error State**.

Note

If you have defined several Error Actions with different settings, the **Goto Error State** setting overrides the **Continue after Error** setting.

- **Exceptions to React to**: You can choose which kind of exceptions trigger the execution of the Error Action. If you want the Error Action to react on any kind of exception, select the **Catch All Exceptions** radio button. In that case, the **Add** and **Remove** buttons are disabled.

  Select the **Catch Specific Exceptions** radio button if you want the Error Action to react on specific exceptions. In that case, you have to specify the exception class names (for example, java.lang.NullPointerException), using the **Add** and **Remove** buttons.

  Note that if you specify java.lang.Exception as exception class, the behavior will be the same as when specifying **Catch All Exceptions**, because all exceptions belong to this class unless you are more specific.

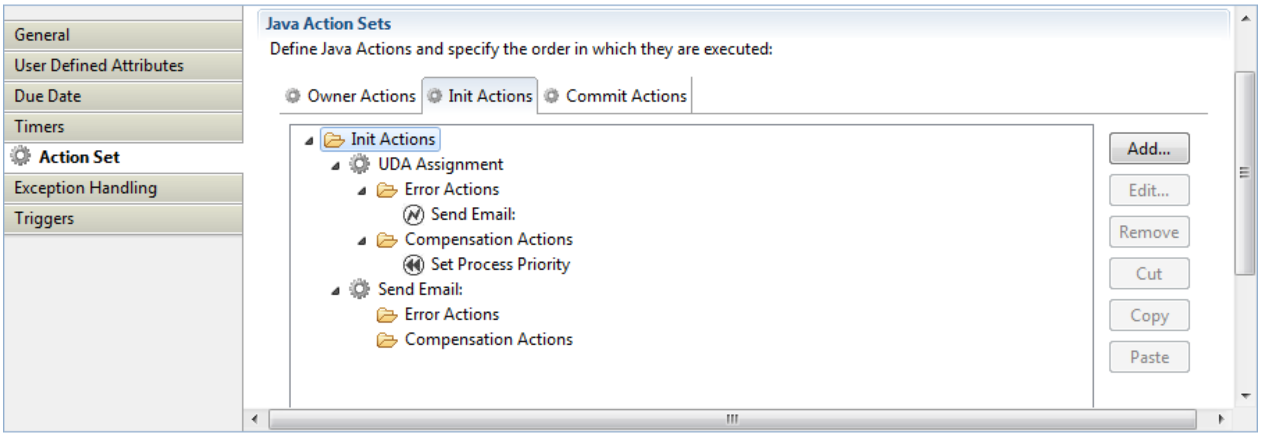
You can combine all error handling settings. Each combination results in a different error handling procedure. The following table shows all possible combinations and their impact on executing the defined Java Actions:

Error Handling Settings	Goto Error State	Continue after Error
Catch All Exceptions	If a Java Action throws any kind of exception, the specified Error Action is executed. The transaction is rolled back, and the process instance goes to error state.	If a Java Action throws any kind of exception, the specified Error Action catches this exception, and the process instance continues.
Catch Specific Exceptions	If a Java Action throws the specified exception, the Error Action is executed. The transaction is rolled back, and the process instance goes to error state. If a Java Action throws an exception that is not specified, the Error Action is not executed, the transaction is rolled back, and the process instance goes to error state (default error handling behavior).	If a Java Action throws the specified exception, the Error Action catches this exception, and the process instance continues. If a Java Action throws an exception that is not specified, the Error Action is not executed, the transaction is rolled back, and the process instance goes to error state (default error handling behavior).

7. Fill in details for the selected Java Action, specify the Error Action behavior, and click **OK**.

   The new Java Action is displayed in the Properties view. In the example, one Error Action and one Compensation Action have been added to the Prologue Action.

Figure 11.7 Adding Error and Compensation Actions



11.2 Using Server Actions

Server Actions enable you to interact with the Systemwalker Runbook Automation Server. Using Server Actions, you can

- Assign an activity to a user or escalate an activity
- Make an automatic choice
- Retrieve the process initiator or performer of an activity
- Set standard attributes like process instance name, description, or priority
- Set the value of User Defined Attributes (UDAs)
- Evaluate a JavaScript
- Assign a task to the performer of a completed activity
- Set an assignee from relationship

11.2.1 Assigning an Activity to a User

When modeling an activity, you assign it to a Role according to who is responsible for completing the activity. In some situations, however, you may want to assign an activity to a particular user, and not just to any user who is a member of a certain Role. This can be achieved using the Assign Task to User Java Action. You can use this Java Action as a Role Action with Activity Nodes, Voting Activity Nodes, and Compound Activity Nodes.

Note

You can assign the activity only to users within the Role to which the activity is currently assigned. If you want to reassign the activity to users from another Role, you must change the activity's Role assignment.

To assign an activity to a user:

1. Select the Activity Node, Voting Activity Node or Compound Activity Node to display the **Properties** view for the nodes.
2. Select the **Action Set** tab. Click **Add** on the **Role Actions** tab. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **Assign Task To User**.
4. Specify a JavaScript expression for the users to be assigned.

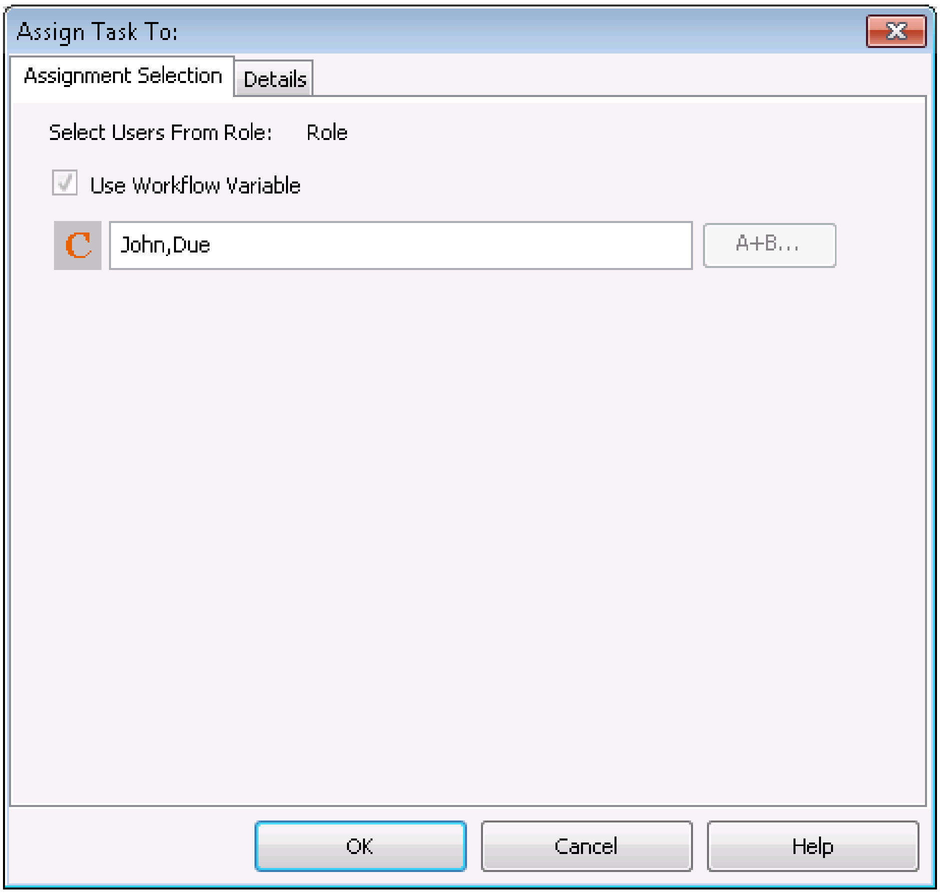
   You can type a constant (that is, the name of a user), select a User Defined Attribute (UDA) that has the name as its value, or build a complex JavaScript expression that evaluates to a name. For details, refer to section 11.14 Defining JavaScript Expressions.

   Note

   If you select a UDA, the UDA must be of type STRING. Otherwise an error will occur when the Java Action is executed.

If you want to specify multiple users, use commas to separate the names. The following figure shows an example.

Figure 11.8 Assigning an Activity to Users



John,Due

Note

The JavaScript expression whether it is a simple UDA or a complex JavaScript expression must be resolved into one or multiple users within the Role to which the activity is currently assigned.

5. On the **Details** tab, type a descriptive name and your notes for the Java Action.

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

Note

By defining the Role Java Action, the activity is assigned to a particular user. In this case, defining the Role of the user to whom you want to assign the activity is not required. You can assign a Voting Activity and a Compound Activity in the same manner.

11.2.2 Assigning Task to Performer of Completed Activity

You can assign tasks to users in the **General** tab of **Properties** view of the Process Definition or by using Java Actions. If you want to assign tasks to users after the Process Definition has been designed, you can do it by setting a Java Action.

Depending on the business requirements, an assignee of a completed activity might need to also complete another activity. For example, User 1 has completed Activity 1. The same user also needs to complete Activity 2. In this scenario, you can assign the Activity 2 to the User 1.

Note

You can assign a user of a completed activity to another activity only by using Role Java Actions.

Note

You can assign an activity only to the users within the Role to which the activity is currently assigned. If you want to reassign the activity to users from another Role, you must add this user to the current Role.

To assign a task to the performer of a completed activity:

1. Select an activity in the Process Definition which needs to be assigned to a user.

   Note

   You can either select an Activity Node or a Voting Activity Node or a Compound Activity Node to set the Java Action on it in order to assign it to the performer of a completed activity.

2. Click **Action Set** tab of the **Properties** view.

   Java Action Sets area is displayed in **Action Set** tab. By default, **Role Actions** tab is selected in **Java Action Sets**.

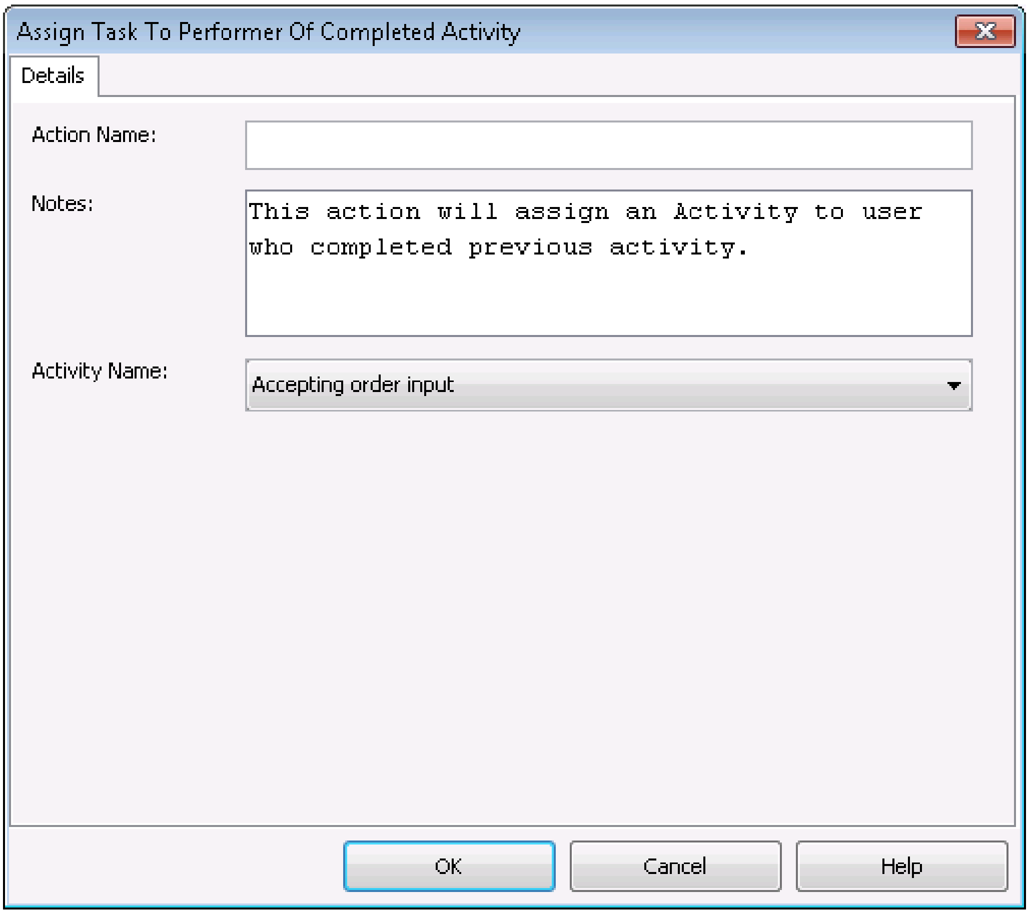
3. Select **Role Actions** in the work area and click **Add**.

   Action Type List dialog is displayed.

4. Double-click on the **Server Actions** action type.

5. Select **Assign Task To Performer Of Completed Activity** in the **Action Type List** dialog and click **Create**.

Figure 11.9 Assigning task to performer of completed activity



Assign Task To Performer Of Completed Activity

Details

Action Name:

Notes: This action will assign an Activity to user who completed previous activity.

Activity Name: Accepting order input

6. Optional: Enter a name for the action in **Action Name** field and edit notes about the action in **Notes** field. By default, a note will be displayed in the **Notes** field.

7. From the **Activity Name** drop-down list, select the activity that has been completed by a user to whom you intend to assign the selected activity.

   Note

   Only Activity Nodes are displayed in the **Activity Name** drop-down list.

8. Click **OK**.

   The activity for which the Java Action has been defined is assigned to the user who has completed the activity selected in the **Activity Name** drop-down list.

Note

By defining the Role Java Action, the activity is assigned to a particular user. In this case, defining the Role of the user to whom you want to assign the activity is not required. You can assign a Voting Activity and a Compound Activity in the same manner.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.3 Setting Assignee from Relationship

You can assign tasks to users in the **General** tab of **Properties** view of the Process Definition or by setting Java Actions. If you want to assign tasks to users after the Process Definition has been designed, you can do it by setting a Java Action.

Depending on the business requirements, an activity might need to be assigned to a user who is related to the performer of a completed activity. This relationship could be manager, colleague, team leader etc. For example, Activity 1 has been completed by User 1. Activity 2 needs to be completed by the user (User 2) who is the manager of User 1. In this scenario, you can specify the relationship of User 2 with User 1 and assign Activity 2 to User 2.

Similarly, you can assign an activity to a user depending on the user names associated with the UDAs and relationships between the users. For example, Variable 1 is a String type UDA, which is associated with User 1. In this scenario, Activity 2 needs to be assigned to User 2 who is the manager of User 1. You can assign Activity 2 to User 2 by specifying this relationship in the Java Action.

Note

Only UDAs of type STRING are considered for setting assignees to activities from relationships using UDAs.

To set assignee from relationship:

1. Select an activity in the Process Definition which needs to be assigned to a user.

   Note

   You can select an Activity Node or a Voting Activity Node or a Compound Activity Node to set the Java Action on it in order to assign it to the performer of a completed activity.

2. Click **Action Set** tab of the **Properties** view.

   The **Java Action Sets** area is displayed in **Action Set** tab. By default, **Role Actions** tab is selected in **Java Action Sets**.

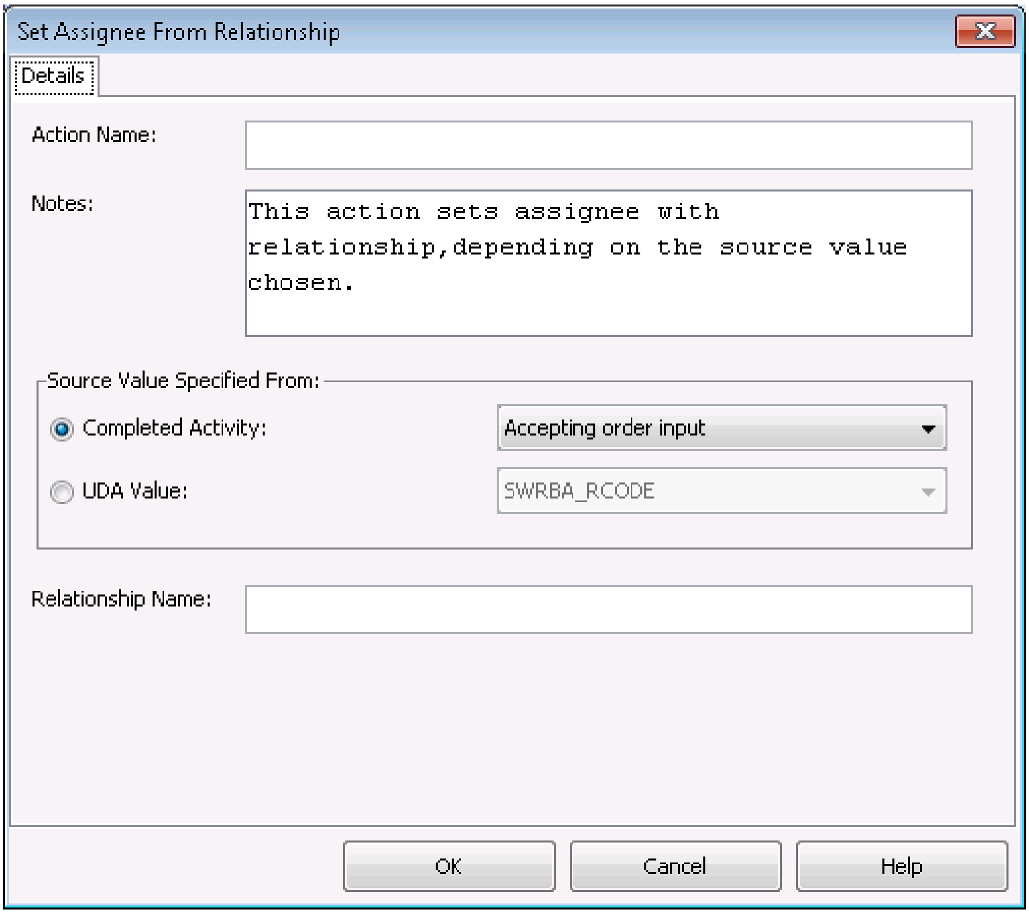
3. Select **Role Actions** in the work area and click **Add**.

   Action Type List dialog is displayed.

4. Double-click on the **Server Actions** action type.

5. Select **Set Assignee From Relationship** and click **Create**.

Figure 11.10 Setting assignee from relationship



This action sets assignee with relationship,depending on the source value chosen.

● Completed Activity: Accepting order input
○ UDA Value: SWRBA_RCODE

The following sections describe the procedures to set assignee to an activity depending on his/her relationship with the user of a completed activity and to set assignee to a task depending on his/her relationship with the user names associated with the UDAs.

Note

Relationship Name is a mandatory field.

- **To assign an activity to user who has a relationship with the performer of a completed activity**

  1. Optional: Enter the name of the action in **Action Name** field.
  2. Optional: Edit notes in **Notes** field. By default, a note will be displayed in the **Notes** field.
  3. Select the **Completed Activity** radio button (selected by default) in **Source Value Specified From**. Upon selecting this radio button, all the completed activities are displayed in the drop-down list except the one which needs to be assigned to a user. The **UDA Value** radio button and the drop-down list are disabled.
  4. Select the completed activity from the drop-down list.
  5. Enter the relationship name in the **Relationship Name** field. For example, User 1 has completed Activity 1. User 2 is the manager of User 1. If you want to assign User 2 to Activity 2, select Activity 1 in the **Completed Activity** drop-down list and specify **Manager** in the **Relationship Name** field.
  6. Click **OK**. In case of the above example, Activity 2 will be assigned to the manager of User 1.

- **To assign an activity to user who has a relationship with the user associated with a UDA**

  1. Optional: Enter the name of the action and edit notes in **Action Name** and **Notes** fields respectively. By default, a note will be displayed in the **Notes** field.

2. Select the **UDA Value** radio button in **Source Value Specified From**. Upon selecting this radio button, all the STRING type UDAs of the Process Definition are displayed in the drop-down list. Each UDA is associated with a user name

3. Select the UDA from the **UDA Value** drop-down list. For example, if you want to assign User 2 to Activity 2, select UDA Variable 1 in **UDA Value**. User 1 is associated with Variable 1 and User 2 is the manager of User 1.

4. Enter the relationship name in the **Relationship Name** field. Specify **Manager** in the **Relationship Name** field.

5. Click **OK**. In case of the above example, Activity 2 will be assigned to the manager of User 1 that is User 2.

Note

You can assign an activity only to the users within the Role to which the activity is currently assigned. If you want to reassign the activity to users from another Role, you must add this user to the current Role.

Note

By defining the Role Java Action, the activity is assigned to a particular user. In this case, defining the Role of the user to whom you want to assign the activity is not required. You can assign a Voting Activity and a Compound Activity in the same manner.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

Relationship names

Relationship names are explained below.

Activity actors

After an activity has been allocated to a role, the people in that role group can then operate on the activity. If, for a particular instance, an activity has been allocated to a manager role, all members of the manager group can operate on that activity. However, in fact, only one manager will execute the activity. That manager is referred to as the "actor" for that activity.

Depending on the workflow, after an activity has been executed, there may be cases where you want to decide who will work with the activity. You may then want to take action based on that information.

For example, consider an activity called "record customer incident" allocated to the role "senior customer support staff". First it is necessary to check which of the "senior customer support staff" actually performed the "record customer incident" activity. Then, it would be desirable to allocate the same actor to the next activity called "escalate incident" within the process instance.

Determining the activity actors

The ServerEnactmentContext interface of the com.fujitsu.iflow.server.intf package includes a getActivityActor() method used to determine the actor for the activity.

The actor for a completed Activity Node can be acquired by passing an Activity Node name as a parameter for the getActivityActor() method.

- Voting Activity Nodes do not support this method.

- It is recommended to assign a unique name to each Activity Node within a process instance. This method will throw an exception if there is a node with the same name as a node that the activity has already passed through.

- The next sample code gets the actor for a past activity ("Activity A") and then allocates the same actor to the current activity.

```
public void assignActivityActor(ServerEnactmentContext sec){
  String[] actor = new String[1];
  actor[0] = sec.getActivityActor("Activity A");
  sec.setActivityAssignees(actor);
}
```

- This method must be called by a JavaAction defined as part of the JavaScript. Refer to the API Javadoc Manual for details.

Relationships

In certain workflows, work is sometimes allocated by referring to UDA values, or the hierarchical relationship to the activity actor. For example, if the value "Fujitsu" has been set for the UDA "company name", it is possible to allocate only the executives to particular activities by checking that the actor is the executive for the Fujitsu key. Also, if a certain activity has been allocated to an actor called "Jim", it is possible to find out who Jim's manager is and then allocate a particular activity to that manager.

Determining relationships

Determine relationships using the resolveRelationship() method of the ServerEnactmentContext interface included in the com.fujitsu.iflow.server.intf package.

The resolveRelationship() method returns the target value by using a source (reference) value and a relationship as parameters. This method is functioned by defining the source value, the relationship and the target value beforehand. This enables the method to return the appropriate values.

Source value	Relationship	Target value
Jim	manager	Robert
Jim	assistant	Arthur
Fujitsu	executive	Bob

For example, resolveRelationship("assistant", "Jim") returns "Arthur".

The mappings between source values, relationships, and target values must be stored as user profiles in either the directory service or the local user store. Here, the source value is the user ID, the relationship is a user attribute name, and the target value is a user attribute value.

- To create a user profile, refer to the API Javadoc Manual for information on the DirectoryServices interface that is included in the com.fujitsu.iflow.model.workflow package.
- For source values, both humans and non-humans (such as company or group names) can be set. If a user profile for a non-human object is added to the directory server or the local user store, ensure that the user profile is not used for login purposes.
- The next sample code gets the actor for a past activity ("Activity A") and the Manager for that actor, and then allocates that Manager to the current activity.

```
public void assignManagerOfActivityActor(ServerEnactmentContext sec){
  String[] managers =
  sec.resolveRelationship("Manager",sec.getActivityActor("Activity A"));
  sec.setActivityAssignees(managers);
}
```

- This method must be called by a JavaAction that has been defined as part of the JavaScript. Refer to the API Javadoc Manual for details.

11.2.4 Setting Priority for an Activity

You can set activity level priority for Activity Nodes, Voting Activity Nodes and Compound Activity Nodes to an integer greater than or equal to 0. You can use the Set Activity Priority Java Action to set the priority for the activity.

Priority at the activity level can be set either by using a Java Action or through the **General** tab in the **Properties** view. For more information on setting activity level priority through the **General** tab, refer to section 6.7.3 Setting Activity Level Priority.

To set the activity priority using Java Action:

1. Select the Activity Node or Voting Activity Node or a Compound Activity Node to display the Properties view for the nodes.
2. Select the **Action Set** tab.
3. Select the **Prologue Actions** tab from the Java Action types and click **Add**. The **Action Type List** dialog is displayed.

4. Expand **Server Actions** and double click **Set Activity Priority**. The **Action Editor - Set Activity Priority** is displayed.

5. Type a descriptive name and your notes for the Java Action.

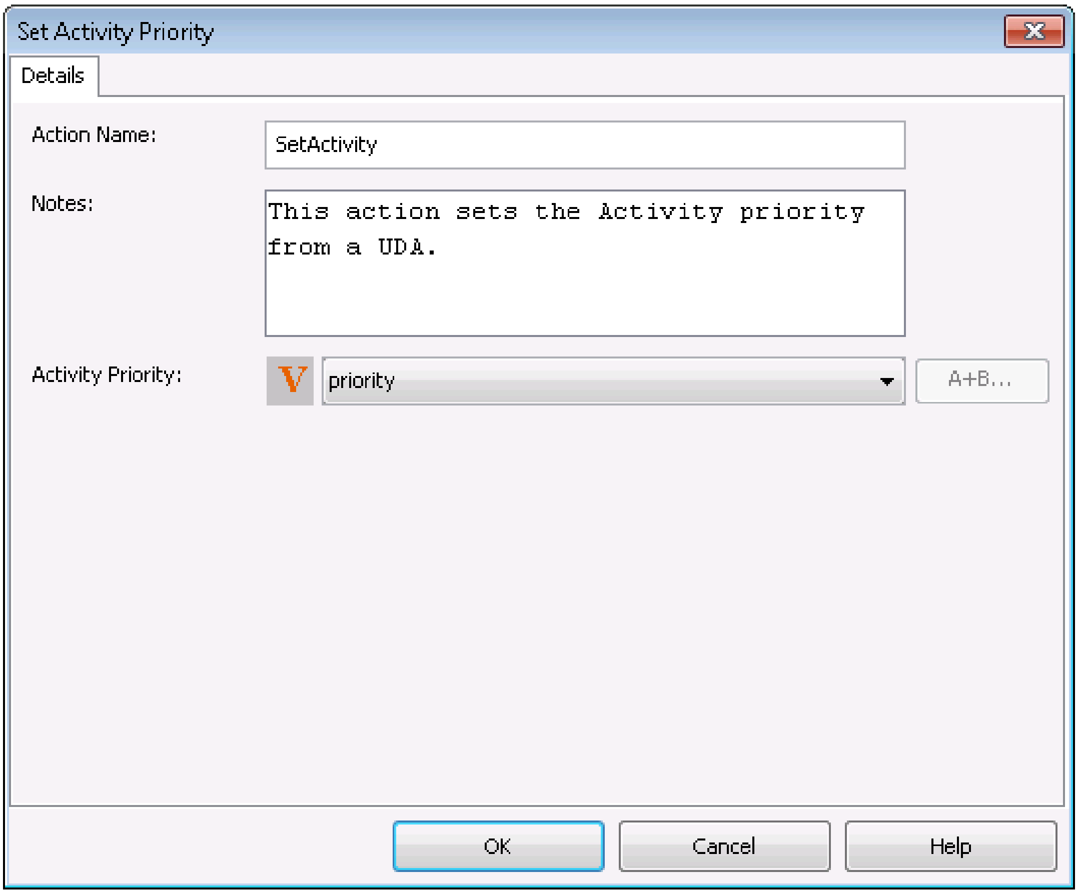
6. Enter a JavaScript expression in the **Activity Priority** field that evaluates to an Integer value. There are different ways to set the priority. You can type a constant (that is, a number), select a User Defined Attribute (UDA) that has the priority as its value, or build a complex JavaScript expression. For details, refer to *Defining JavaScript Expressions* .

   Note

   If you select a UDA, the UDA must be of type INTEGER or LONG. Otherwise an error will occur when the Java Action is executed.

   In the following example, the activity priority is set to the value of the UDA Priority.

   Figure 11.11 Setting Activity Priority



Set Activity Priority

Details

Action Name: SetActivity

Notes: This action sets the Activity priority from a UDA.

Activity Priority: priority

7. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.5 Escalating an Activity

When defining a due date for an activity, you also define what happens when the due date is reached and the activity has not been completed. One option is to escalate the activity to additional users using the Escalate Task Java Action.

The same applies when you define a timer for an activity: You can escalate the activity when the timer expires.

To escalate an activity:

1. Define a due date or timer for an Activity Node or Voting Activity Node or Compound Activity Node.

   Refer to section 6.22.1 Defining Due Dates or 6.22.2 Defining Timers for instructions.

2. On the **Due Date** or **Timers** tab, click **Add**. The **Action Type List** dialog is displayed.

3. Expand **Server Actions** and double click **Escalate Task**.

4. Specify a JavaScript expression for the users to whom you want to escalate.

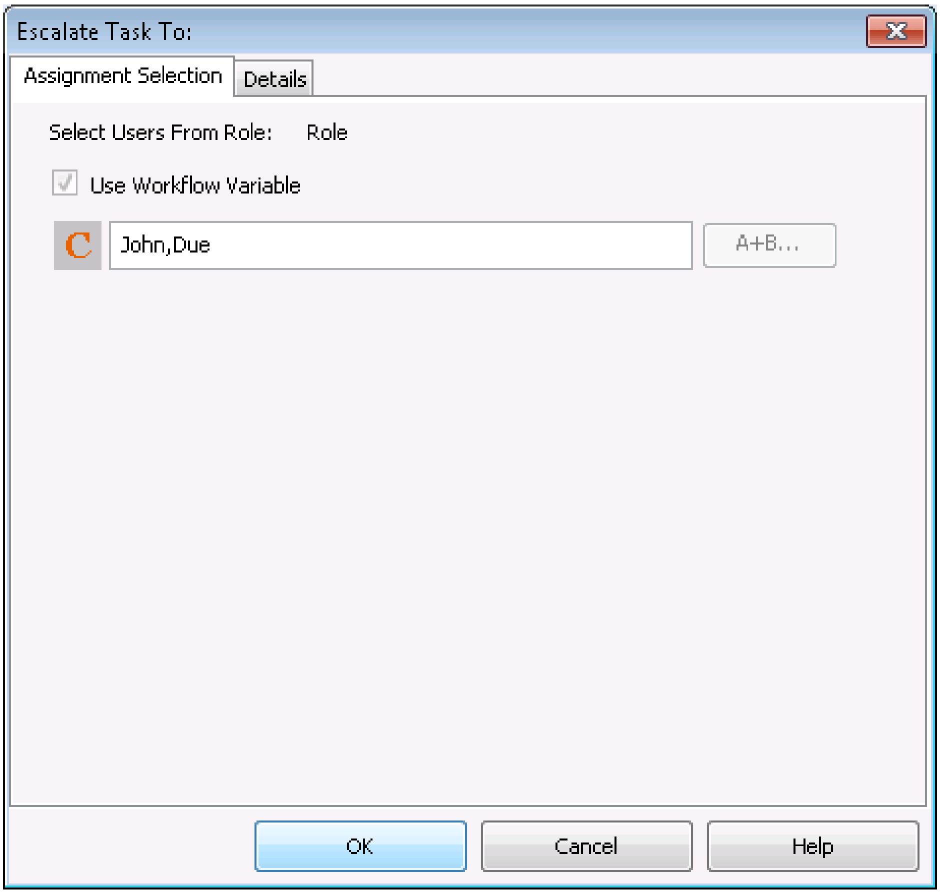
   You can type a constant (that is, the name of a user), select a User Defined Attribute (UDA) that has the name as its value, or build a complex JavaScript expression that evaluates to a name. For details, refer to section 11.14 Defining JavaScript Expressions.

   Note

   If you select a UDA, the UDA must be of type STRING. Otherwise an error will occur when the Java Action is executed.

   If you want to specify multiple users, use commas to separate the names. The following figure shows an example.

   Figure 11.12 Escalating an Activity



Escalate Task To:

Select Users From Role: Role

☑ Use Workflow Variable

John,Due

5. On the **Details** tab, type a descriptive name and your notes for the Java Action.

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.6 Evaluating a JavaScript

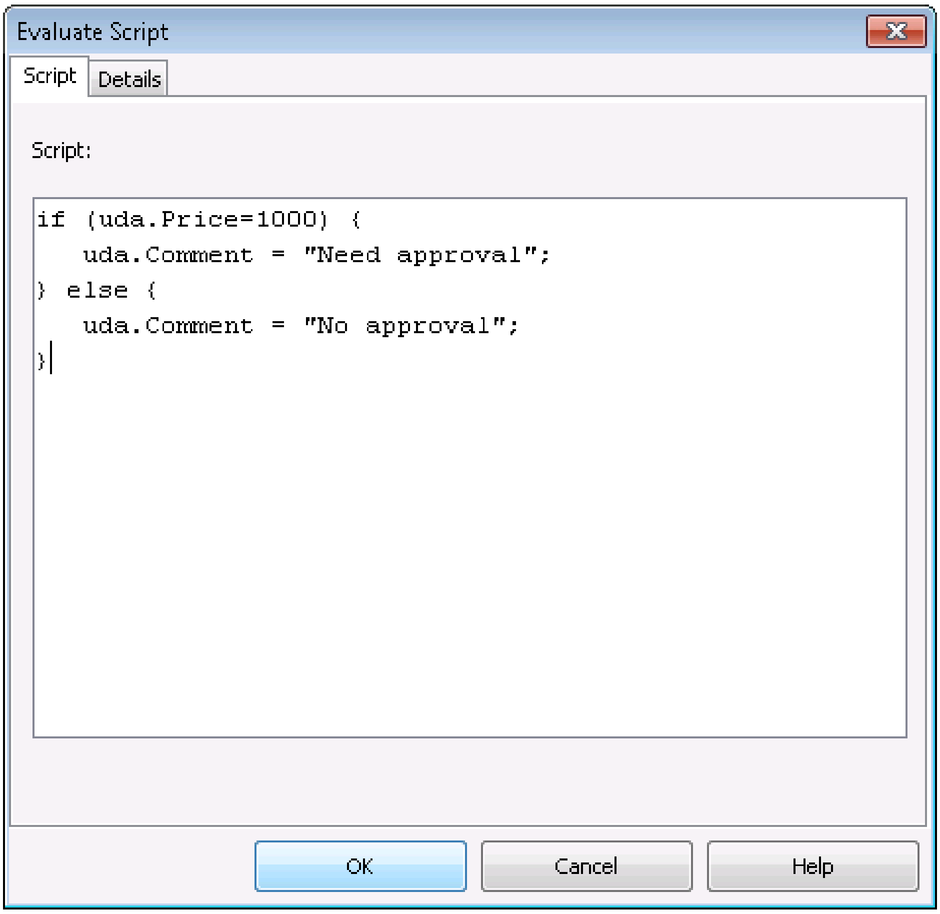
You can use JavaScript functionality within a Java Action. The Evaluate Script Java Action allows you to combine other Java Actions with JavaScript in an Action Set and control the order of JavaScript execution in relation to the other JavaScripts. Also, multiple JavaScripts can be evaluated in any order using a series of these actions in Action Sets.

This section only provides instructions on assigning the Java Action. It provides no information on writing the JavaScript for it. For more information on JavaScript, refer to appendix Appendix A Supported JavaScript Functions.

To evaluate a JavaScript:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **Evaluate Script**.
4. Type your JavaScript in the text area.

Figure 11.13 Evaluating a JavaScript



Script:

```
if (uda.Price=1000) {
   uda.Comment = "Need approval";
} else {
   uda.Comment = "No approval";
}
```

5. On the **Details** tab, type a descriptive name and your notes for the Java Action.
6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.7 Getting the Performer of an Activity

You can assign the name of the user completing an activity to a UDA using the Get Performer Java Action. You can use this Java Action as an Epilogue Action of an Activity Node or Voting Activity Node or Compound Activity Node.

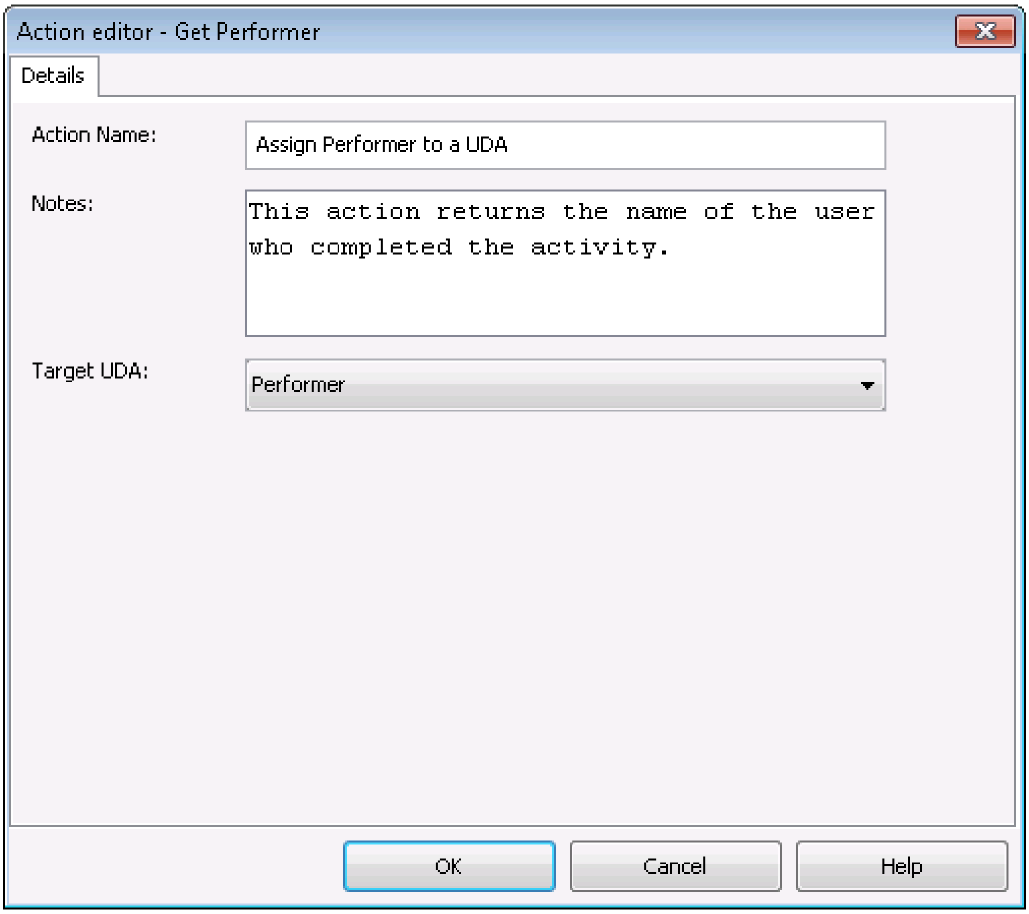
Prerequisite:

- The process definition has a User Defined Attribute (UDA) to which the performer can be assigned.

To assign the performer to a UDA:

1. Select the Activity Node or Voting Activity Node or Compound Activity Node to which you want to assign the Java Action, to display the **Properties** view.
2. Select the **Action Set** tab and then the **Epilogue Actions** tab and click **Add**. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **Get Performer**.
4. In the **Get Performer** dialog, type a descriptive name and your notes for the Java Action.
5. In the **Target UDA** field, select the UDA to which the performer's name is to be assigned.

Figure 11.14 Assigning the Performer's Name to a UDA



Action Name: Assign Performer to a UDA

Notes: This action returns the name of the user who completed the activity.

Target UDA: Performer

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.8 Getting the Process Initiator

Using the Get Process Initiator Java Action, you can assign the name of the user who starts the process instance to a UDA.

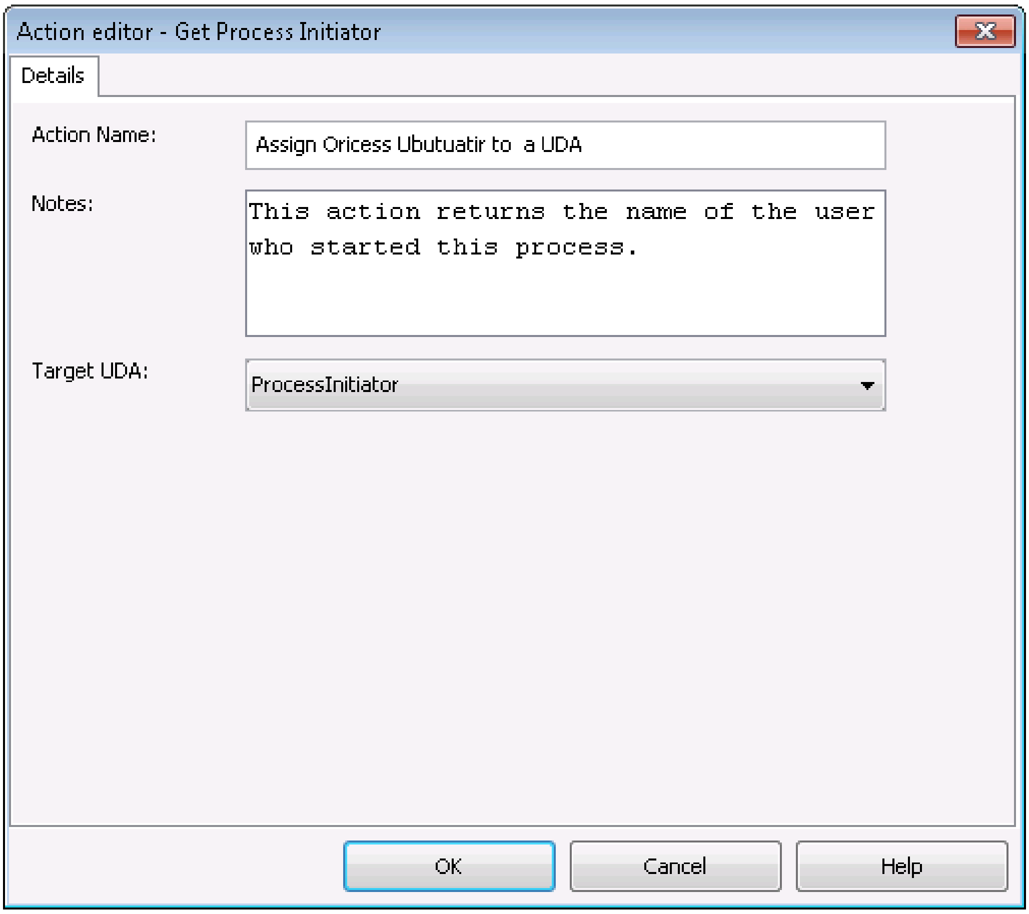
Prerequisite:

- The process definition has a User Defined Attribute (UDA) to which the process applicant can be assigned.

To assign the process initiator to a UDA:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **Get Process Initiator**.
4. In the **Get Process Initiator** dialog, type a descriptive name and your notes for the Java Action.
5. In the **Target UDA** field, select the UDA to which the initiator's name is to be assigned.

Figure 11.15 Assigning the Initiator 's Name to a UDA



Action editor - Get Process Initiator

Details

Action Name: Assign Oricess Ubutuatir to a UDA

Notes: This action returns the name of the user who started this process.

Target UDA: ProcessInitiator

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions .

11.2.9 Making an Automatic Choice

Using the Make Choice Java Action, a choice on a work item can be made automatically. You can use Make Choice as a Timer Action with Activity Nodes, Voting Activity Nodes, and Compound Activity Nodes.

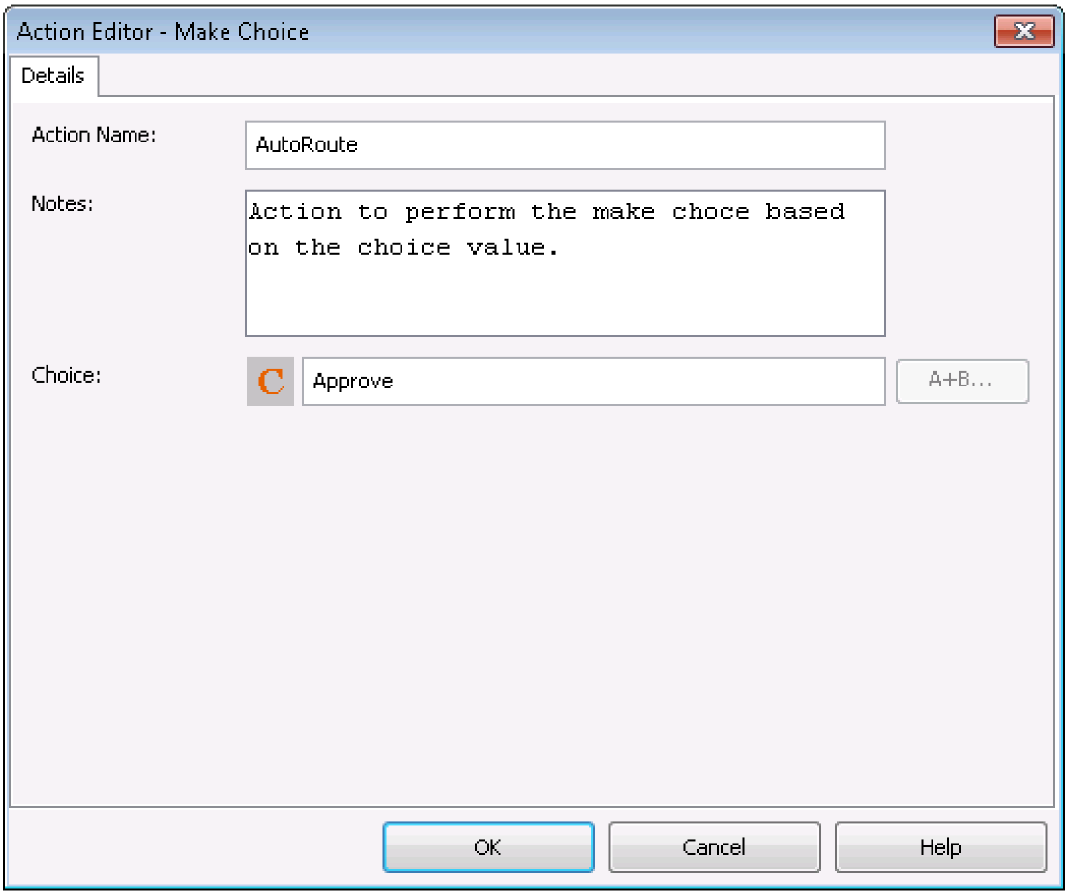
To make an automatic choice:

1. Define a timer for the Activity Node or Voting Activity Node or Compound Activity Node.

   Refer to section 6.22.2 Defining Timers
2. On the **Timers** tab, in the **Specify Java Actions** area click **Add**. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **Make Choice**.
4. Type a descriptive name and your notes for the Java Action.
5. Specify a JavaScript expression for the arrow that is to be chosen in the **Choice** field.

   You can type a constant (the arrow name), select a User Defined Attribute (UDA) that has the arrow name as its value, or build a complex JavaScript expression that evaluates to an arrow name. For details, refer to section 11.14 Defining JavaScript Expressions .

   Figure 11.16 Making an Automatic Choice

   Action Editor - Make Choice

   Details

   Action Name: AutoRoute

   Notes: Action to perform the make choce based on the choice value.

   Choice: Approve

6. Click **OK**.

11.2.10 Setting the Process Instance Name

By default, a process instance has the same name as the process definition from which it is created. However, its name can be changed while the process instance is running using the Set Process Instance Name Java Action.

To set the process instance name:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. Expand **Server Actions** and double click **Set Process Instance Name**.

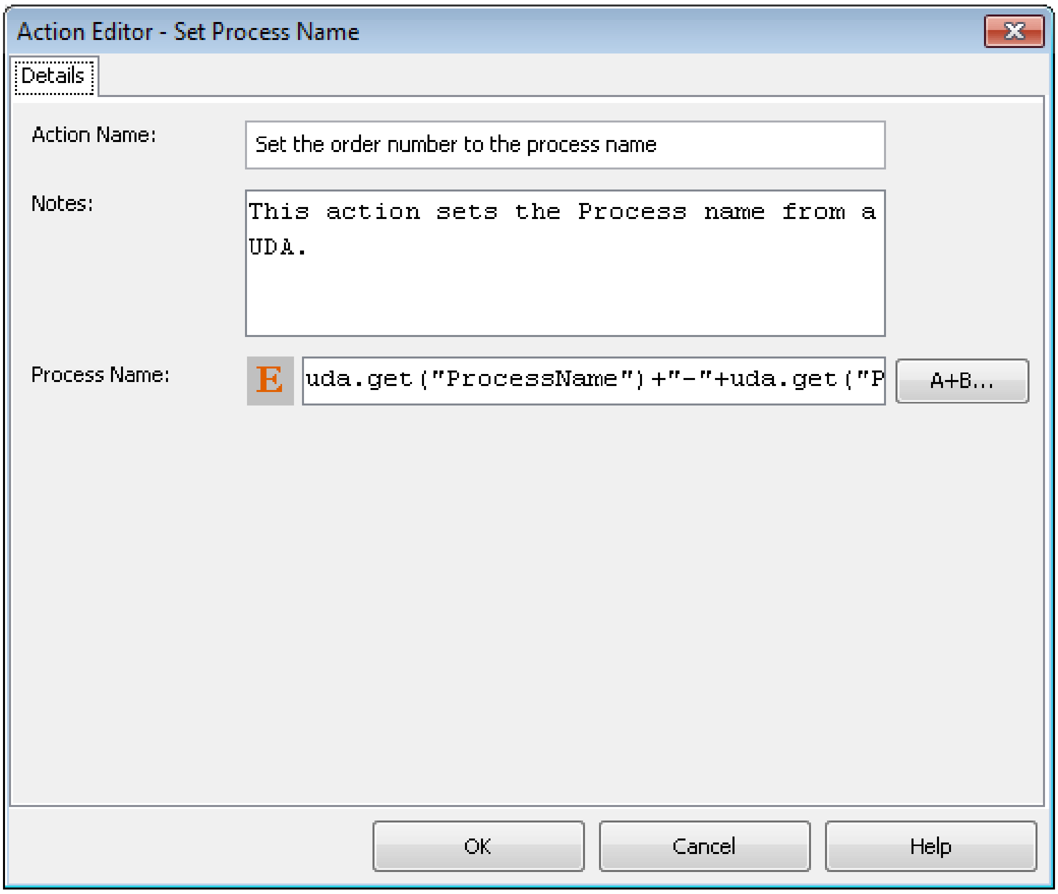
4. Type a descriptive name and your notes for the Java Action.

5. Specify a JavaScript expression for the process instance name in the **Process Name** field.

   You can type a constant (that is, a name), select a User Defined Attribute (UDA) that has the process instance name as its value, or build a complex JavaScript expression that evaluates to the name. For details, refer to section 11.14 Defining JavaScript Expressions.

   The following figure shows an example of how this Java Action can be used. At process initialization, the Java Action appends the purchase order number to the process instance name. The purchase order number is stored in the PONumber UDA.

   Figure 11.17 Setting the Process Instance Name

   Action Editor - Set Process Name

   Action Name: Set the order number to the process name

   Notes: This action sets the Process name from a UDA.

   Process Name: uda.get("ProcessName")+"-"+uda.get("P

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.11 Setting the Process Instance Priority

By default, a process instance is given a medium priority (8). However, its priority can be changed while the process instance is running using the Set Java Action.

To set the process instance priority:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. Expand **Server Actions** and double click **Set Process Instance Priority**.

4. Type a descriptive name and your notes for the Java Action.

5. Enter a JavaScript expression in the **Process Priority** field that evaluates to an Integer value.

   There are different ways to set the priority. You can type a constant (that is, a number), select a User Defined Attribute (UDA) that has the priority as its value, or build a complex JavaScript expression. For details, refer to 11.14 Defining JavaScript Expressions.

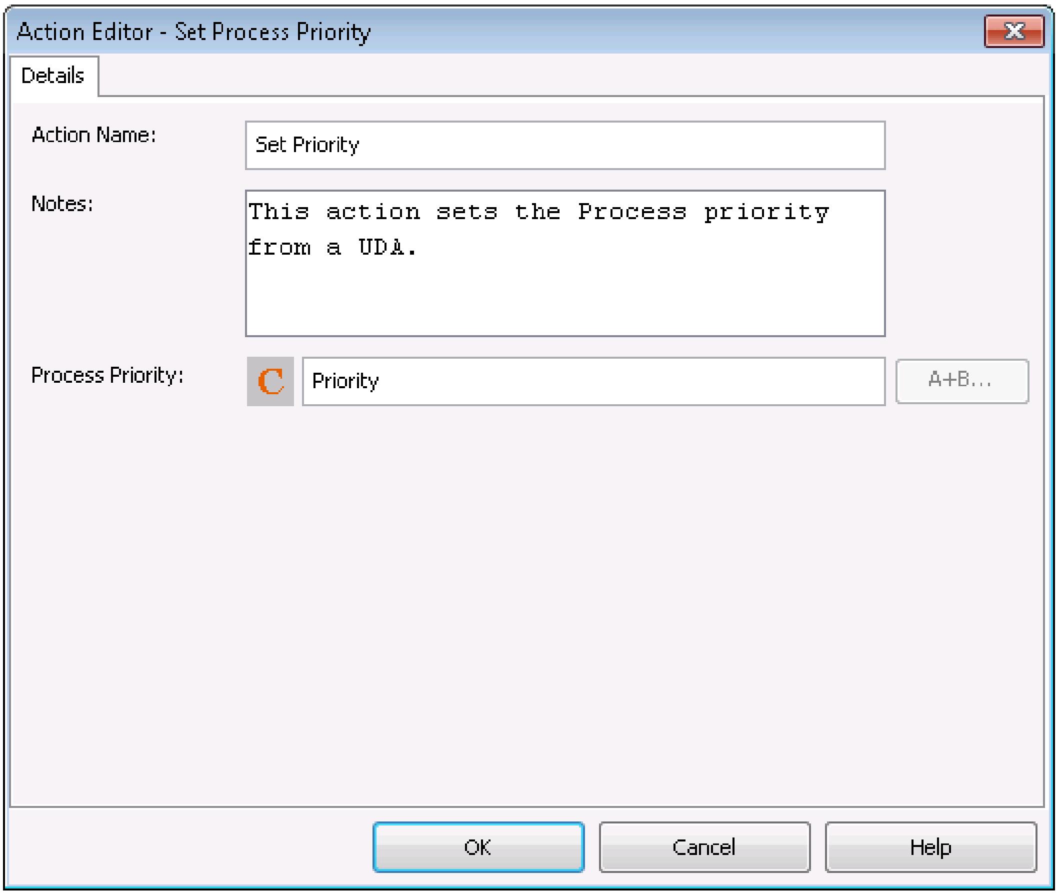
   Note

   If you select a UDA, the UDA must be of type INTEGER or LONG. Otherwise an error will occur when the Java Action is executed.

   Define the evaluation result of a JavaScript expression so as to be a value of type INTEGER.

   In the following example, the process instance priority is set to the value of the UDA Priority.

   Figure 11.18 Setting Process Instance Priority



Action Editor - Set Process Priority

Action Name: Set Priority

Notes: This action sets the Process priority from a UDA.

Process Priority: Priority

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.12 Setting the Process Instance Description

By default, a process instance is given the same description as the process definition from which it is created. However, its description can be changed while the process instance is running using the Set Process Instance Description Java Action.

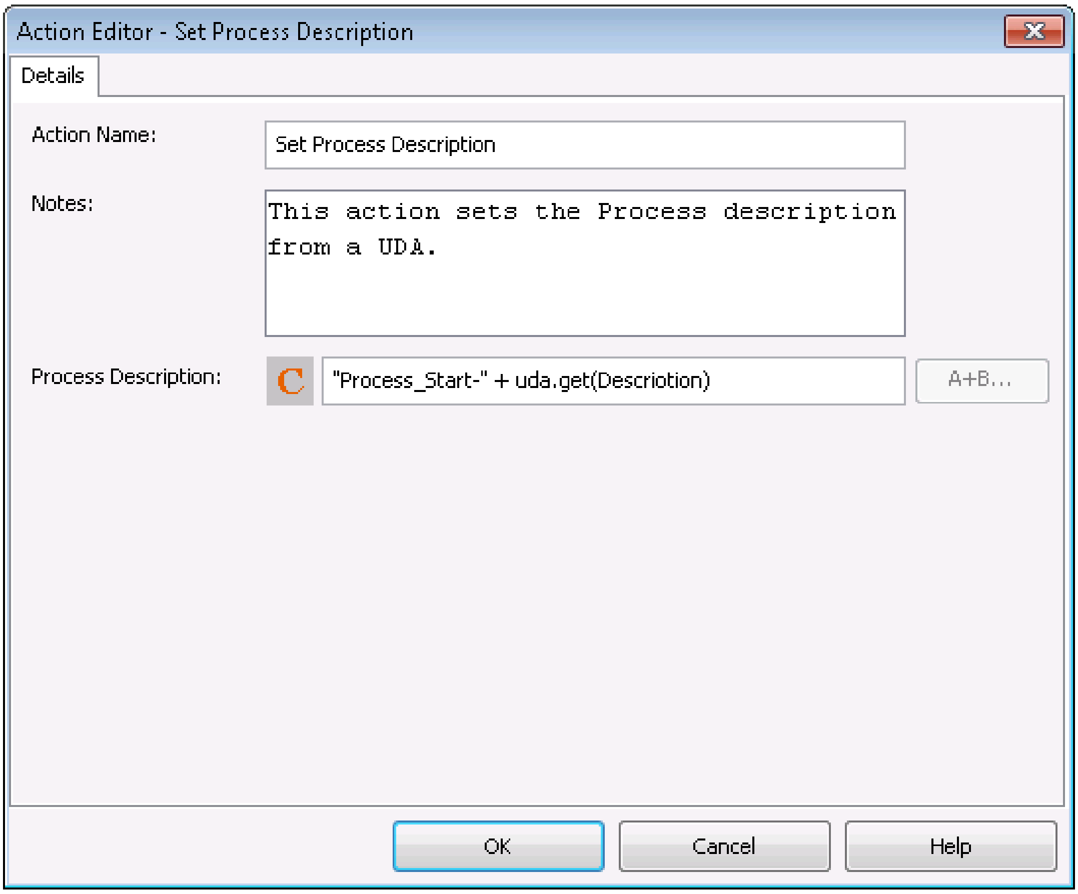
To set the process instance description:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **Set Process Instance Description**.
4. Type a descriptive name and your notes for the Java Action.
5. Enter a JavaScript expression in the **Process Description** field that evaluates to a String value.

   There are different ways to set the description. You can type a constant (that is, a description), select a User Defined Attribute (UDA) that has the description as its value, or build a complex JavaScript expression that evaluates to a description. For details, refer to section 11.14 Defining JavaScript Expressions.

   In the following example, the process instance description is composed of the string Process started and the value of the UDA Description.

   Figure 11.19 Setting the Process Instance Description

   Action Editor - Set Process Description

   Action Name: Set Process Description

   Notes: This action sets the Process description from a UDA.

   Process Description: "Process_Start-" + uda.get(Descriotion)

6. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.2.13 Assigning Values to User Defined Attributes

You can set the value of a UDA using the UDA Assignment Java Action. You can set it, for example, to the value of another UDA.

Prerequisite:

- You have added User Defined Attributes (UDAs) to the process definition.

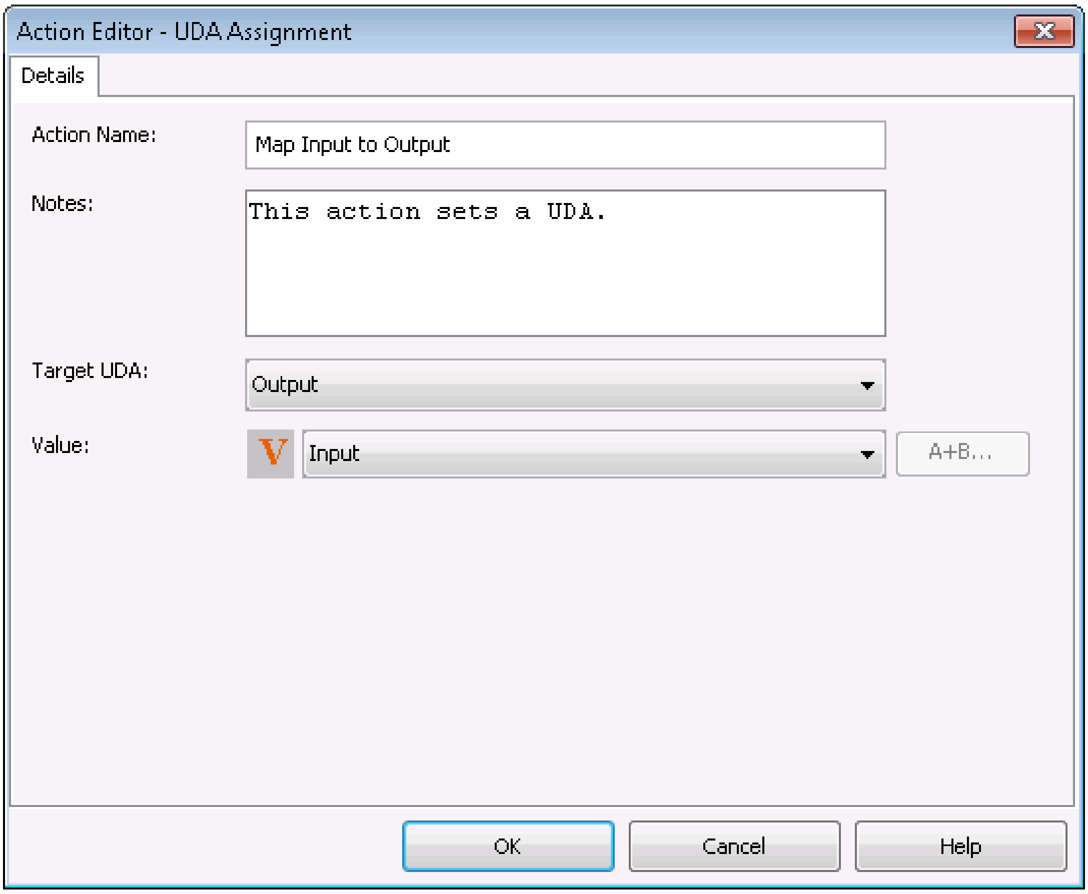
To set the value of a UDA:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **Server Actions** and double click **UDA Assignment**.
4. In the **Action Editor - UDA Assignment** dialog, type a descriptive name and your notes for the Java Action.
5. From the **Target UDA** list, select the UDA whose value you want to set.
6. Specify a JavaScript expression for the value in the **Value** field.

   There are different ways to set the value. You can type a constant (that is, the value itself), set it to the value of another UDA, or build a complex JavaScript expression. For details, refer to section 11.14 Defining JavaScript Expressions.

In the following example, the value of the UDA Input is assigned to the UDA Output.

Figure 11.20 Setting a UDA



Action Name: Map Input to Output

Notes: This action sets a UDA.

Target UDA: Output

Value: V Input

7. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.3 Using XML Actions

You can use XML Actions to add XML substructures, text elements, or attribute values to User Defined Attributes (UDAs) of type XML. In addition, you can delete XML substructures, text elements or attribute values from a UDA of type XML, assign an XML string to a UDA of type XML and extract values from XML data.

11.3.1 Adding a Substructure in XML

When specifying a UDA of type XML, you might want to add an XML substructure to it. This can be achieved using the Add Substructure in XML Java Action.

Prerequisite:

- The process definition has a UDA of type XML to which a new substructure can be added.

Note

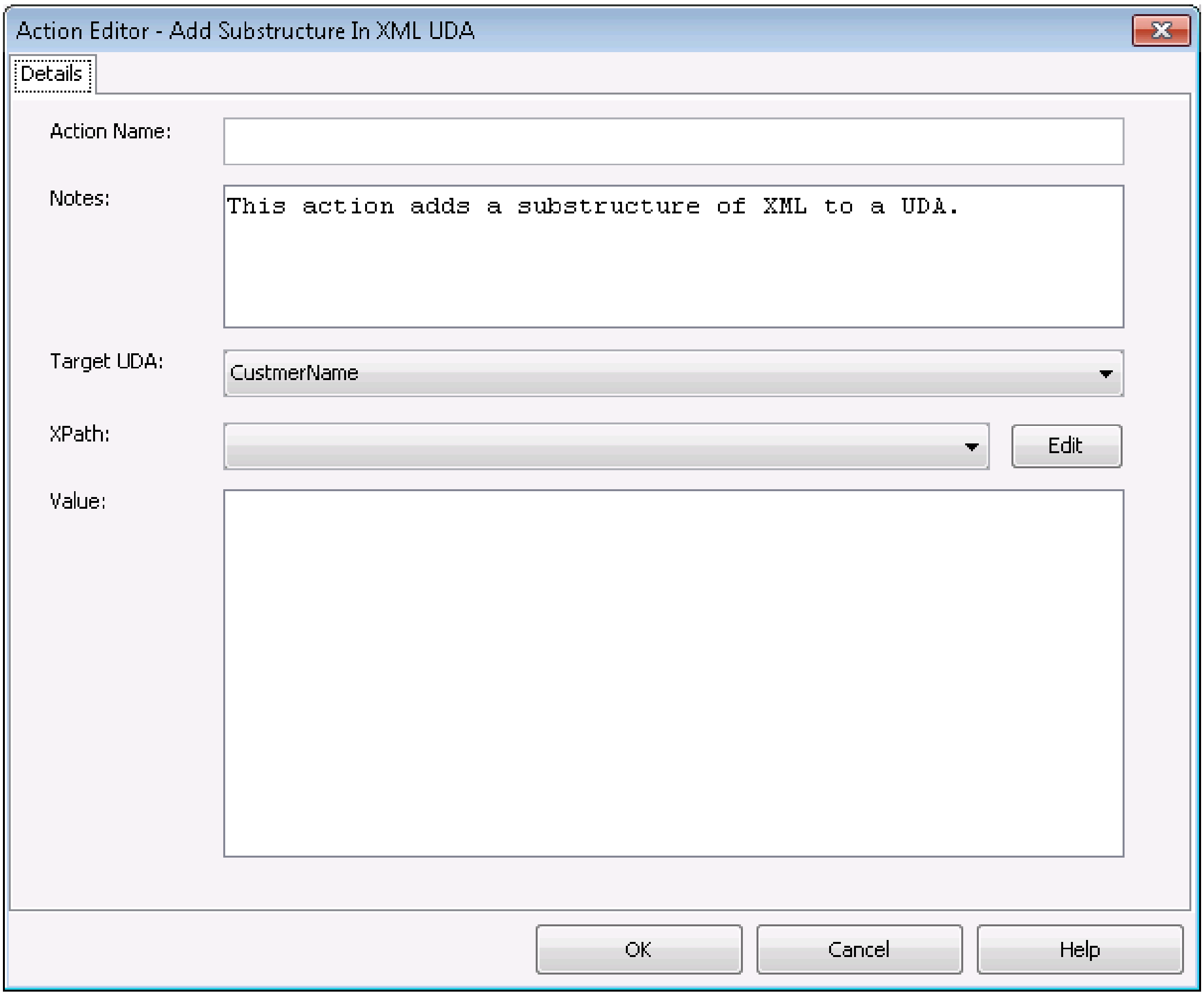
You can assign this Java Action to process definitions and all nodes.

To add a substructure to a UDA of type XML:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **XML Actions** and double-click **Add Substructure in XML**.

   The **Action Editor - Add Substructure In XML UDA** dialog is displayed. It automatically shows the name of the first UDA of type XML you have defined for the process definition or node (CustomerName UDA, in the example), and provides a default description:

   Figure 11.21 Displaying the Action Editor dialog

   Action Editor - Add Substructure In XML UDA

   Details

   Action Name:

   Notes: This action adds a substructure of XML to a UDA.

   Target UDA: CustmerName

   XPath:

   Value:

4. Type a descriptive name and your notes for the Java Action.
5. From the **Target UDA** drop-down list, select the UDA to which the new XML substructure is to be added.

   The **Target UDA** drop-down list displays only UDAs of type XML.
   All the XPaths related to the target UDA (type XML) selected in the **Target UDA** drop-down list, are displayed in the **XPath** drop-down list.
6. Select the XPath expression of the Target UDA from the **XPath** drop-down list.

7. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the **Edit** button located next to the **XPath** drop-down list.
   XPath Editor dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

   Note

   The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

   Note

   The XPath expressions related to the selected XML UDA are displayed in the **XPath** drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

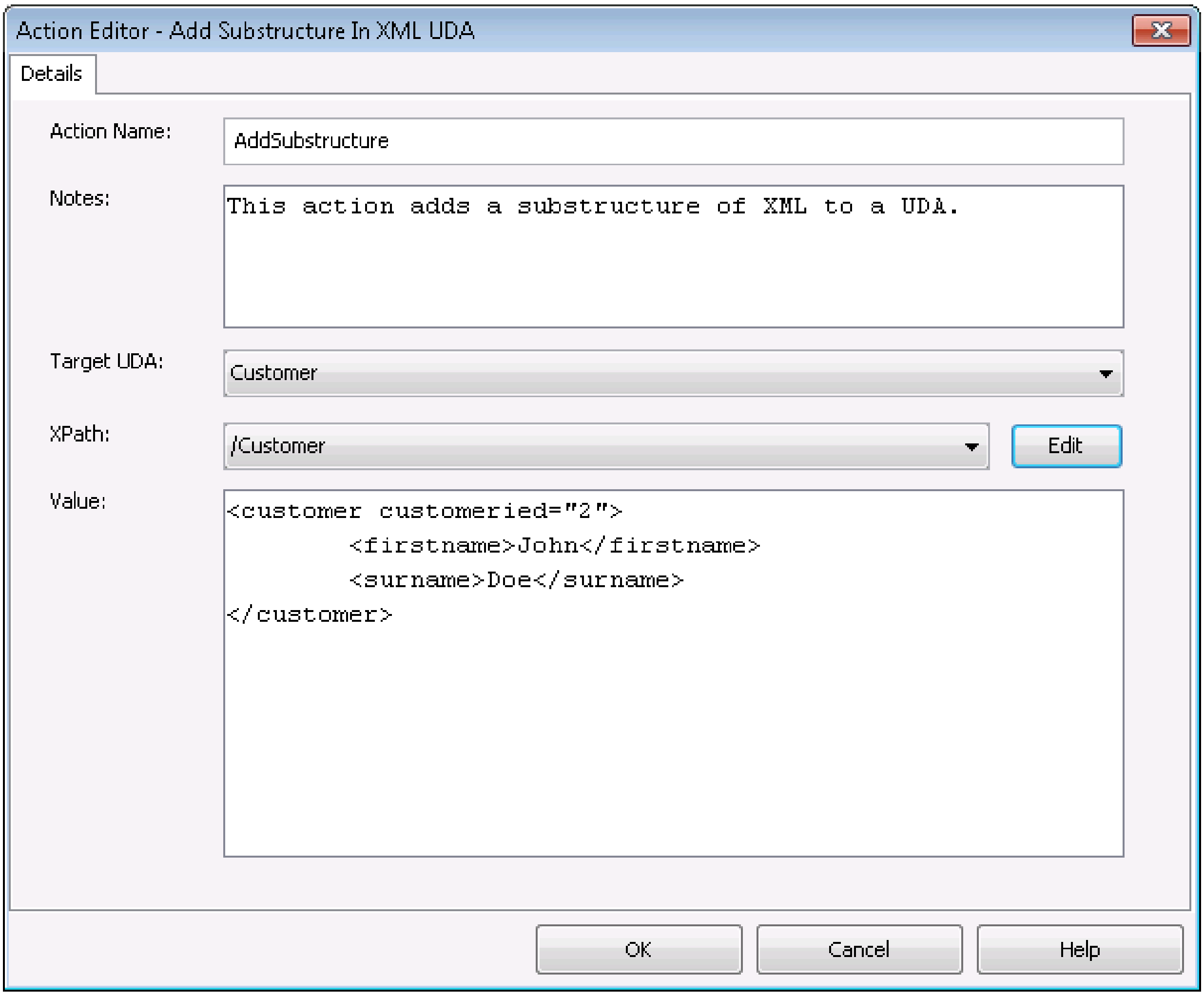
   Note

   XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

8. In the **Value** field, specify the value of the XML substructure.

   When entering a UDA value, use the same tags as for specifying assigning an XML string to a UDA. For more information, refer to sections 11.3.3 Assigning an XML String to a User Defined Attribute The following example shows the Action Editor dialog how the customer substructure is added to the CustomerName UDA:

   Figure 11.22 Adding a Substructure in XML UDA

   Action Editor - Add Substructure In XML UDA

   Details

   Action Name: AddSubstructure

   Notes: This action adds a substructure of XML to a UDA.

   Target UDA: Customer

   XPath: /Customer

   Value:

   ```
   <customer customeried="2">
           <firstname>John</firstname>
           <surname>Doe</surname>
   </customer>
   ```

9. Click **OK**.

   When closing the dialog, Systemwalker Runbook Automation Studio automatically checks whether the entered value is well-formed XML. If it is not, a warning message will be displayed.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation Studio only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.3.2 Extracting User Defined Attribute Values from XML Data

This section explains the procedure for extracting User Defined Attribute (UDA) values from XML data.

Prerequisites:

- The process definition has a UDA that contains an XML string that is to be parsed.
- The process definition has UDAs to which the extracted XML data can be assigned.

You can extract data from an XML string coming in to Systemwalker Runbook Automation and assign the data to UDAs using the Assign UDA from XPath Expression Java Action. You specify where the data can be found in the XML string using an XPath expression.

This Java Action is usually used in conjunction with the Assign XML to UDA Java Action and an HTTP Agent. This three-component system forms a data-transfer interface to a system external to Systemwalker Runbook Automation.

To extract UDA values from incoming XML data:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **XML Actions** and double click **Assign UDA from XPath**.
4. In the **Assign UDA from XPath expression** dialog, type a descriptive name for the Java Action in the **Action Name** field.
5. From the **Source UDA** field, select the UDA that contains the XML string from which the value(s) will be extracted.

   The drop-down list displays all available UDAs. For more details on the supported data types, refer to section 6.18 Specifying User Defined Attributes.

6. Map the data that you want to extract from the XML string to UDAs. To define a mapping:

   a. Click in the field in the **Target UDA** column. A drop-down list is displayed. From the drop-down list, select the UDA to which you want to assign the data.

   b. Click in the field in the **XPath of target UDA** column. A drop-down list is displayed. Select from the drop-down list, the XPath expression that specifies where the data will be stored in the XML string of the target UDA.

      This field is active only if you have selected a UDA of type XML.

      The XPath expressions related to the selected Target UDA are displayed in the **XPath of target UDA** drop-down list.

   c. Optional: If you want to edit the XPath expression that you selected in the **XPath of target UDA** drop-down list, click the ellipsis (**...**) button that is displayed next to the **XPath of target UDA** drop-down list. **XPath Editor** dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

   d. Click in the field in the **XPath of source UDA** column. A drop-down list is displayed. Select from the drop-down list, the XPath expression that specifies where the data can be found in the XML string of the Source UDA.

      This field is active only if you have selected a UDA of type XML.

      The XPath expressions related to the selected Source UDA are displayed in the **XPath of source UDA** drop-down list.

   e. Optional: If you want to edit the XPath expression that you selected in the **XPath of source UDA** drop-down list, click the ellipsis (**...**) button that is displayed next to the **XPath of source UDA** drop-down list. **XPath Editor** dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

   f. If you want to define another mapping, click **Add** and repeat the previous substeps.

   g. If you want to remove a mapping, select it and click **Remove**.

   Note

   The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

   Note

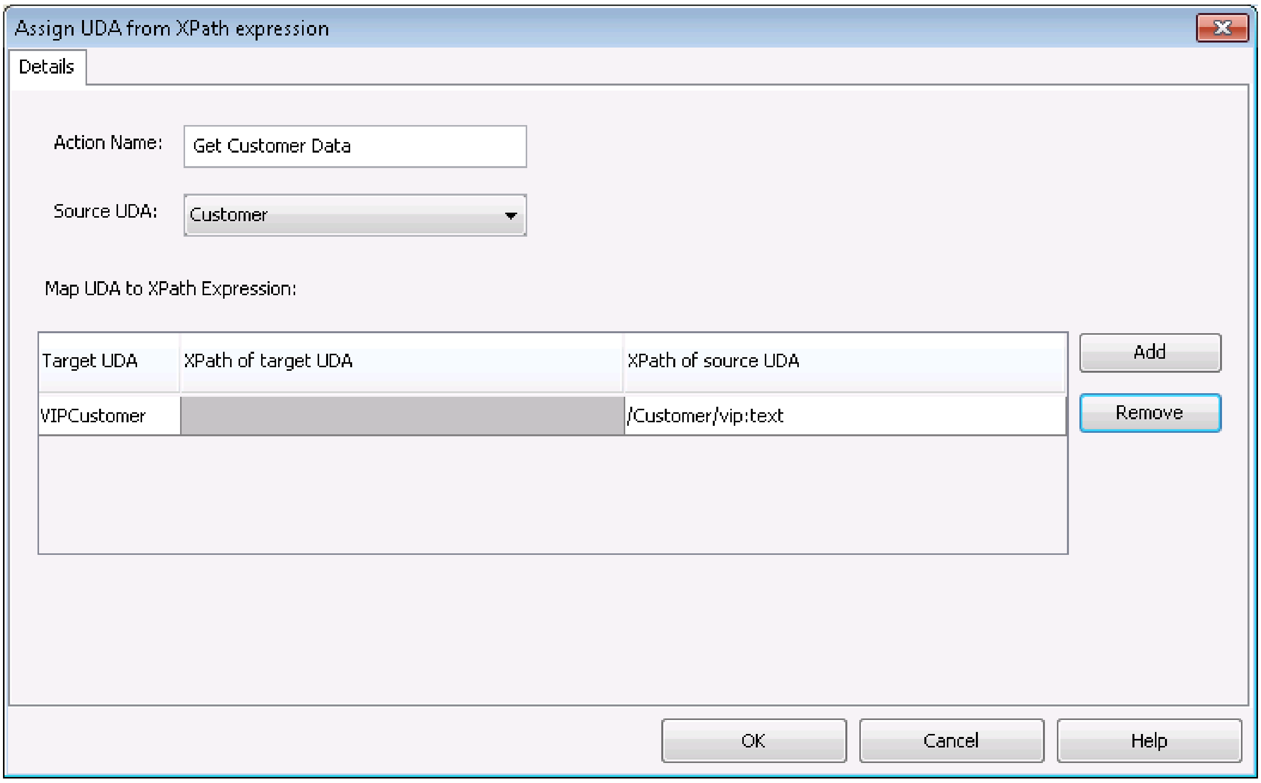
   The XPath expressions related to the selected XML UDA are displayed in the XPath drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

Note

XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

The following figure shows an example where customer data are sent to Systemwalker Runbook Automation as an XML string. They are stored in the Customer UDA. The customer's name is extracted from the XML string and mapped to the VIPCustomer UDA.

Figure 11.23 Extracting Data from an XML String - XPath of source UDA



Assign UDA from XPath expression

Details

Action Name: Get Customer Data

Source UDA: Customer

Map UDA to XPath Expression:

Target UDA	XPath of target UDA	XPath of source UDA
VIPCustomer		/Customer/vip:text

7. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.3.3 Assigning an XML String to a User Defined Attribute

This section explains the procedure for assigning an XML string to a User Defined Attribute (UDA).

Prerequisite:

- The process definition has a UDA to which the XML string can be assigned.

You can generate an XML string and assign it as the value of a UDA using the Assign XML to UDA Java Action. This Java Action is usually used in conjunction with the Assign UDA from XPath Java Action and an HTTP Agent. This three-component system forms a data-transfer interface to a system external to Systemwalker Runbook Automation.

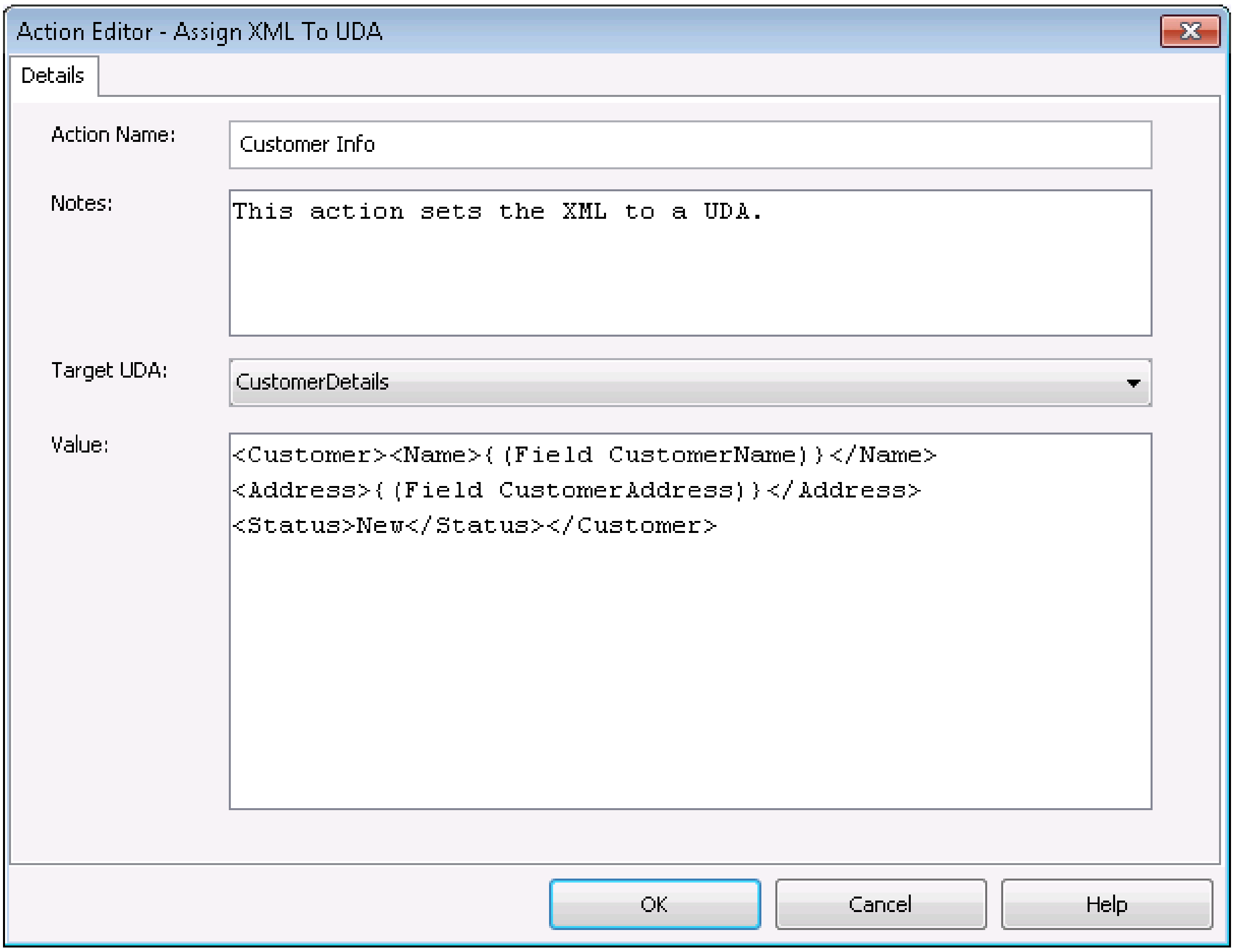
To assign an XML string as the value of a UDA:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.
3. Expand **XML Actions** and double click **Assign XML to UDA**.
4. In the **Action Editor - Assign XML To UDA** dialog, type a descriptive name and your notes for the Java Action.
5. From the **Target UDA** field, select the UDA to which the generated XML string is to be assigned.

   The drop-down list displays only UDAs of type STRING or XML.
6. In the **Value** field, type the XML string to be assigned.

Figure 11.24 Assigning an XML String to a UDA



Action Name: Customer Info

Notes: This action sets the XML to a UDA.

Target UDA: CustomerDetails

Value:

```
<Customer><Name>{{Field CustomerName}}</Name>
<Address>{{Field CustomerAddress}}</Address>
<Status>New</Status></Customer>
```

In the example, Customer Info is executed at process initialization. The Java Action generates the XML string displayed in the above figure and stores it in the CustomerDetails UDA.

Within the XML string, you can use the following tags to specify UDAs or JavaScript code:

- {{Field <UDAName>}}

  This tag can be replaced by UDA value. The value is encoded XML. When it is encoded XML, characters that has a significance meaning will be changed. These characters are less than character (<), greater than character (>), ampersand (&), and double quotation ("). These characters are changed &lt;, &gt;, &amp;, and &quot; by XML encode.
- {{Xml <UDAName>}}

  This tag can be replaced by UDA value. The value is not encoded XML.

  If it is XML fragment passed value, you can use this tag.
- {{Js <JavaScriptExpression>}}

  JavaScript expression is valuated, and this tag is replaced by the evaluation result. The evaluation result is encoded XML.

- {{JsXml <JavaScriptExpression>}}

  JavaScript expression is valuated, and this tag is replaced by the evaluation result. The evaluation result is not encoded XML. If it is XML fragment passed value, you can use this tag.

7. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.3.4 Deleting Substructures, Text and Attribute Values from a UDA

This section explains the procedure for deleting substructures, text and attribute values from a UDA.

Prerequisite:

- The process definition has a User Defined Attribute of type XML from which a substructure, text element or attribute value can be removed.

You might want to remove substructures, text or attribute values from a UDA of type XML. This can be achieved using the Delete from XML Java Action.

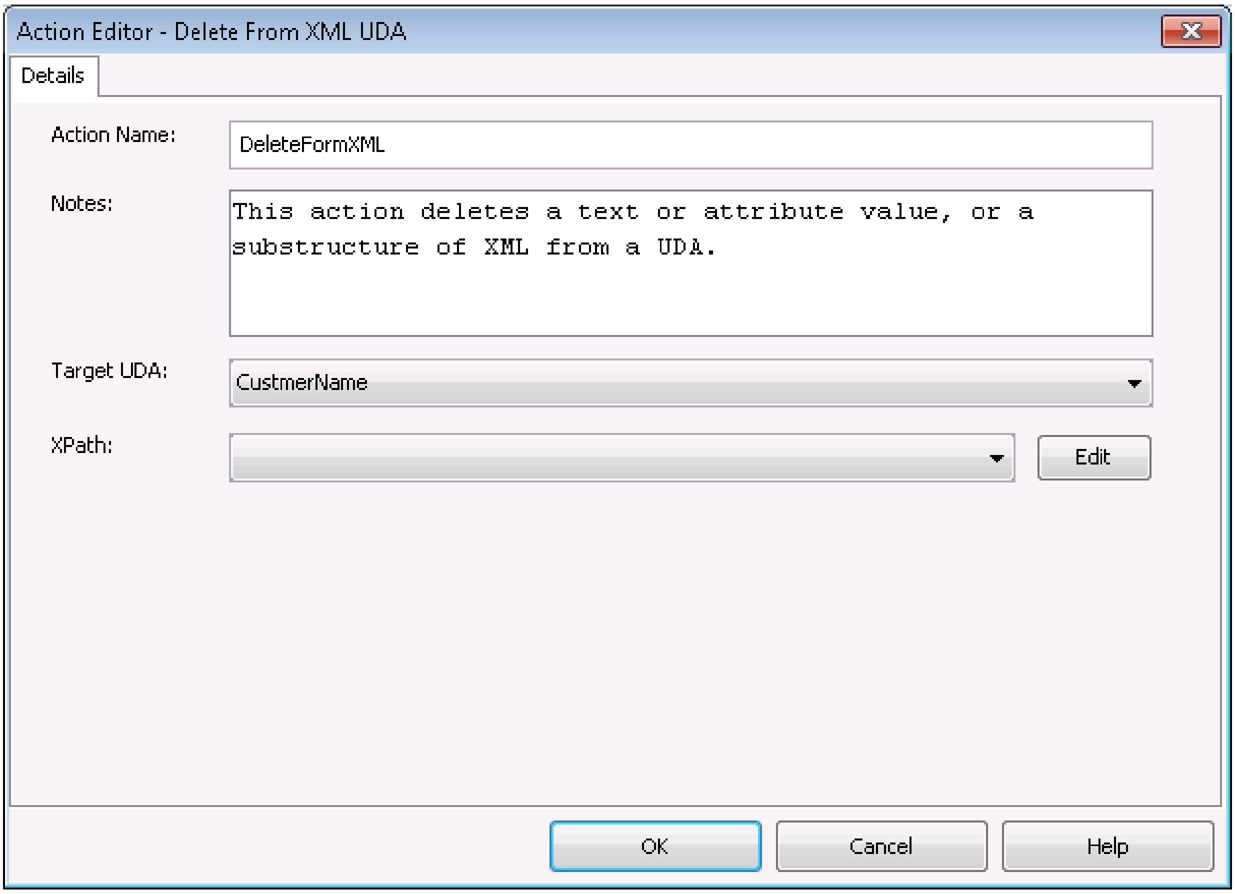
To delete substructures, text and attribute values:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. Expand **XML Actions** and double-click **Delete from XML**.
   The **Action Editor - Delete From XML UDA** dialog is displayed. It automatically shows the first UDA of type XML you have defined for the process definition or node (CustomerName UDA, in the example), and provides a default description:

   Figure 11.25 Displaying the Action Editor dialog

   Action Editor - Delete From XML UDA

   Action Name: DeleteFormXML

   Notes: This action deletes a text or attribute value, or a substructure of XML from a UDA.

   Target UDA: CustmerName

   XPath:

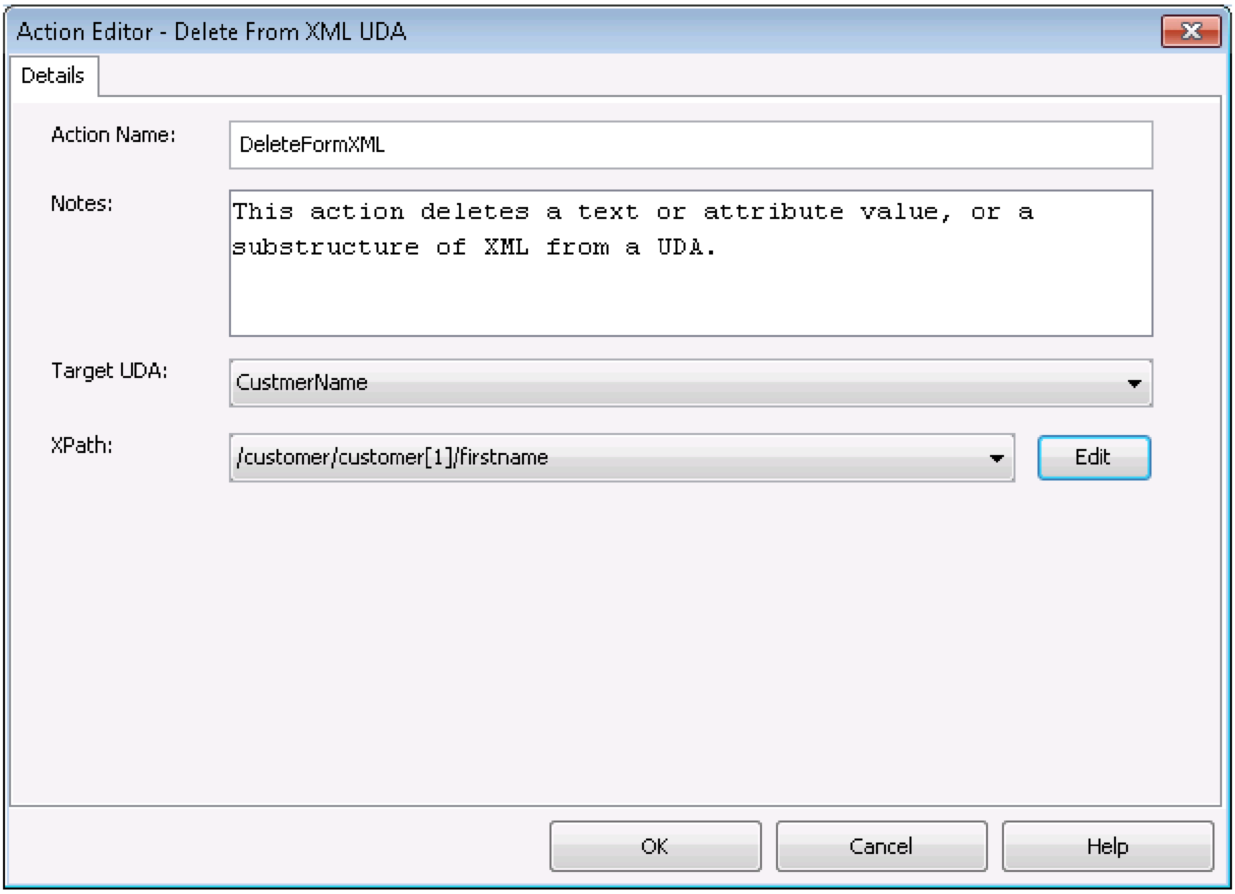
4. Type a descriptive name and your notes for the Java Action in **Action Name** and **Notes** fields respectively.

5. From the **Target UDA** drop-down list, select the UDA from which the substructure, text or attribute value is to be removed.
   The **Target UDA** drop-down list displays only UDAs of type XML.

6. From the **XPath** drop-down, select an XPath expression of the Target UDA.
   The XPath expression refers to that part of the UDA where the substructure, text or attribute value will be removed from. In the

example, the first name is removed from the UDA. Only the surname remains (refer to section 11.3.1 Adding a Substructure in XML, for the complete substructure):

Figure 11.26 Deleting a substructure



Action Name: DeleteFormXML

Notes: This action deletes a text or attribute value, or a substructure of XML from a UDA.

Target UDA: CustmerName

XPath: /customer/customer[1]/firstname

7. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the **Edit** button located next to the **XPath** drop-down list.

   XPath Editor dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

   Note

   The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

   Note

   The XPath expressions related to the selected XML UDA are displayed in the XPath drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

   Note

   XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

8. Click **OK**.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.3.5 Setting a Substructure in XML

You might want to set a new XML substructure to a UDA of type XML. This can be achieved using the Set Substructure in XML Java Action. You can assign this Java Action to process definitions and all nodes.

Prerequisite:

- The process definition has a UDA of type XML whose structure you want to change by replacing an existing substructure.

Note

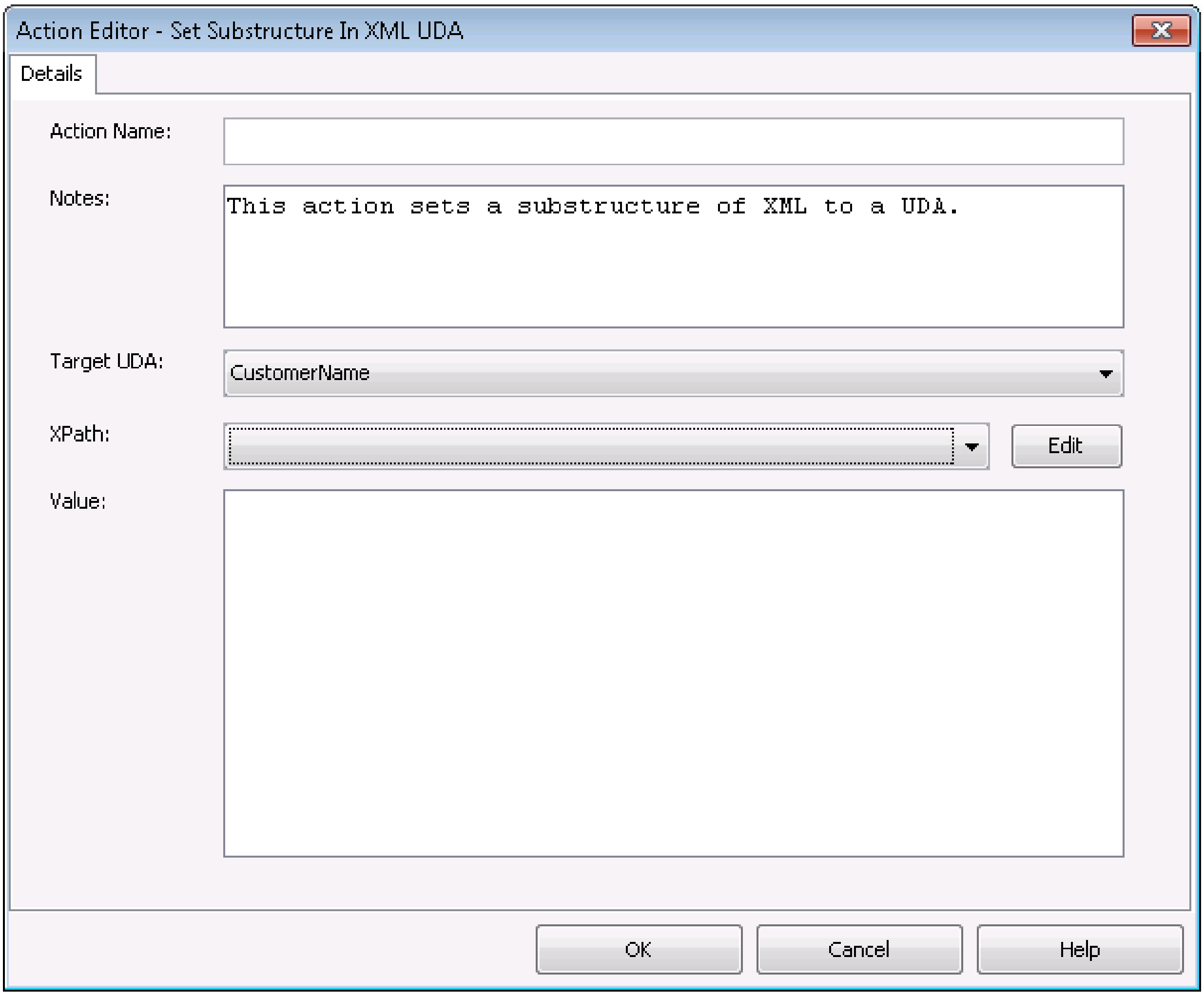
As opposed to the Add Substructure in XML Java Action, the Set Substructure in XML Java Action replaces an existing structure and overwrites it.

To set a substructure in a UDA of type XML:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. Expand **XML Actions** and double-click **Set Substructure in XML**.
   The **Action Editor - Set Substructure In XML UDA** dialog is displayed. It automatically shows the name of first UDA of type XML you have defined for the process definition or node (CustomerName UDA, in the example), and provides a default description:

   Figure 11.27 Displaying the Action Editor dialog

   Action Name:

   Notes: This action sets a substructure of XML to a UDA.

   Target UDA: CustomerName

   XPath:

   Value:

4. Type a descriptive name and your notes for the Java Action in **Action Name** and **Notes** fields respectively.
5. From the **Target UDA** drop-down list, select the UDA in which you want to set a new XML substructure.

   The **Target UDA** drop-down list displays only UDAs of type XML.

   All the XPaths related to the target UDA (XML type) selected in the **Target UDA** drop-down list, are displayed in the **XPath** drop-down list.

6. Select the XPath expression of the Target UDA from the **XPath** drop-down list.
   The XPath expression refers to that part of the UDA where the new text or attribute is to be stored.
7. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the **Edit** button located next to the **XPath** drop-down list.
   XPath Editor dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

   Note

   The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note

The XPath expressions related to the selected XML UDA are displayed in the **XPath** drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

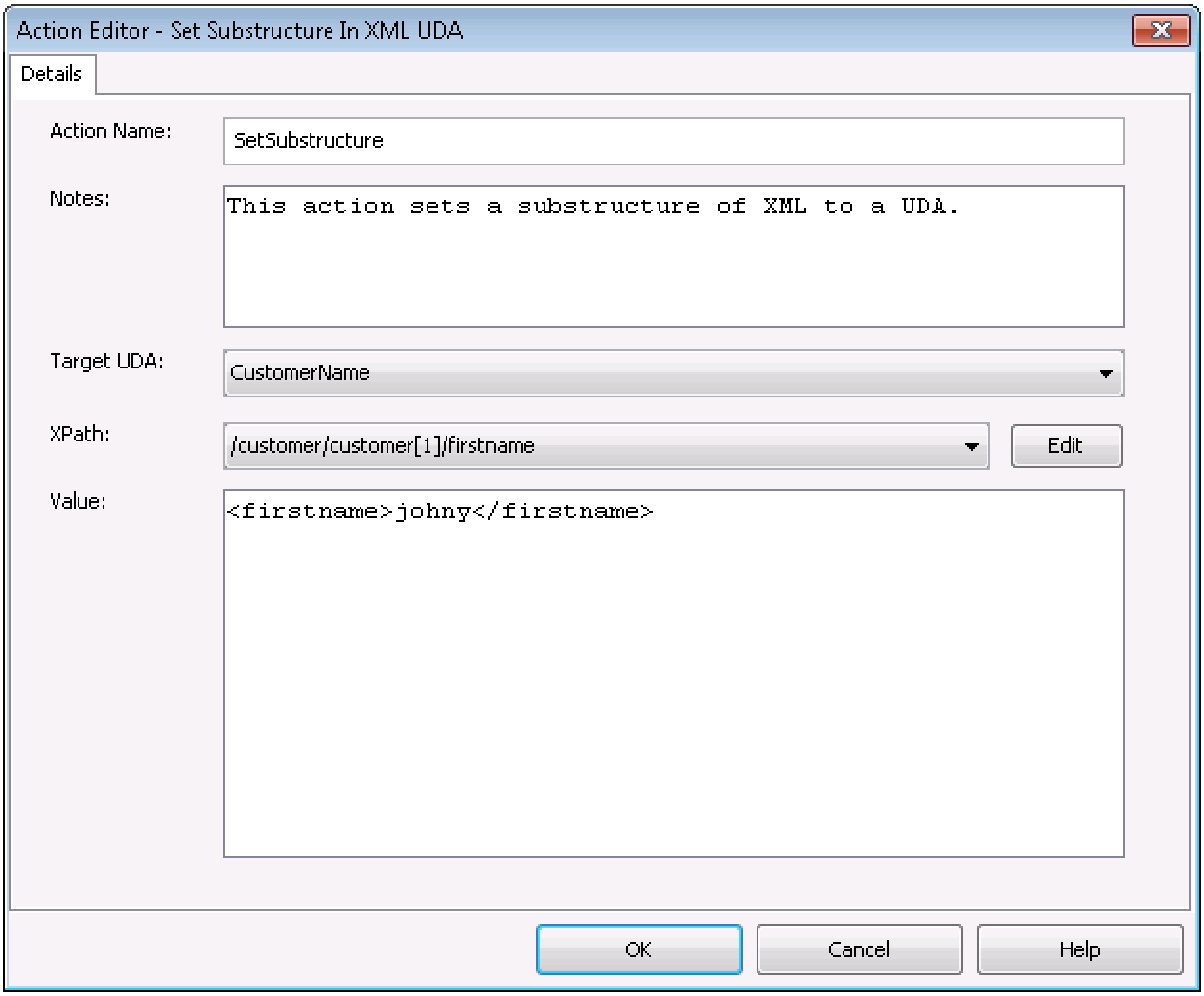
Note

XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

8. In the **Value** field, specify the value of the text or attribute.

   When entering a UDA value, use the same tags as for assigning an XML string to a UDA. For more information, refer to sections 11.3.3 Assigning an XML String to a User Defined Attribute The following example shows the Action Editor dialog used for replacing the customer substructure you have added before (refer to section 11.3.1 Adding a Substructure in XML for more details):

   Figure 11.28 Setting a Substructure in XML



Action Editor - Set Substructure In XML UDA

Details

Action Name: SetSubstructure

Notes: This action sets a substructure of XML to a UDA.

Target UDA: CustomerName

XPath: /customer/customer[1]/firstname

Value: <firstname>johny</firstname>

9. Click **OK** to close the Action Editor dialog.
   When closing the dialog, Systemwalker Runbook Automation Studio automatically checks whether the entered value is well-formed XML. If it is not, a warning message will be displayed.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.3.6 Setting Text or Attribute Values in XML

When specifying a UDA of type XML, you might want to set a new text or attribute value to the UDA. This can be achieved using the Set Text or Attribute Value in XML Java Action.

Prerequisite:

- The process definition has a UDA of type XML for which you want to set a new text or attribute value.

Note

You can assign this Java Action to process definitions and all nodes.

To set a new text or attribute value:

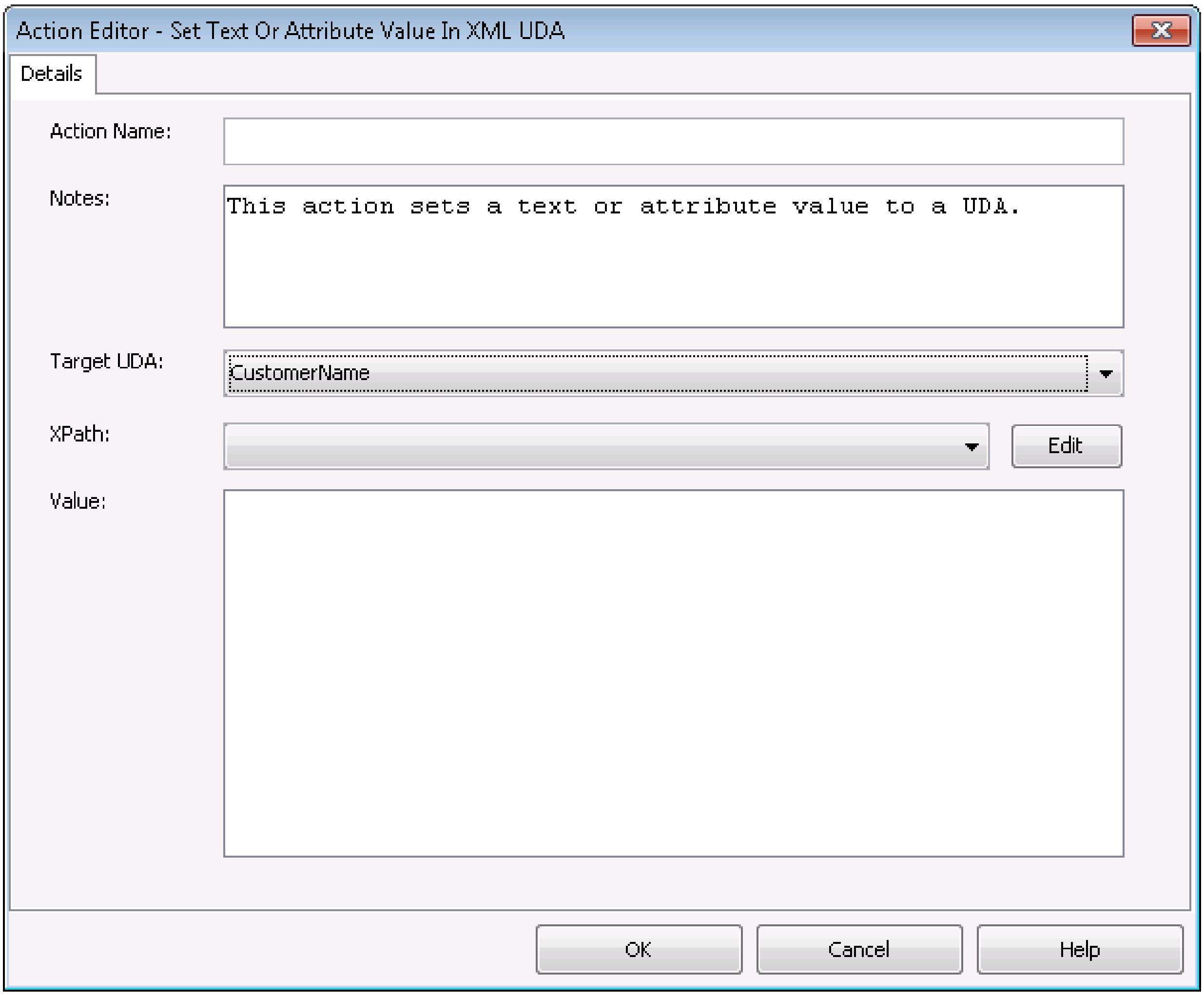
1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. Expand **XML Actions** and double-click **Set Text or Attribute Value in XML**.

   The **Action Editor - Set Text Or Attribute Value In XML UDA** dialog is displayed. It automatically shows the first UDA you have defined for the process definition or node (CustomerName UDA, in the example), and provides a default description:

   Figure 11.29 Displaying the Action Editor dialog



Action Editor - Set Text Or Attribute Value In XML UDA

Details

Action Name:

Notes: This action sets a text or attribute value to a UDA.

Target UDA: CustomerName

XPath:

Value:

4. Type a descriptive name and your notes for the Java Action in **Action Name** and **Notes** fields respectively.

5. From the **Target UDA** drop-down list, select the UDA in which you want to set a text or attribute value.
   The Target UDA drop-down list displays only UDAs of type XML.
   All the XPaths related to the target UDA (XML type) selected in the **Target UDA** drop-down list, are displayed in the **XPath** drop-down list.

6. Select the XPath expression of the Target UDA from the **XPath** drop-down list.
   The XPath expression refers to that part of the UDA where the new XML text or attribute is to be stored.

7. Optional: If you want to edit the XPath expression that you selected in the **XPath** drop-down list, click the **Edit** button located next to the **XPath** drop-down list.
   XPath Editor dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

Note

The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note

The XPath expressions related to the selected XML UDA are displayed in the **XPath** drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

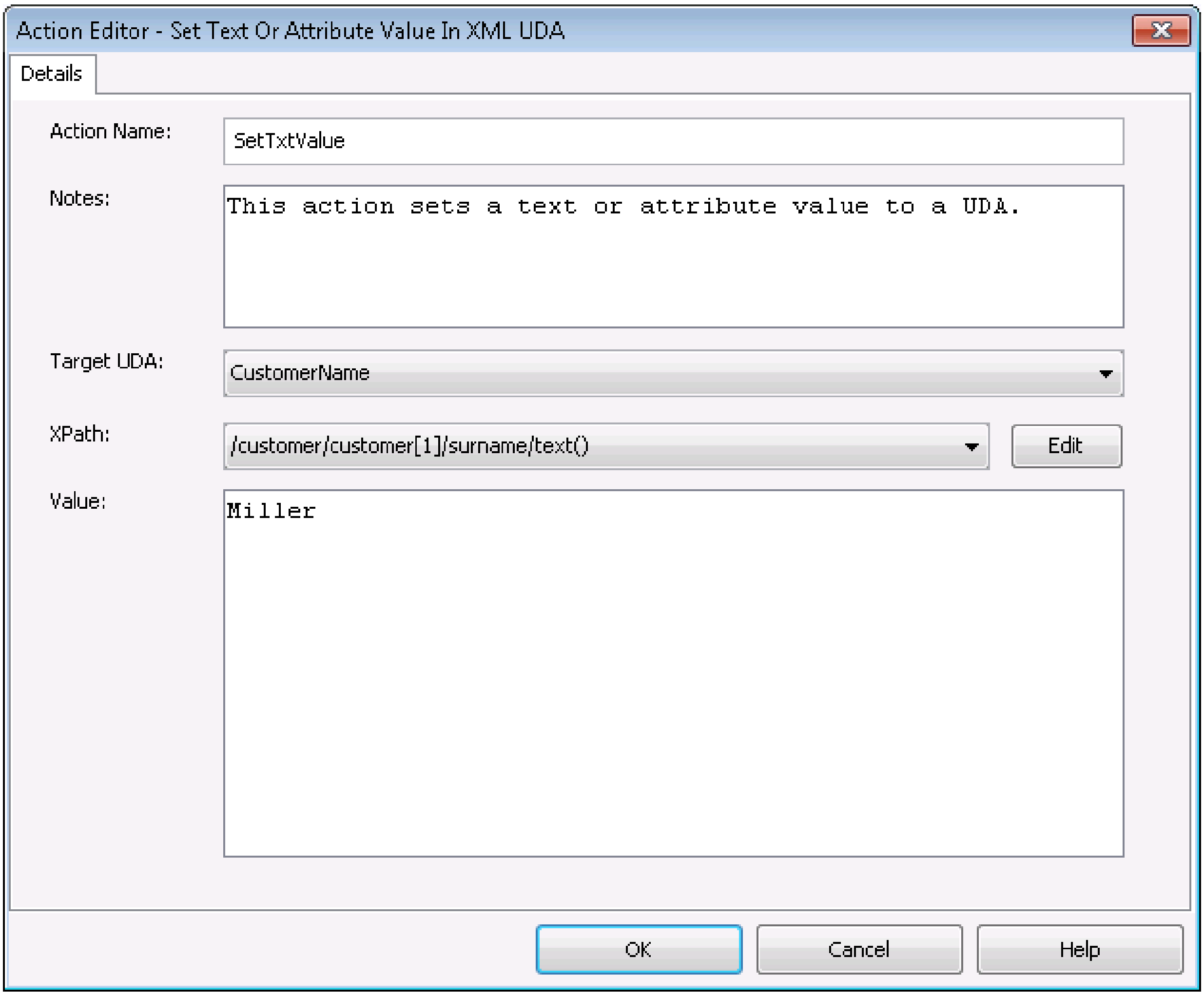
Note

XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

8. In the **Value** field, specify the value of the XML text or attribute.

   When entering a UDA value, use the same tags as for assigning an XML string to a UDA. For more information, refer to sections 11.3.3 Assigning an XML String to a User Defined Attribute The following example shows the Action Editor dialog for changing the last name of the customer from "Doe" (refer to section 11.3.1 Adding a Substructure in XML for the complete substructure) to "Miller". Therefore, use the following XML value and XPath expression:

Figure 11.30 Setting Text or Attribute Values in XML



Action Name: SetTxtValue

Notes: This action sets a text or attribute value to a UDA.

Target UDA: CustomerName

XPath: /customer/customer[1]/surname/text()

Value: Miller

9. Click **OK** to close the Action Editor dialog.

Note

It is not necessary to compensate this action using a compensation action, because changes made by this action are in Systemwalker Runbook Automation only and they will be rolled back after the process instance goes into error state. For information on compensation actions, refer to section 11.1.7 Dealing With Errors in Java Actions.

11.4 Using Rules Actions

Rules Actions enable you to use the functions that provide advanced rule engines with process definitions. The actions described in "Chapter 10 Decision Tables" belong to Rules Actions. Refer to "10.6 Using the Decision Table Action" for more information on these actions.

11.5 Using Notification Actions

Notification Actions notify users on events related to process execution. Users can be notified, for example, that a process or a single activity has been started. Currently, emails can be sent for notification.

11.5.1 Sending Emails

Email messages can be sent using the SendEmail Java Action. You can use this Java Action, for example, to notify anyone that a process or an activity has been started.

You can also attach logs or reports of processes, along with the email.

You can assign SendEmail to any node and to the process definition itself. When using Email Nodes, this is the only Java Action that can be assigned.

To send emails:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. If the **Action Type List** dialog is displayed, expand **Notification Actions** and double click **SendEmail**.

   This operation is not required for Email Nodes.

4. On the **Addresses** tab, in the **Action Editor - Send Email** dialog, specify the users to which emails are to be sent.

   a. Specify a JavaScript expression for the address.
   You can type a constant (that is, an email address), select a User Defined Attribute (UDA) that has an address as its value, or build a complex JavaScript expression that evaluates to an address. For details, refer to section 11.14 Defining JavaScript Expressions.

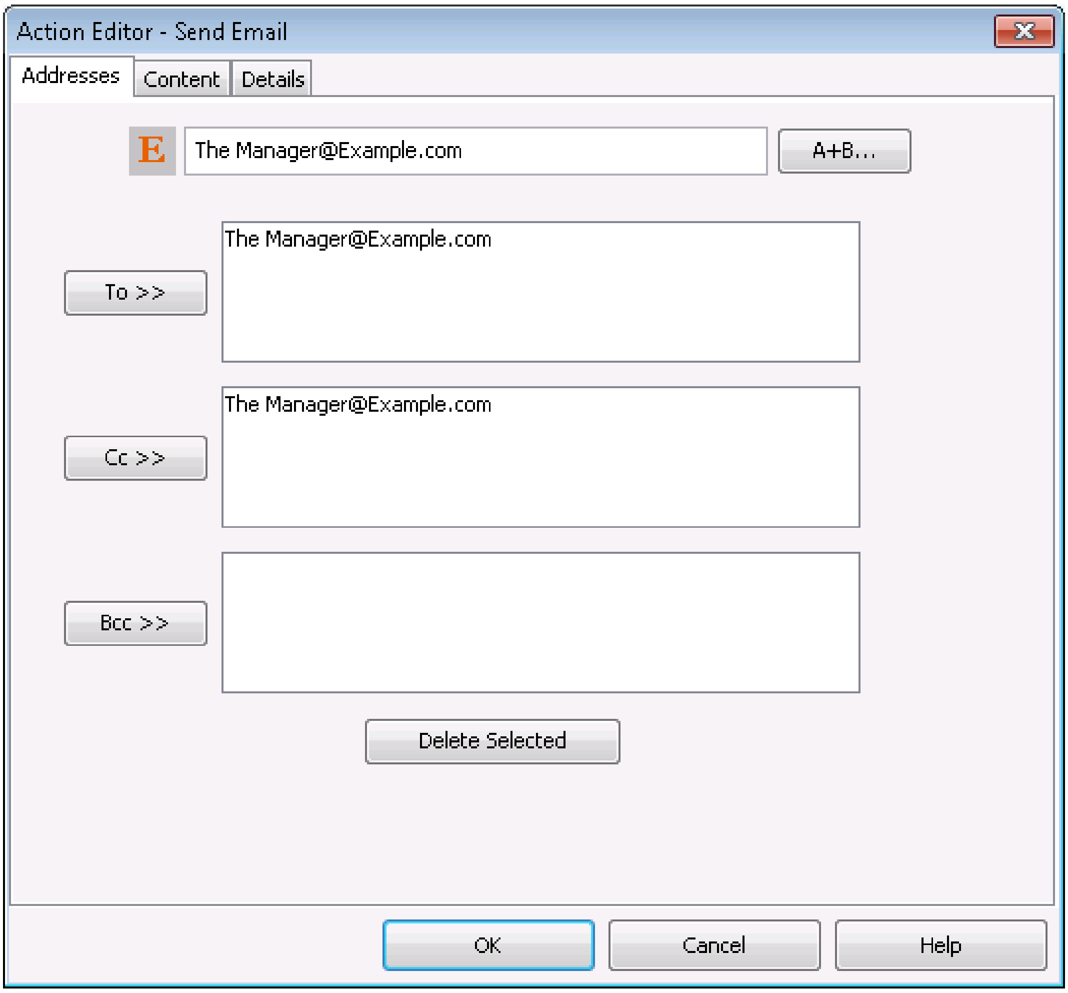
   Note

   If you select a UDA, the UDA must be of type STRING. Otherwise an error will occur when the Java Action is executed.

   b. Click **To**, **Cc**, or **Bcc** depending on how you want the message to be addressed.

c. If you want to remove an address, select it and click **Delete Selected**.

Figure 11.31 Using Addresses



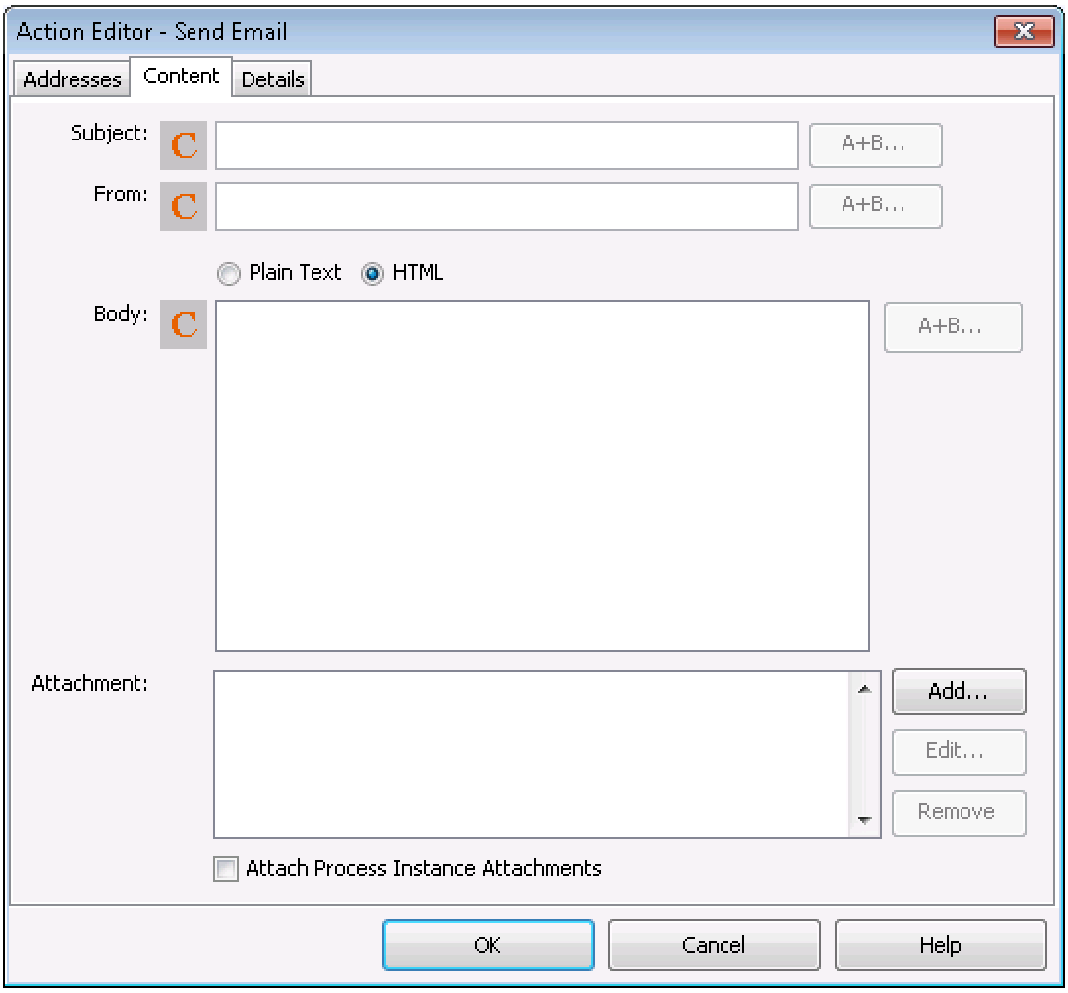
The Manager@Example.com

The Manager@Example.com

The Manager@Example.com

5. On the **Content** tab, specify JavaScript expressions for the **Subject** and **From** fields and the message body. Select the format in which you want the email to be sent.

Figure 11.32 Specifying Email Content



6. Optional: To send an attachment with the email:

   a. Click the **Add** button located near the **Attachment** field. **Attachment** dialog is displayed.

   Note

   You can select an attachment in the **Attachment** field and click **Edit** to edit it or click **Remove** to remove the selected attachment. When you select an attachment and click **Edit**, **Attachment** dialog is displayed.

   b. Attach a file using either the **Browse** button or the expression mode button. To attach a file by browsing, click the **Browse** button. **Select or enter file location and name of source file** dialog is displayed. This dialog displays the tree view of the files in **dms** folder of the application. You can browse the file that you want to attach with the email using this dialog. You can also enter the path to the file in the **Path** field and select the required file in the dialog.

Note

Refer 11.14 Defining JavaScript Expressions for more information about expression mode button.

Figure 11.33 Attaching File

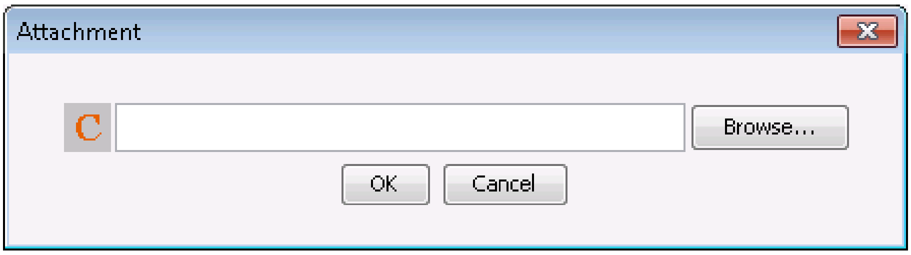
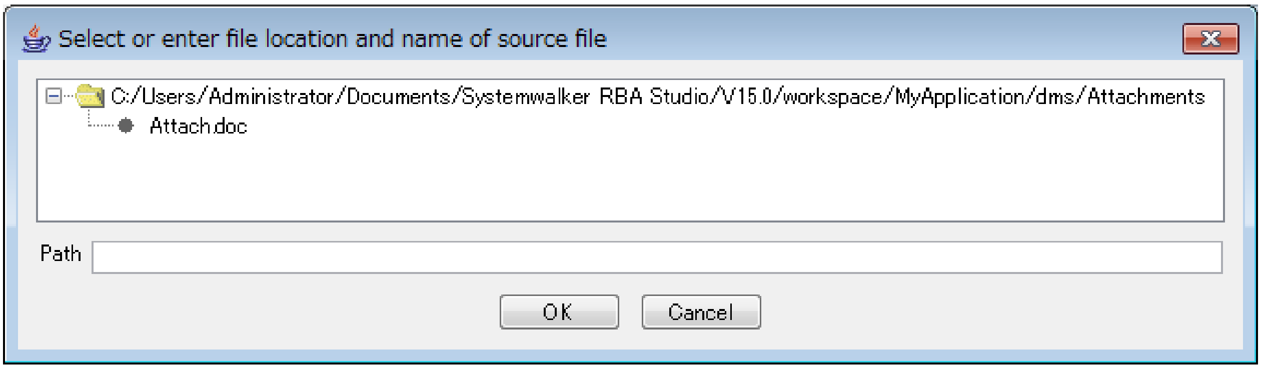


Figure 11.34 Browsing for Attachment


Select or enter file location and name of source file

C:/Users/Administrator/Documents/Systemwalker RBA Studio/V15.0/workspace/MyApplication/dms/Attachments

Attach.doc

Note

The value of attachment is specified by the relative path from the route of the application project. The wild-card cannot be used. For example, **dms/Attachments/abc.doc** or **uda.get("DMSattachment") + ".txt"**.

c. Click **OK** on **Attachment** dialog.

7. Optional: On the **Action Editor - Send Email** dialog, select the **Attach Process Instance Attachments** checkbox to attach the process instances with the email.

8. On the **Details** tab, type a descriptive name and your notes for the Java Action in **Action Name** and **Notes** fields respectively.

9. Click **OK**.

11.6 Using Generic Java Actions

This section explains the procedure for using generic Java actions.

Prerequisites:

- You have implemented a Java class whose methods can be integrated as Generic Java Actions.
- The process definition has User Defined Attributes (UDAs) for all parameters that are to be passed to the method.
- If the method has a return value, the process definition must have a UDA to which the return value can be assigned.

Note

- If you do not add the correct UDAs to your process definition, the Java Action may work unpredictably.

- Store the Java class to be independently implemented in the class file format. Java Actions cannot be called for files in the jar file format.

Generic Java Actions allow you to execute Java methods that are outside the scope of Systemwalker Runbook Automation. They enable you to customize process execution using methods of Java classes that you have implemented yourself.

To assign a Generic Java Action:

1. Copy the Java class that you want to use. You must differentiate whether you are working with a Workflow Application project or a Server project:

    - **Workflow Application project**:

      In the Workflow Application project's folder structure, store the Java class file(s) to the
      Application Classes >> engine >> classes

    - **Server project**:

      Copy the Java class to the <Systemwalker Runbook Automation Studio Installation Directory>\ibpm\Data\attachments folder.

2. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.

3. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

4. Double click **Generic JavaAction**. The **Action Properties** dialog is displayed.

5. Type a descriptive name for your Java Action in the **Action Name** field.

6. Click the Browse button [...] and select the Java class that you want to use.

7. Select the method that you want to call from the **Method Name** list.

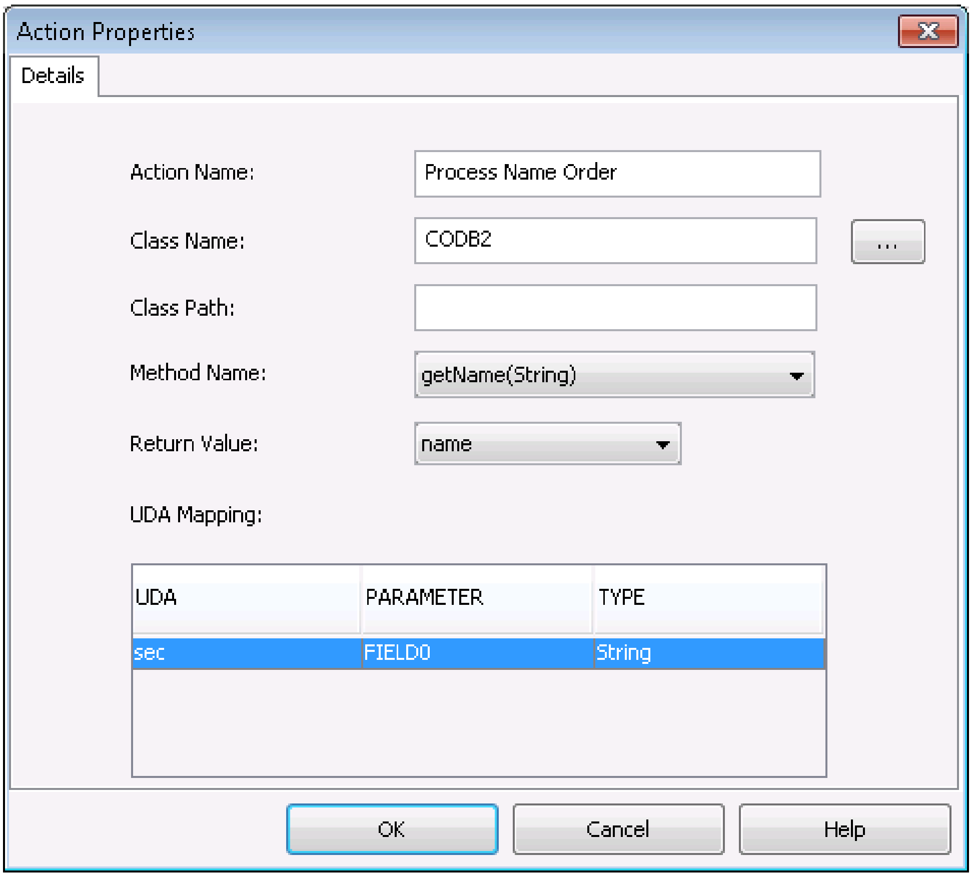
    Note

    A method will be available only if all parameters are available as UDAs and if their data types match.

8. From the **Return Value** list, select the UDA that will receive the return value, if any, from the method.

9. Check the UDA Mapping. Verify that the UDAs are mapped correctly to the method parameters. These UDAs will provide input values to the method. To change a mapping, select another UDA from the drop-down list that appears when you click a UDA.

Figure 11.35 Assigning a Generic Java Action



Action Properties

Details

Action Name: Process Name Order

Class Name: CODB2

Class Path:

Method Name: getName(String)

Return Value: name

UDA Mapping:

UDA	PARAMETER	TYPE
sec	FIELD0	String

10. Click **OK**.

Note

When you send the process definition to a Management Server, or deploy your Workflow Application project, the generic Java Actions are automatically sent to the Management Server.

11.7 Defining No-Operation Java Actions

No-Operation Java Actions are built-in Java Actions that specify no operation. These Java Actions allow you to catch a Java Action exception without executing any additional actions. Using No-Operation Java Actions, Systemwalker Runbook Automation simply moves on to the next sequential instruction.

Note

You can assign No-Operation Java Actions to process definitions and all nodes.

To define a No-Operation Java Action

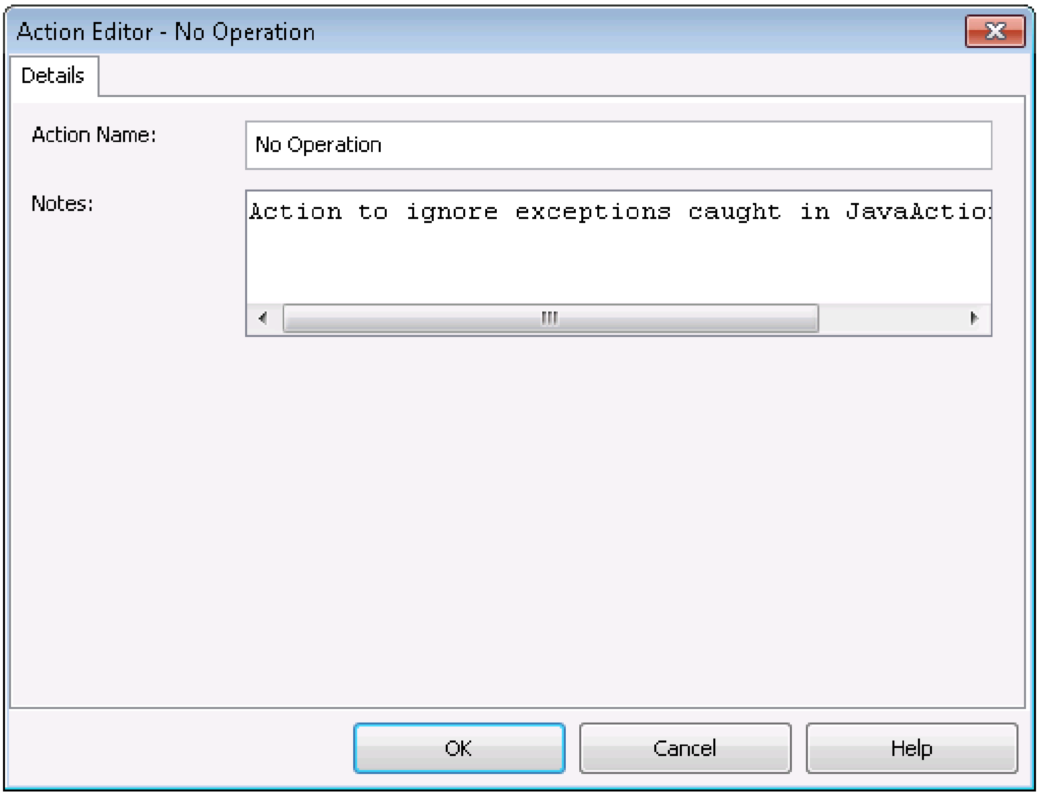
1. Click the empty space in the Process Definition editor or select the node to display the Properties view for the process definition or the node respectively.

2. Select the **Action Set** tab. Click **Add** corresponding to the position where you want to add the Java Action. The **Action Type List** dialog is displayed.

3. Double-click **No-Operation Java Action**.

4. In the **Action Editor - No Operation** dialog, type a descriptive name and your notes for the Java Action. Providing information about the new Java Action is optional.

Figure 11.36 Defining a No-Operation Java Action



Action Name: No Operation

Notes: Action to ignore exceptions caught in JavaActio

5. Click **OK**.

No-Operation Java Action Sample

The following example shows how to use a No-Operation Action.

In the example, you have created a regular Java Action, for example a Sending an email message Java Action. For this Java Action, you have defined a No-Operation as an Error Action. An error might occur if the recipient's mailbox is full so that all incoming emails will be bounced back to the sender. If executing the process instance is more important than sending the email, you can define a No-Operation Action as an Error Action. This assures continuing the process instance regardless of an error.

11.8 Defining Exception Handling

With Systemwalker Runbook Automation Studio, you can define exception handling behavior as follows:

- **Error Actions** can be used to handle specific errors and to determine the behavior of a process instance in case an error occurs. If you do not define any error handling, a process instance will go into error state as soon as an exception is thrown, irrespective of when the error occurs, for example the sending of an email was unsuccessful.

  Error Actions can be defined on different levels:

  - On **Process Definition level**:
    These Error Actions are executed in case of any error, independent of the activity in which an error occurs and independent of the severity of an error. Error Action Sets defined on process definition level will be executed immediately before the process instance will go into error state. Note that such Error Actions cannot influence the behavior of the process instance. Error Actions on process definition level may, for example, be used for sending a notification email or for writing additional information into a log file. Refer to section 11.8.1 Using Error Actions on Process Definition Level for details.

- On **Java Action level**:
  An Error Action Set on this level is executed when an error occurs during the execution of a "regular" Java Action. Error Actions can be defined for all types of Java Actions, except for an Error or Compensation Action. Assigning Error Actions to Java Actions is described in section 11.1.7 Dealing With Errors in Java Actions.

- **OnSuspend**, **OnResume**, and **OnAbort Actions** (On*Actions) can be used to deal with the situations when an Administrator suspends, resumes or aborts the execution of a process instance. For example, you can use these Actions to send a notification email. Refer to section 11.8.2 Using OnSuspend, OnResume, and OnAbort Actions for details.

11.8.1 Using Error Actions on Process Definition Level

Error Actions on process definition level react on any kind of exception, independent of the activity in which an exception occurred, and independent of how severe a problem is. Error Actions on process definition level are typically used to execute general Java Actions, for example sending an email or writing important information into a log file.

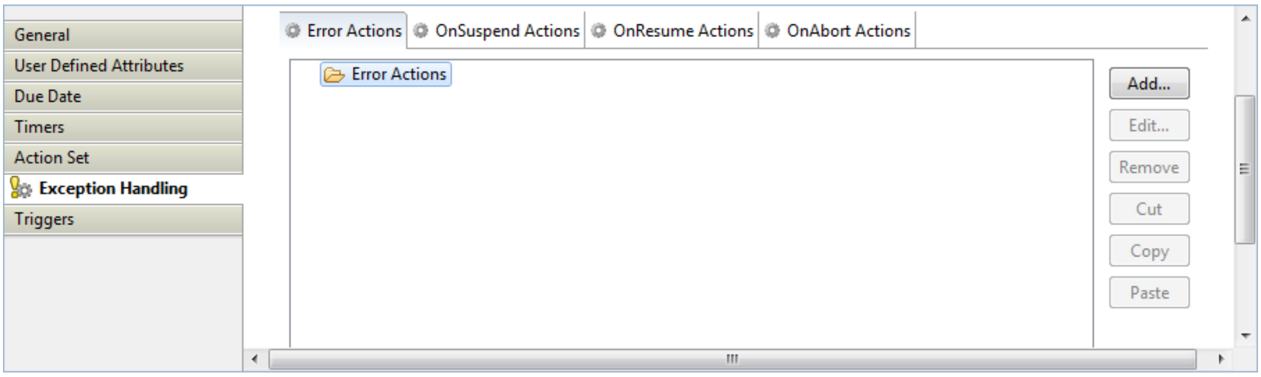
This section explains how to define Error Actions for process definitions.

To define Error Actions on process definition level:

1. Click the empty space in the Process Definition editor to display the **Properties** view for the process definition.

2. Select the **Exception Handling** tab.
   Here you can add new Error Actions to your process definition. If you have not yet defined any Error Action, the dialog displays an empty Error Actions folder.

   Figure 11.37 Displaying the Exception Handling Tab



Note

In this tab, you can also create OnSuspend, OnResume and OnAbort actions. These actions are not used to define particular error processing behavior - they are used when the process instance status has been changed by an administrator. Refer to section 11.8.2 Using OnSuspend, OnResume, and OnAbort Actions for more details.

3. Click **Add**.

   The **Action Type List** dialog is displayed.

4. Proceed with defining a Java Action as an Error Action. Refer to section 11.1 Using Java Actions for details.

5. Click **OK**.

   The new Error Action will react to any kind of exception that is thrown during process execution. You can assign the Error Action to a regular Java Action specified for a process definition. Refer to section 11.1.7 Dealing With Errors in Java Actions for more details.

11.8.2 Using OnSuspend, OnResume, and OnAbort Actions

OnSuspend, OnResume, and OnAbort Actions (On*Actions) are executed just before a process instance changes its state due to the fact that an administrator has issued the command to suspend, resume or abort the execution of a process instance.

Note

On*Actions are not related to exceptions or error cases. They are regular Java Actions that are executed when the command for a state transition has been invoked. Apart from this special use, they are treated like regular Java Actions.

On* Actions can be defined for individual activities (for example individual nodes) or for process definitions. Assume you define, for example, an OnSuspend Java Action both for a node and for a process definition. If this node is active when an Administrator issues the command to suspend the process instance, the Java Actions are executed in the following order:

1. The OnSuspend Java Action of the Node is executed.
2. The OnSuspend Java Action of the process definition is executed.

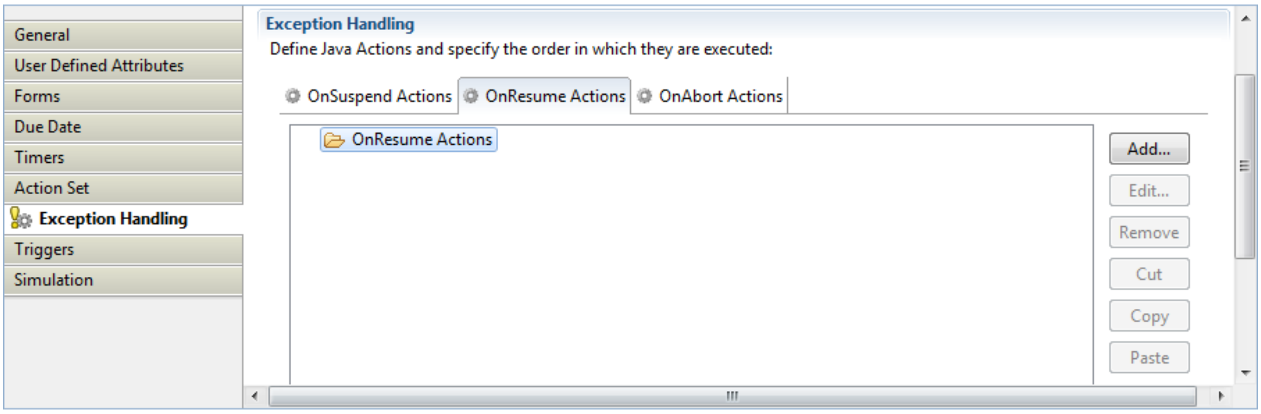
Similar to all other Java Actions, On* Action Sets can contain a number of other Java Actions.

To define On* Actions:

1. Click the empty space in the Process Definition editor or select the node to which you want to assign the Java Action, to display the **Properties** view for the process definition or the node respectively.
2. In the Properties view, select the **Exception Handling** tab.

   The **Exception Handling** dialog is displayed. If you have not defined any On* Actions before, an empty folder is displayed for each of the three actions, for example:

   Figure 11.38 Displaying On* Actions

   Exception Handling

   Define Java Actions and specify the order in which they are executed:

   OnResume Actions

3. Select the tab for the On* Action you want to create, and click **Add**.

   The **Action Type List** dialog is displayed.

4. Proceed with defining a Java Action as an Error Action. Refer to section 11.1 Using Java Actions for details.
5. Click **OK**.

11.9 Using Triggers

In Systemwalker Runbook Automation, triggers move data coming from an external system into process instances.

The data comes in as an XML file. This file is also referred to as a Data Event file. The external system stores the XML file in a particular directory configured on the Management Server. In response to the incoming data, a trigger either starts a process instance or makes a choice on a particular activity. It also maps incoming data to User Defined Attributes (UDA) and thus makes it available for further processing.

Depending on what the trigger is supposed to do (start a process instance or make a choice), you define it either

- on process definition level for starting a process instance
- on node level (Activity Node) for making a choice

The following sections explain how to prepare for triggers and how to define them.

11.9.1 Preparing for Using Triggers

This section explains the procedure for preparing for the use of triggers.

When defining a trigger, you will be mapping the incoming XML data to User Defined Attributes (UDAs). To simplify data mapping, you are recommended to provide an XML schema (*.xsd file) that describes the format of the incoming XML data. This way, the elements of the XML file are available as a drop-down list in Systemwalker Runbook Automation Studio and can be easily mapped to UDAs. If you don't provide an XML schema, you need to specify the location of the elements to be mapped using XPath expressions.

To prepare for using triggers:

1. Recommended: Provide an XML schema that describes the format of the incoming data:

    a. Create an XML schema.

    b. Store the XML schema on a Web Server to which Systemwalker Runbook Automation Studio has access.

    c. Ensure that you know the URL of the XML schema.

    For example, if you have stored the XML schema in the root directory of the Web Server, the URL would be:

    ```
    http://<computername>/<xml schema name>
    ```

2. To properly map XML data, you will need to know the element names, data types, and element description used in the incoming XML file.

    As an example, the following table lists the information you need to know if the incoming XML files were order forms:

Element Name	XSD Data Type	Element Description
orderperson	string	Name of the person taking the order
name, address, city, state, zip	string	Name, address, etc. of the person making the order
title, note	string	Item being ordered
quantity	positiveInteger	Quantity of the item being ordered
price	decimal	Price of the item being ordered

3. Before starting process instances on the Management Server, ensure that the File Listener is configured to check for incoming files. Refer to the "11.13 Defining File Listener" for information on how to configure the File Listener.

11.9.2 Defining Start Process Triggers

A Start Process Trigger starts a process instance in response to data that comes in from an external system.

To define a Start Process Trigger:

1. Add the User Defined Attributes (UDAs) you will need for your trigger to the process definition.

2. Click the empty space in the Process Definition editor to display the **Properties** view for the process definition.

3. Select the **Triggers** tab.

4. Click **Add**.

    A new trigger with a default name is added.

5. Select the trigger to view the details for that trigger in the **Trigger Details** area.

6. Modify the name of your trigger as required on the **General** sub-tab, in the **Trigger Details** area.

7. Optional: Describe the purpose of your trigger on the **General** sub-tab.

8. Select the **Event** sub-tab.

    If you have an XML schema that describes the format of the incoming data, type its URL in the **URL to XML Schema** field. Click **Retrieve** to load the XML schema.
    The icon next to the **URL to XML Schema** field tells you if the XML schema has been loaded.

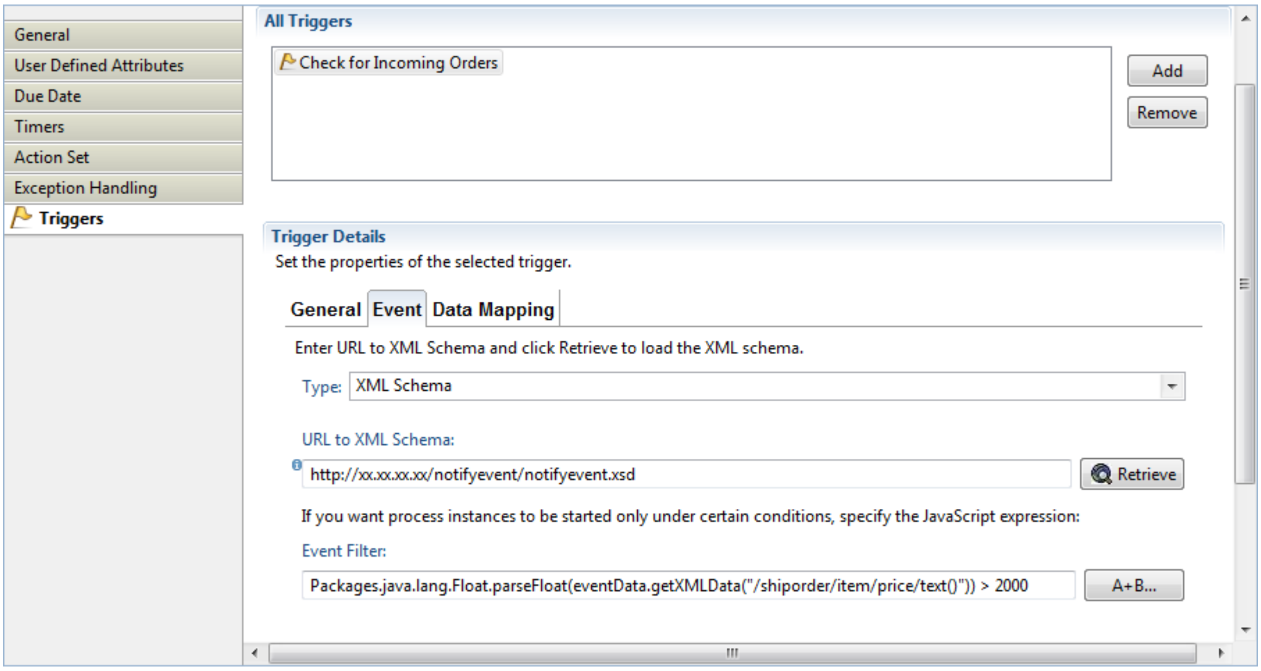
9. If you want process instances to be started only under certain conditions, specify a JavaScript expression in the **Event Filter** field that defines those conditions.

   For information on JavaScript expressions, refer to section 11.14 Defining JavaScript Expressions.

   The following is an example of a JavaScript expression that starts a process instance only if the price of the item is greater than $20.00:

```
Packages.java.lang.Float.parseFloat(eventData.getXMLData("/shiporder/item/price/text()"))> 20
```

Figure 11.39 Specifying an Event Filter



Check for Incoming Orders

Set the properties of the selected trigger.

Enter URL to XML Schema and click Retrieve to load the XML schema.

Type: XML Schema

URL to XML Schema: http://xx.xx.xx.xx/notifyevent/notifyevent.xsd

If you want process instances to be started only under certain conditions, specify the JavaScript expression:

Event Filter: Packages.java.lang.Float.parseFloat(eventData.getXMLData("/shiporder/item/price/text()")) > 2000

10. Select the **Data Mapping** sub-tab. Map the elements of the incoming XML file to UDAs. To map an element:

    a. Click **Add**.

    b. If you specified an XML schema, select an XML element from the **Event Element** list.

    c. Select the UDA from the **Variable** list to which you want to map the XML element.

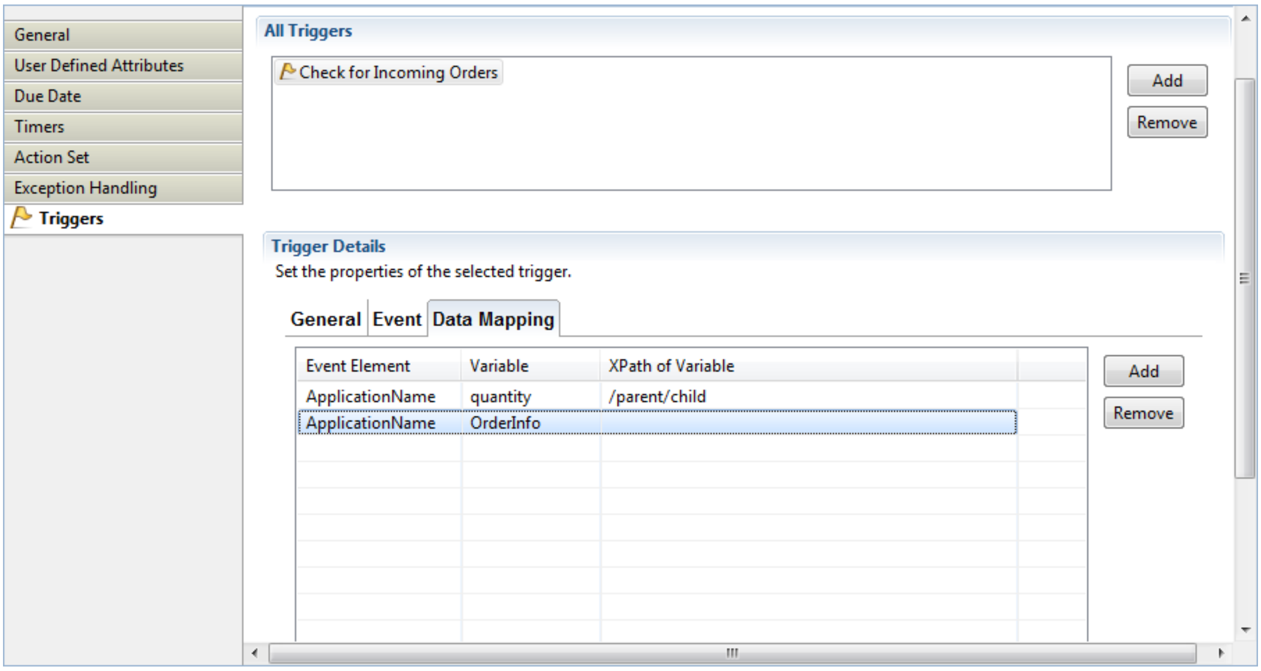
Note

Triggers are flexible enough so that you can map external data of type String to a UDA of type INTEGER as long as the external data consists only of numerals.

a. In the **XPath of Variable** field, type in an XPath expression of the selected UDA. This expression is used to map the incoming XML file to a specific attribute of the UDA.

The XPath field is activated only if you have selected a UDA of type XML from the **Variable** list.

Figure 11.40 Defining Data Mappings



Event Element	Variable	XPath of Variable
ApplicationName	quantity	/parent/child
ApplicationName	OrderInfo	

b. If you want to remove a mapping, select it and click **Remove**.

11. When you have completed the data mapping, select the **General** sub-tab. Select the **Enable** check box to enable your trigger.

11.9.3 Defining Make Choice Triggers

A Make Choice Trigger makes a choice on an activity in response to data that comes in from an external system.

To define a Make Choice Trigger:

1. Add the User Defined Attributes (UDAs) you will need for your trigger to the process definition.
2. Select the Activity Node to display the **Properties** view for it.
3. Select the **Triggers** tab.
4. Click **Add**.

   A new trigger with a default name is added.
5. Select the trigger to view the details for that trigger in the **Trigger Details** area.
6. Modify the name of your trigger as required on the **General** sub-tab, in the **Trigger Details** area.
7. Optional: Describe the purpose of your trigger on the **General** sub-tab.
8. Select the **Event** sub-tab.
9. If you have an XML schema that describes the format of the incoming data, type its URL in the **URL to XML Schema** field. Click **Retrieve** to load the XML schema.

   The icon next to the **URL to XML Schema** field tells you if the XML schema has been loaded.
10. If you want the trigger to fire only with certain incoming data, specify a JavaScript expression in the **Event Filter** field that defines those conditions.

    For an example, refer to section 11.9.2 Defining Start Process Triggers. For information on JavaScript expressions, refer to section 11.14 Defining JavaScript Expressions.

11. Select the **Data Mapping** sub-tab. Map the elements of the incoming XML file to UDAs. To map an element:

    a. Click **Add**.

    b. If you specified an XML schema, select an XML element from the **Event Element** list.

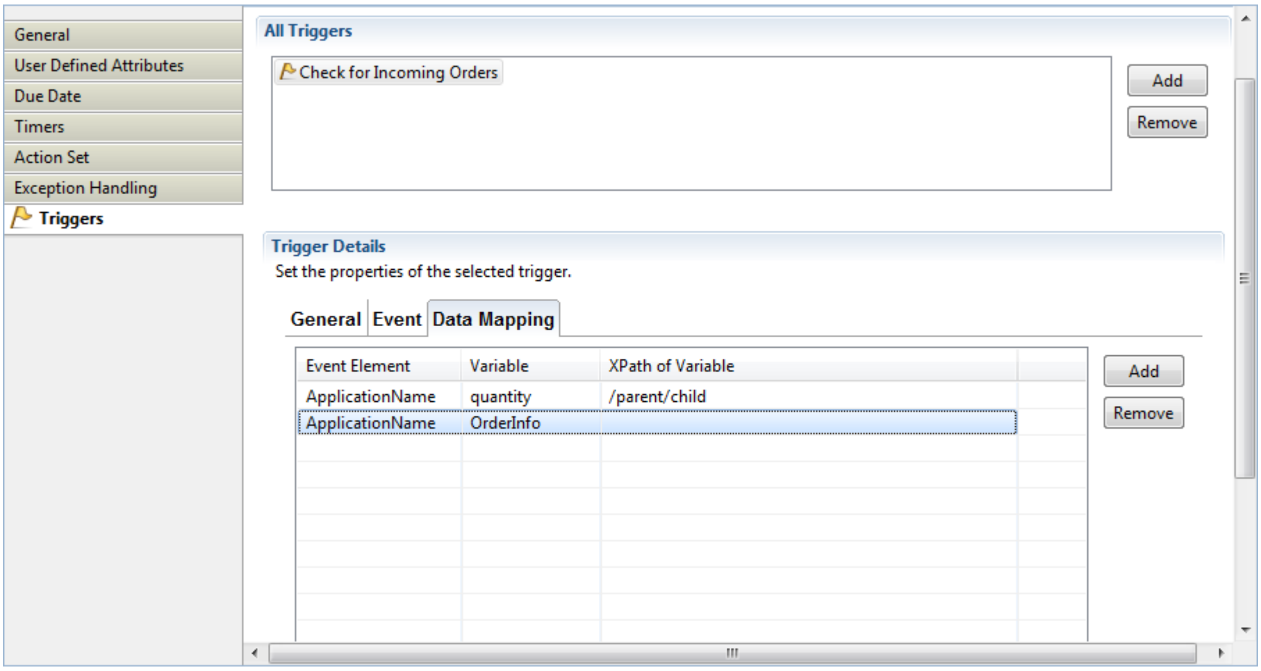
    c. Select the UDA from the **Variable** list to which you want to map the XML element.

Note

Triggers are flexible enough so that you can map external data of type String to a UDA of type INTEGER as long as the external data consists only of numerals.

    d. Click in the field in the **XPath of Variable** column.

Figure 11.41 Defining Data Mappings



Event Element	Variable	XPath of Variable
ApplicationName	quantity	/parent/child
ApplicationName	OrderInfo	

    e. Optional: If you want to edit the XPath expression that you selected in the **XPath of Variable** drop-down list, click the ellipsis (**...**) button that is displayed next to the **XPath of Variable** drop-down list. **XPath Editor** dialog is displayed with the selected XPath expression in its editor area. You can edit this XPath expression and click **OK** to use the edited XPath expression.

Note

The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note

The XPath expressions related to the selected XML UDA are displayed in the **XPath** drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected UDA, the XPath list that can be used with this Initial value is displayed. When both are defined, Xpath list of XML Schema is displayed.

Note

XPath Editor only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

f. If you want to remove a mapping, select it and click **Remove**.

12. If you want the trigger to fire only for certain process instances:

    a. Select the **Process Instance Selection** sub-tab.

    b. Click **Add** to add a new condition.

    c. From the **Variable** list, select the UDA that you want to evaluate. Specify an XPath expression for the element in the incoming XML file whose value you want to compare.
    When the trigger is fired, only those process instances are selected in which the value of the UDA matches the value of the corresponding element in the incoming XML file. If you specify multiple conditions, the trigger will only fire when all conditions are satisfied.

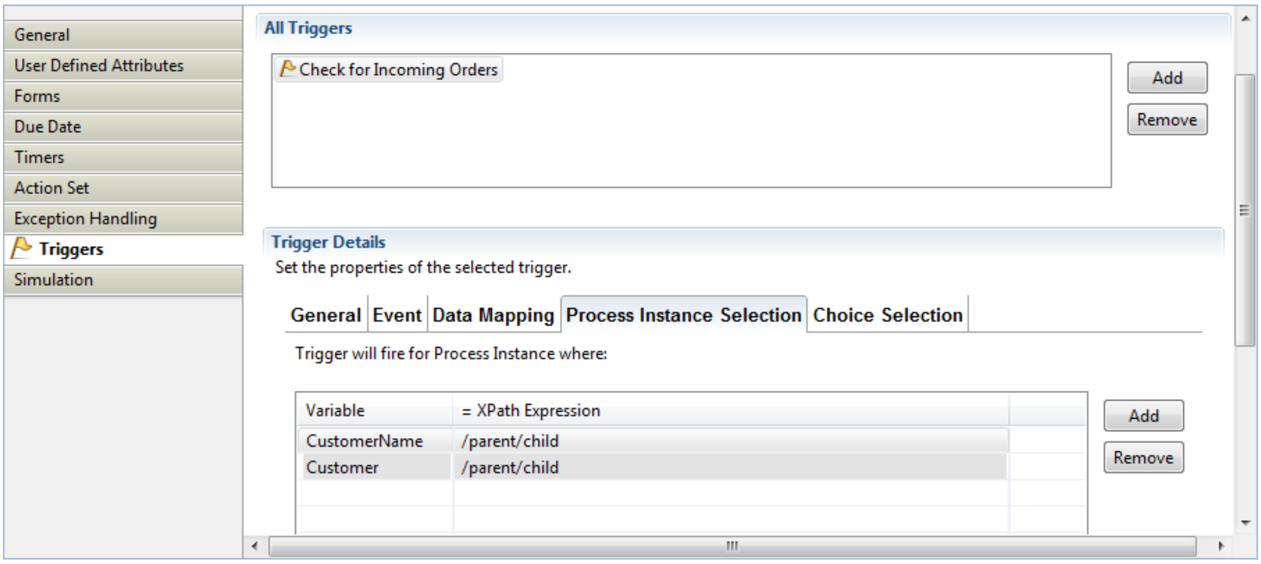
    Note

    Ensure that the UDA is marked as a Worklist UDA on the **User Defined Attributes** tab. Otherwise, the trigger does not work. UDAs of type XML are not displayed in the **Variable** list, as they cannot be Worklist UDAs.

    d. If you want to remove a condition, select it and click **Remove**.

Say you want to trigger process instances for orders from partners. A UDA Partner indicates if an incoming order is from a partner. The UDA is of type BOOLEAN and is marked as a Worklist UDA. In the incoming XML file, the corresponding information is stored in the fromPartners element. The following figure shows the condition that you would specify in this case:

Figure 11.42 Filtering on process instances



Variable	= XPath Expression
CustomerName	/parent/child
Customer	/parent/child

In this example, the trigger will fire if an incoming XML file contains the following data:

```
<order>
  ...
  <fromPartners>true</fromPartners>
  ...
</order>
```

The trigger will not fire if the following data come in:

```
<order>
  ...
  <fromPartners>false</fromPartners>
  ...
</order>
```

13. If you want a particular choice to be made only under certain conditions:

    a. Select the **Choice Selection** sub-tab.

    b. From the list, select the arrow for which you want to define a condition. Specify a JavaScript expression that defines the condition.

    c. You can rearrange the order in which the conditions are evaluated by highlighting a condition and clicking the **Up** or **Down** button.

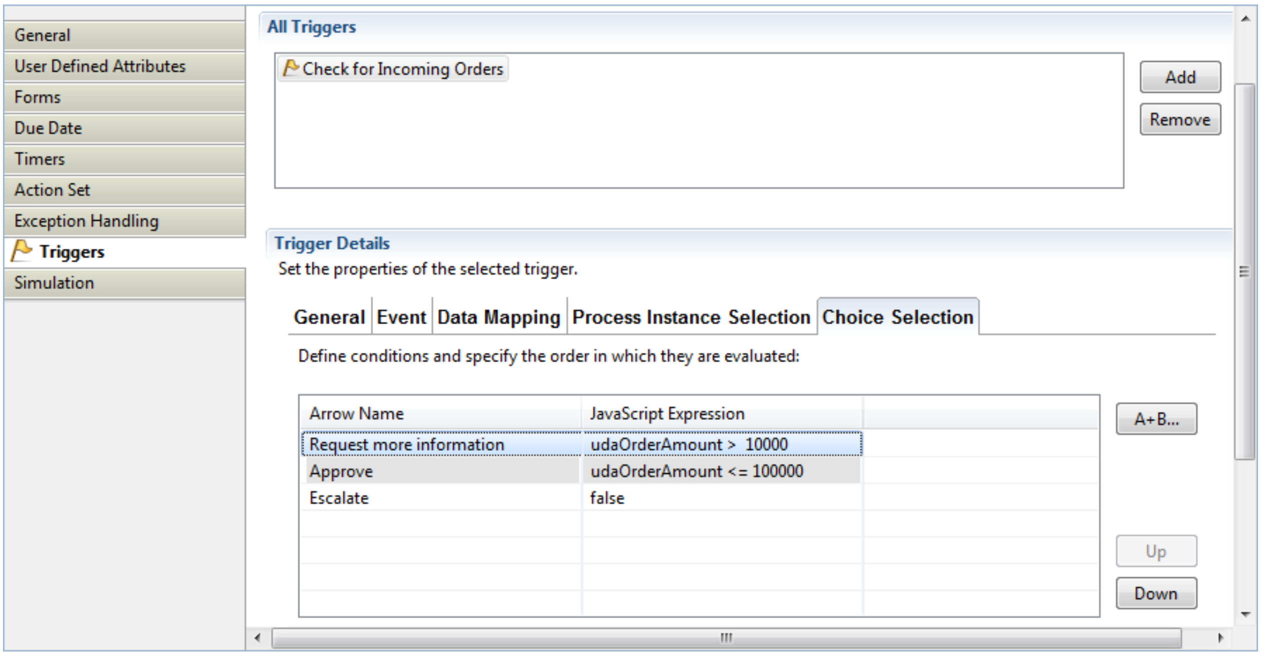
       All conditions are checked in the sequence in which they appear in this dialog. The first matching condition is used and the process flow proceeds along this arrow.

    d. Decide what is to happen if none of the conditions apply. From the **If no expression is true, then** list, you can either select an arrow or specify that the trigger is to wait for another event.

    If you select an arrow, the process flow proceeds along the arrow to the next node. If you specify that the trigger is to wait for another event, the Activity Node remains active.

    Say you wanted the vice president's approval of orders over $10,000. You would build a JavaScript expression like the following:

Figure 11.43 Defining Conditions for Arrows



All Triggers

Check for Incoming Orders

Trigger Details

Set the properties of the selected trigger.

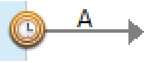
Define conditions and specify the order in which they are evaluated:

Arrow Name	JavaScript Expression
Request more information	udaOrderAmount > 10000
Approve	udaOrderAmount <= 100000
Escalate	false

14. Select the **General** sub-tab. Select the **Enable** check box to enable your trigger.

When a Trigger has been defined, the following symbol is added to this arrow connecting the Activity Node with another node:

Figure 11.44 Timer symbol



11.10 Defining Java Agents

A Java Agent is a special Java program that gathers information or performs some well-defined tasks in the background without your immediate presence and on some regular schedule. Java Agents are created to run automatically and act asynchronously on your behalf. You can use Java Agents to access external systems, such as legacy systems. Using Java Agents, you can incorporate these external services into your process instances.

Systemwalker Runbook Automation Studio allows you to add several Java Agents to your Workflow Application projects. For configuring new Java Agents, Systemwalker Runbook Automation Studio provides an XML file called agentsConfig.xml. Once a new Java Agent is configured, it can run as an activity in a process instance by assigning it to an Activity Node. A Java Agent is assigned to an Activity Node in the same manner that a Role is assigned to an Activity Node. The Java Agent then takes the place of the activity.

To define a new Java Agent:

1. [Right-click your project in the Navigator view and select **New** > **Agents** from the pop-up menu.

   A new Java Agent file (agentsConfig.xml) is automatically stored in the Resources folder of your Workflow Application project. The default agentsConfig.xml file automatically opens in the Resource editor.

2. Type in the properties that are required for the new Java Agent. Refer to "C.1 Java Agent File (agentsConfig.xml)" for information on the Java Agent file (agentsConfig.xml).

3. Click **Save** on the toolbar to save the new agent.

   You can use the new Java Agent to carry out specific tasks in the background.

11.11 Defining FTP Agents

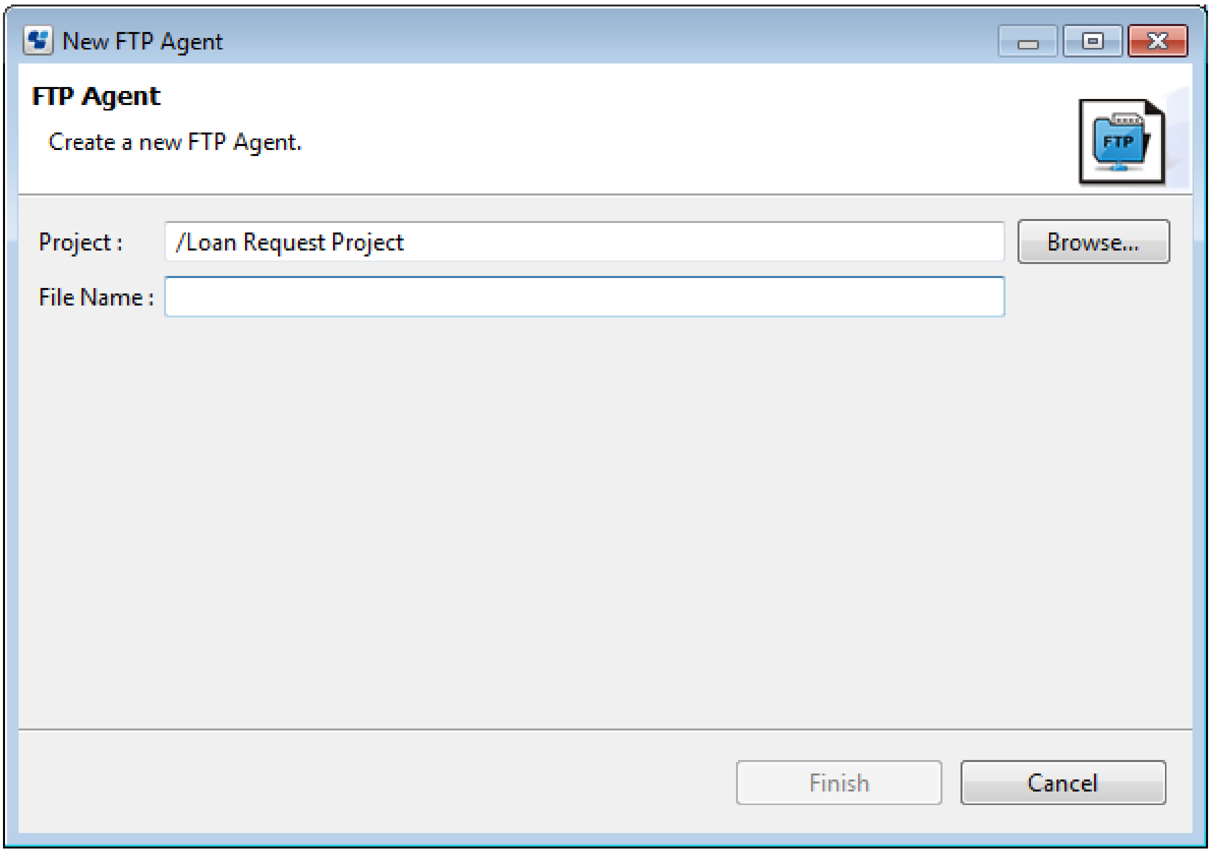
FTP Agents automatically transfer files attached to process instances that contain the FTP Agents. Like all Agents, FTP Agents take the place of Systemwalker Runbook Automation activities. Process instances started from a process definition containing an FTP Agent automatically transfer the files that are attached to them. They transfer the attached file(s) to the FTP Server of any machine to which the Management Server machine can connect.

To define a new FTP Agent:

1. Right-click your project in the **Navigator** view and select **New >> FTP Agent**.

   The **New FTP Agent** dialog is displayed. The following figure displays the **New FTP Agent** dialog.

   Figure 11.45 Defining FTP Agent



New FTP Agent

FTP Agent

Create a new FTP Agent.

Project : /Loan Request Project

File Name :

2. Enter the name of the project for which you want to define a new FTP Agent in **Project** field.

3. Optional: Click **Browse** to select the project.

   Folder Selection dialog is displayed. Select the project from this dialog and click **OK**.

4. After selecting the project, enter a name for the new FTP Agent in **File Name** field.

Note

You must add **.xml** after the name of the file that you enter in the **File Name** field.

5. Click **OK** on the **New FTP Agent** dialog.

   The new FTP Agent file (FTPAgent.xml) is added to the **Resources** folder and also displayed in the editor. Refer to "C.2 FTP Agent File (ftp.xml)" for more information on FTP Agent file.

11.12 Defining HTTP Agents

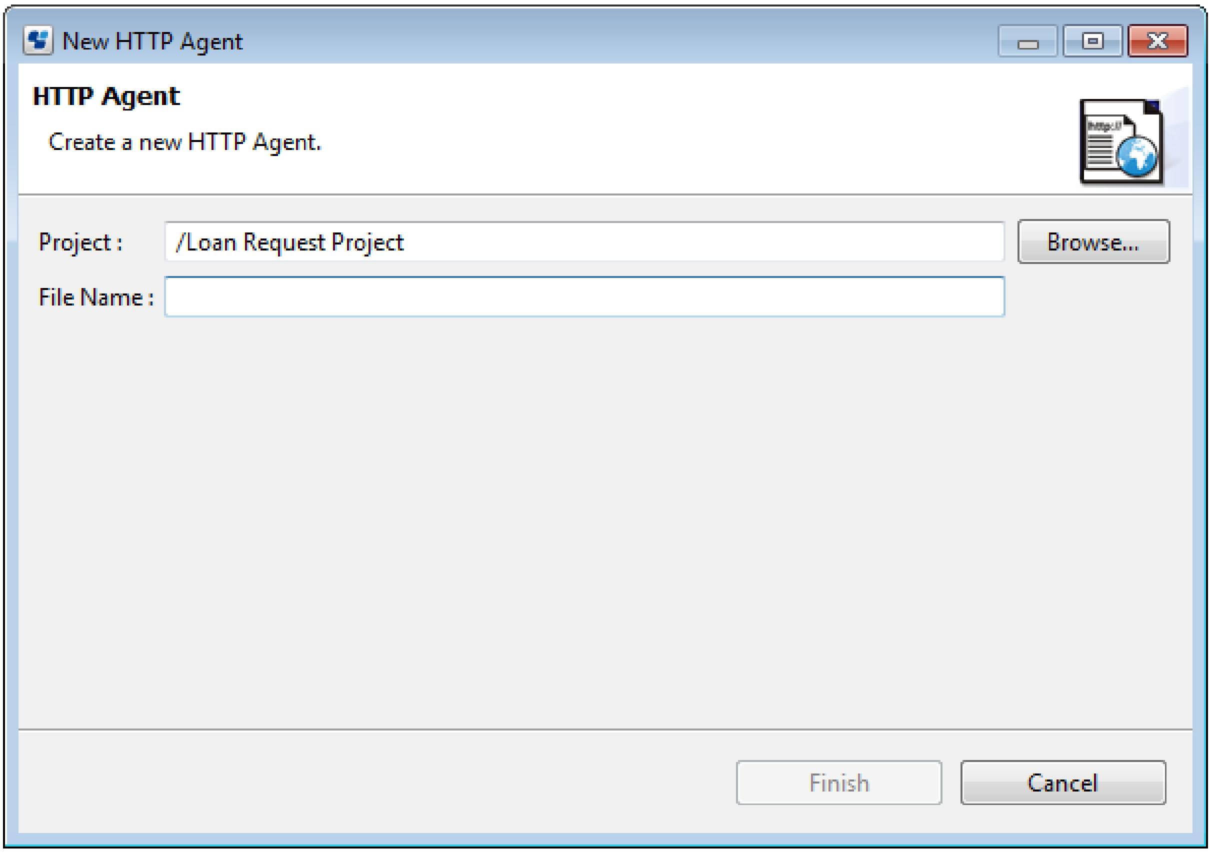
HTTP Agents are used to send data to external locations on the Internet and receive data from the Internet. An Agent sends data to the URL specified in its configuration and receives response data from that URL.

To define a new HTTP Agent:

1. Right-click your project in the **Navigator** view and select **New >> HTTP Agent**.

   The **New HTTP Agent** dialog is displayed. The following figure displays the **New HTTP Agent** dialog.

Figure 11.46 Defining HTTP Agent



New HTTP Agent

HTTP Agent

Create a new HTTP Agent.

Project : /Loan Request Project

File Name :

2. Enter the name of the project for which you want to define a new HTTP Agent in **Project** field.
3. Optional: Click **Browse** to select the project.
   Folder Selection dialog is displayed. Select the project from this dialog and click **OK**.
4. After selecting the project, enter a name for the new HTTP Agent in **File Name** field.

Note

You must add **.xml** after the name of the file that you enter in the **File Name** field.

5. Click **OK** on the **New HTTP Agent** dialog.

   The new HTTP Agent file is added to the **Resources** folder and also displayed in the editor. Refer to "C.3 HTTP Agent File (HTTPagent.xml)" for more information on HTTP Agent file.

11.13 Defining File Listener

File Listeners monitor files in specified directory locations. When they detect new or newly modified files, they notify the file handlers, so they can perform automatic functions like starting Systemwalker Runbook Automation process instances or making choices on activities. File listeners are typically used for integrating Systemwalker Runbook Automation with other enterprise applications.

To define a new File Listener:

1. Right-click your project in the **Navigator** view and select **New >> File Listener**.

   The fileListenerConf.xml file is created and displayed in the editor.

Note

You can create only one File Listener file per project.

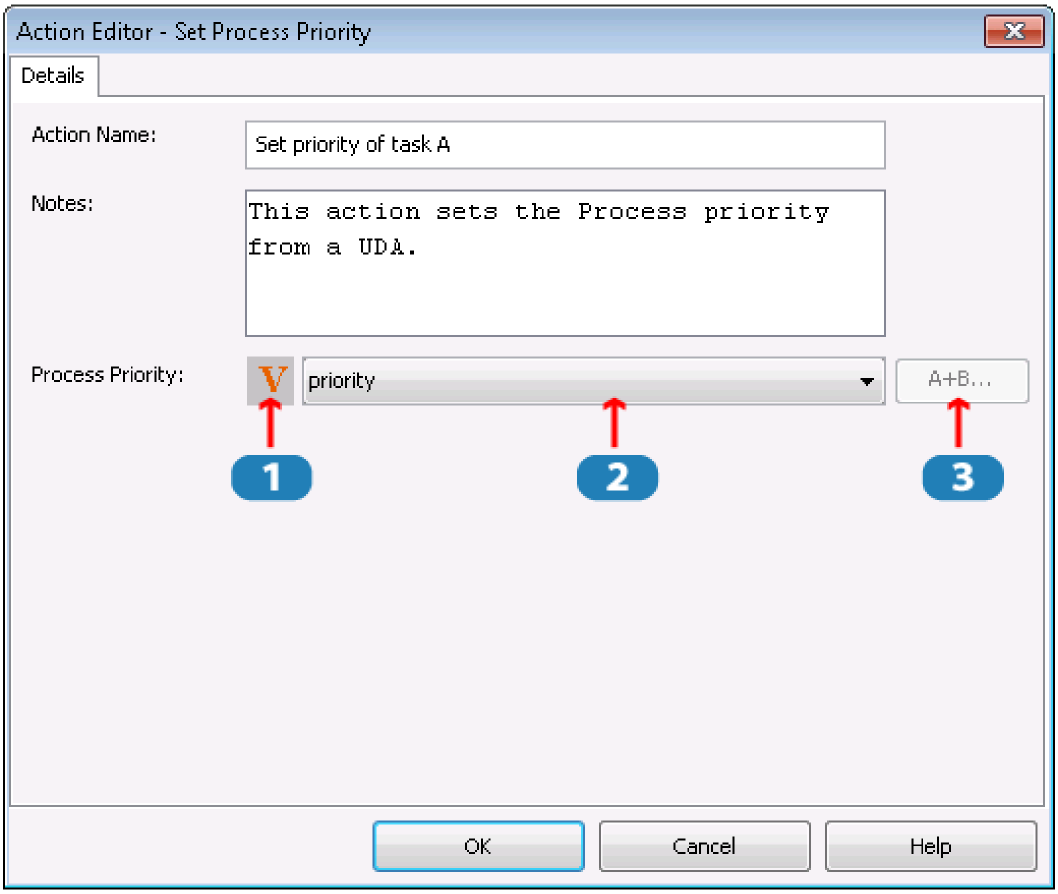
11.14 Defining JavaScript Expressions

In Systemwalker Runbook Automation Studio, JavaScript expressions are used with the following elements:

- Complex Conditional Nodes
- Java Actions
- Triggers
- Sequential Loop Nodes

Several software controls are provided to help you build JavaScript expressions more effectively and easily. The following figure shows an example:

Figure 11.47 Defining JavaScript Expressions



Action Editor - Set Process Priority

Action Name: Set priority of task A

Notes: This action sets the Process priority from a UDA.

Process Priority: V priority

Key:

- 1 Expression Mode Button
- 2 Expression Field
- 3 Expression Builder Button

Expression Mode Button

The Expression Mode Button allows you to switch between different expression modes. The character on the button shows the mode that is currently effective. The modes are:

Mode	Button	Description
Constant Mode	C	Used for simple constant (literal) JavaScript expressions. Examples of constants are 50 or AskVicePresident.
Variable Mode	V	Used for JavaScript expressions that consist of a User Defined Attribute (UDA) or Application Variables or both. When you switch to this mode, a list is displayed that allows you to select the UDA or the Application Variables.
Expression Mode	E	Used for complex JavaScript expressions. A complex JavaScript expression might be composed of UDAs, operators, JavaScript functions and constants. The following is an example: "Process started" + uda.get("Description") With Expression Mode, you can type the JavaScript expression directly or build it using the Expression Builder.

Expression Builder Button

The Expression Builder Button opens a dialog that helps you build complex JavaScript expression.

Expression Field

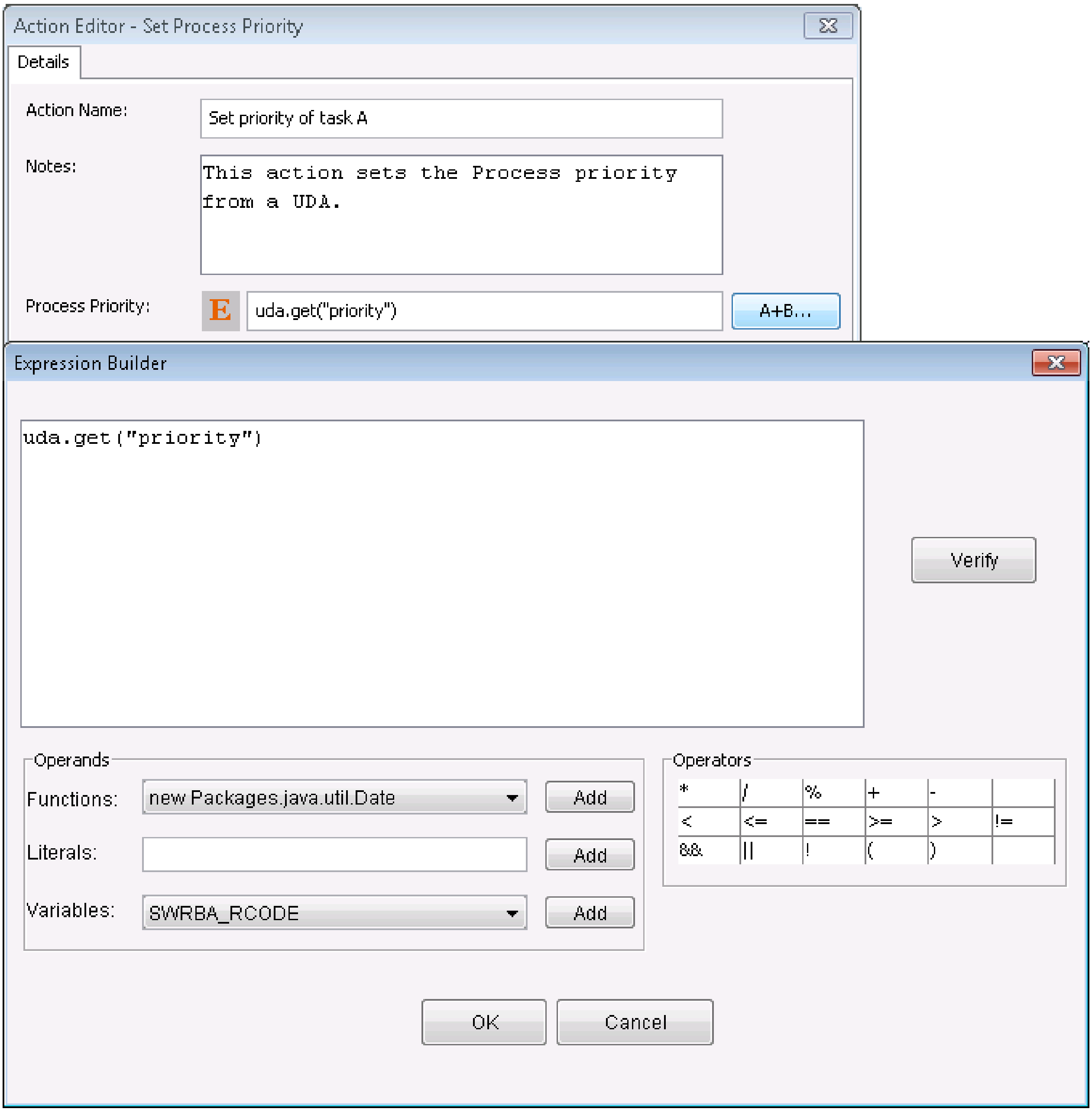
The Expression Field displays the JavaScript expression that you specify.

Expression Builder

The Expression Builder helps you create more effective and viable JavaScript expressions in less time and with less chance for making time-consuming hard-to-find errors. You can select expression building blocks from lists and add operators by clicking them. This is an easier and faster method of building JavaScript expressions, especially if you are unfamiliar with JavaScript syntax. Also, the Expression Builder has a built-in verification function that checks the validity of JavaScript expressions you have built before they are used.

The following figure shows the Expression Builder.

Figure 11.48 Expression Builder



Action Name: Set priority of task A

Notes: This action sets the Process priority from a UDA.

Process Priority: uda.get("priority")

```
uda.get("priority")
```

Functions: new Packages.java.util.Date

Literals:

Variables: SWRBA_RCODE

The Expression Builder consists of the following parts:

- Expression Display

  This is where your JavaScript expression displays as it is built.

- **Verify** Button

  You can verify the validity of your JavaScript expression as you build it. Verification also occurs when you click **OK**. Invalid JavaScript expressions cause an error message that provides diagnostic information.

- **Operands** area

  In this area, you can select a JavaScript function or a UDA from the drop-down lists or type in a constant. Clicking **Add** adds it to your JavaScript expression.

- **Operators** area

  In this area, you can click an operator to add it to your JavaScript expression. The standard JavaScript operators are available.

11.14.1 Example: Building a JavaScript Expression Using the Expression Builder

This example uses the Expression Builder to build a JavaScript expression that defines a process instance description. The process instance description formed by this example tells the purpose of the process instance.

To build a sample JavaScript expression:

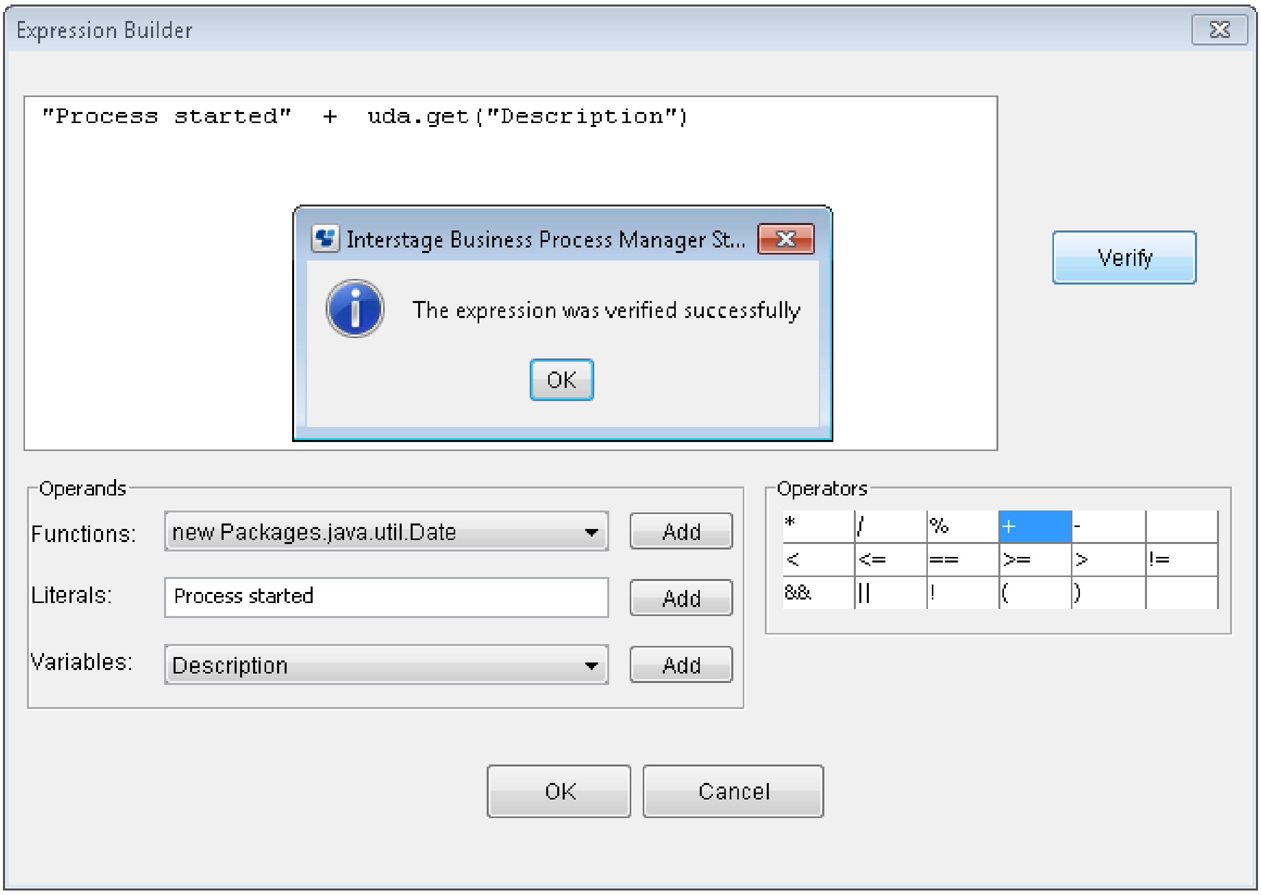
1. Create a process definition.
2. Add a User Defined Attribute (UDA) of type STRING called Description.

   For details, refer to section 6.18 Specifying User Defined Attributes.
3. Click the empty space in the Process Definition editor to display the **Properties** view for the process definition.
4. Select the **Action Set** tab. Add the Java Action Set Process Instance Description as an Init Action:
   a. Click **Add** in the **Init Action** area.
   b. Expand **Server Actions** and double click **Set Process Instance Description**.
5. In the **Action Editor - Set Process Description** dialog, click the Expression Mode Button until an **E** is displayed on it. Then click **A + B** to open the Expression Builder.
6. Type Process started in the **Literals** field. Click **Add** to add this constant to the JavaScript expression.
7. Click the + operator to add it to the JavaScript expression.

8. Select Description from the **Variables** list. Click **Add**.

   The JavaScript expression should now look like this:

Figure 11.49 Building a Sample JavaScript Expression



```
"Process started" + uda.get("Description")
```

The expression was verified successfully

Functions: new Packages.java.util.Date

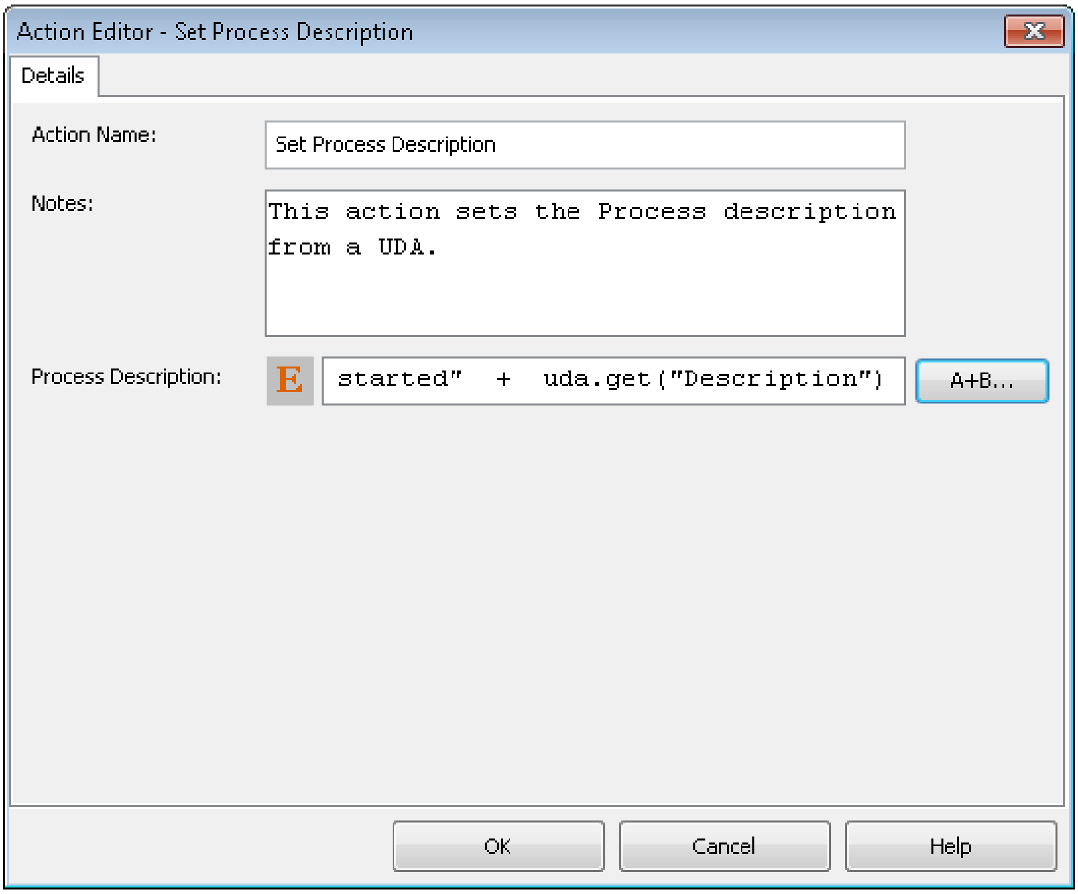
Literals: Process started

Variables: Description

9. Click **OK** to verify the JavaScript expression.

   If it is valid, the Expression Builder is closed and the JavaScript expression is displayed in the **Action Editor - Set Process Description** dialog .

Figure 11.50 Expression Field with Sample



Action Name: Set Process Description

Notes: This action sets the Process description from a UDA.

Process Description: started" + uda.get("Description")

Chapter 12 Simulating Processes

This chapter explains how to simulate the execution of processes.

12.1 Introduction to Process Simulation

Systemwalker Runbook Automation Studio allows you to simulate the execution of processes. This feature can help you in calculating cost and time for specific activities, and thus optimize your business processes.

Simulation Limitations

- You can simulate the execution of the process definitions containing:
    - Activity Nodes
    - Subprocess Nodes
    - Chained-Process Nodes
    - Voting Activity Node
    - Conditional Node
    - Complex Conditional Node
    - Compound Activity
- The simulation scenario must not generate more than 15,000 processes to be simulated. If it does, an error will be thrown and process simulation will fail.
- The simulation of the processing of Java Actions and User Defined Attributes is not supported.
- If the process definition contains two or more nodes with the same name, simulation results may not be correct.

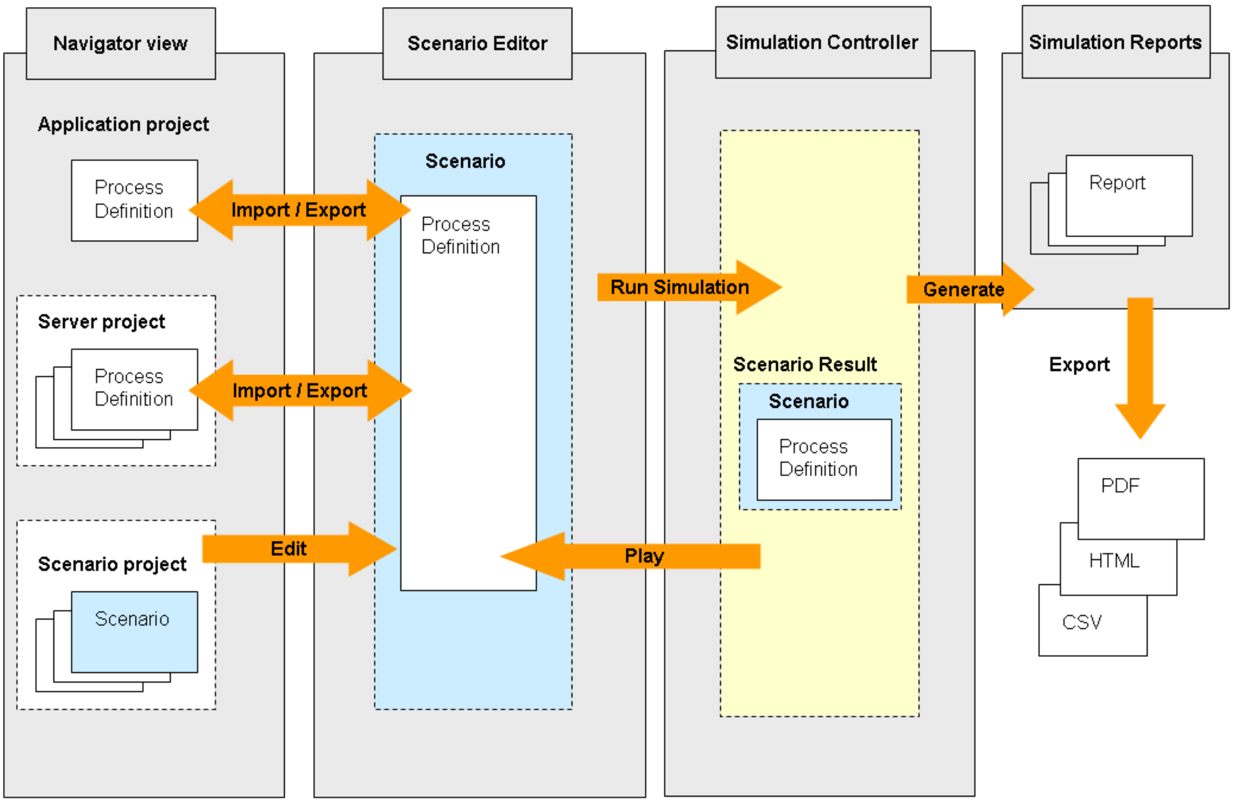
12.1.1 Overview of the Simulation Process

The simulation process consists of various major steps:

1. In the **Navigator** view, when you open a new workspace, a simulation scenario folder called **Simulation** is created, and also when you create a Workflow Application project, a **Simulation** folder is created in the project structure. This folder is initially empty. Here you start with the creation of a new scenario project.
2. In the **Scenario editor**, you define simulation properties for a scenario by editing the scenario. Every scenario must import a process definition from either a Workflow Application or a server project. The process definition used by a scenario can, in turn, be exported to a local or server project. In addition, for every activity defined in the process definition, simulation properties can be specified.
3. The scenario is calculated for preparing the simulation result by calling the **Run Simulation** command. The simulation results are automatically saved to the Scenario (.ssr) file and stored in the Simulation folder of your Workflow Application project. In addition, the Simulation Controller is displayed below the Scenario editor.
4. From the **Simulation Controller**, the execution of the process simulation is started, the imported process definition is displayed in the Scenario editor and shows the progress of simulation, and simulation reports can be generated for analysis purposes.

The figure below shows this process:

Figure 12.1 Simulation Process Overview



12.1.2 General Procedure

This section gives an overview of the steps required to set up a simulation scenario and locally simulate the execution of a process.

1. As a prerequisite for simulating a process, you need to define a **process definition** based on which process instances will be triggered and executed. For the following types of nodes in your process definition, you can specify "Simulation Properties" that are taken into account when the simulation is executed:

   - Activity Node
   - Voting Activity Node
   - Conditional Node
   - Complex Conditional Node

   Note

   You cannot define simulation properties for Subprocess Nodes and Chained-Process Nodes.

   For example, you can specify the probability with which a specific arrow is chosen in case you have a node with several outgoing arrows, or you can specify detailed information on the resources required to perform a specific activity. Refer to section 12.2.4 Defining Simulation Properties for Nodes for details.

   Note

   The **process definition** included Email node can execute simulation, but the procedure executed in these node is exempt from evaluation. For example, e-mail is not sent to the mail address specified in Email node.

2. You create a new simulation scenario. Refer to section 12.2.1 Creating a Simulation Scenario for details.

3. For the simulation scenario, you need to specify, among other things, the start and end date for the simulation, the currency in which costs are to be calculated, the process definition whose execution is to be simulated, the interval in which new process instances are created. In addition, for every role defined in the process definition, you can specify an individual business calendar to be applied and costs per defined unit (e.g. cost/hour). Refer to section 12.2.3 Defining Simulation Properties for a Scenario for details.

   Note

   You can also simulate execution of process definitions containing Subprocess Nodes and Chained-Process Nodes. If the process definition has Subprocess Nodes or Chained-Process Nodes, execution of the parent process definition is simulated and then execution of the subprocess definitions is simulated automatically.

4. Once finished with the definition of the simulation properties, you simulate the scenario and generate a simulation result. Refer to section 12.3.1 Preparing a Simulation Result for details.

5. Based on the prepared simulation results, you can replay the simulation with the simulation results. While the simulation is running, you can observe the processing of every node in your process definition. Refer to section 12.3.2 Using the Simulation Controller for details.

6. The simulation results can then be evaluated using the extensive simulation report functionality. Refer to section 12.3.3 Generating Simulation Reports for details.

12.2 Defining a Simulation Scenario

This section describes how to create a scenario and define simulation properties.

12.2.1 Creating a Simulation Scenario

This section explains the procedure for creating a new scenario.

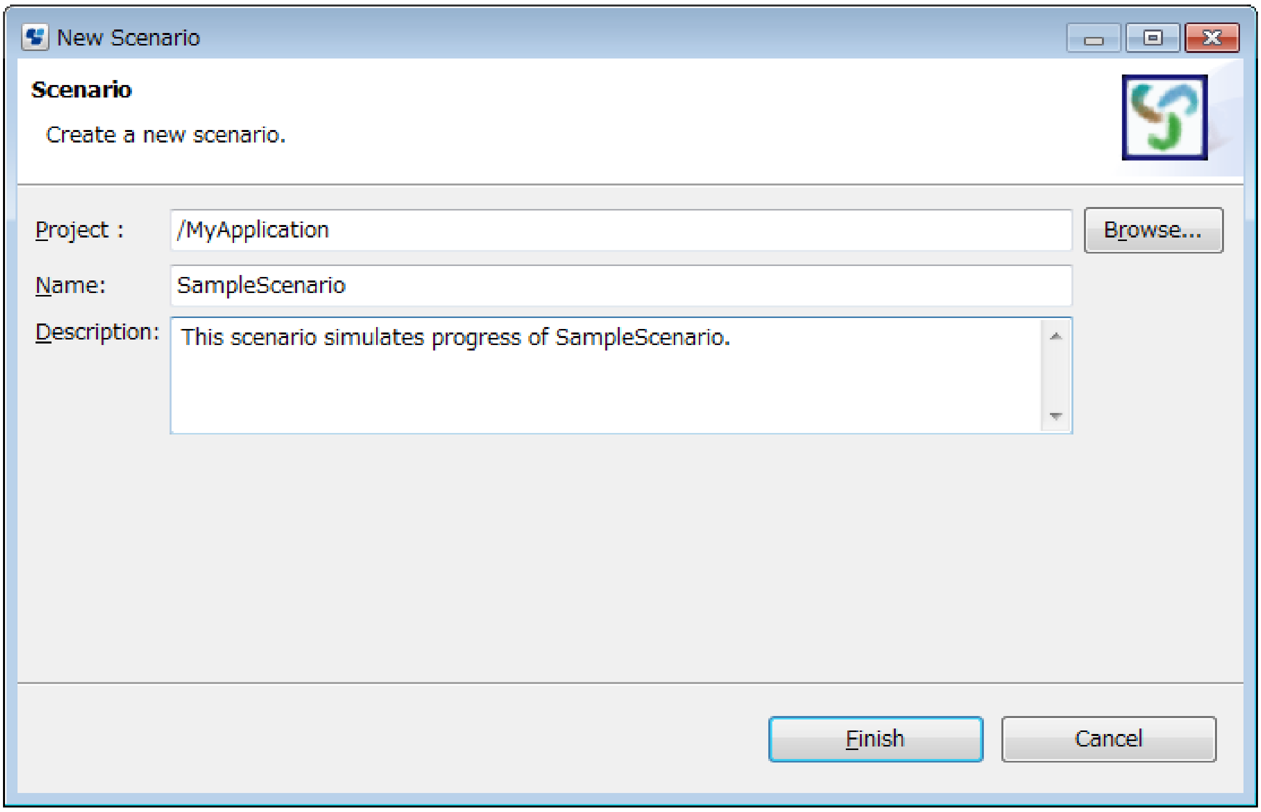
Prerequisite:

- You have created process definitions for which you want to simulate the execution of processes.

To create a new scenario:

1. Right-click the empty **Simulation** folder in the Navigator view and select **New** > **Scenario**. The **New Scenario** dialog box is opened:

   Figure 12.2 Creating a Scenario

   Scenario

   Create a new scenario.

   Project : /MyApplication

   Name: SampleScenario

   Description: This scenario simulates progress of SampleScenario.

2. In the **Project** field, the current Workflow Application project or the Simulation Scenarios project is displayed. You can change to another project by clicking **Browse** and then selecting a different project. You can create a simulation scenario for a process definition of a Workflow Application project or a server project.

3. Type a name for your scenario in the **Name** field, and a brief description of the scenario in the **Description** field.

4. Click **Finish**.

   The new scenario is automatically stored in the **Simulation** folder in the Navigator view. The Scenario editor is opened and the name and description you entered are automatically added.

   Apart from creating a new simulation scenario, you can also open existing scenarios. To do so, open the Simulation folder where you have stored the new scenario, and double-click the scenario (.ssr) file.

5. Continue with defining the simulation scenario using the Scenario editor. Refer to section 12.2.3 Defining Simulation Properties for a Scenario for details.

12.2.2 Using Simulation Values From History

This section explains the procedure for performing a process simulation using historical values.

Prerequisite:

- You have created a simulation scenario for which you want to use historical values.

To use values from history for simulating processes:

1. Open a simulation scenario in the Scenario editor.

2. In the **Simulation Period And Arrival Setting** area, click the Gather Default Values from History link.

   The **Gather Default Simulation Values from History** dialog is displayed. Here you can specify the server from which you want to retrieve the simulation values.

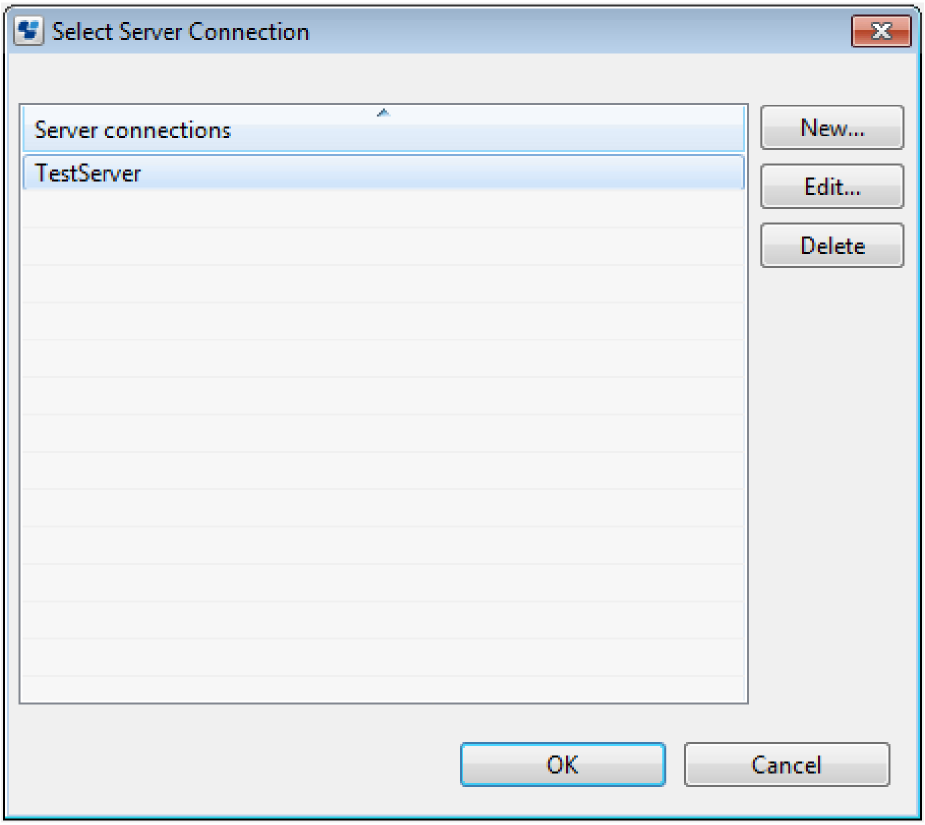
3. Specify the source of the historical simulation values.

   You can select an available server connection or browse for a new server connection:

   - **Select a server connection**: From the **Server Connection** drop-down list, select the remote server from which you want to fetch the historical simulation values.
   - Browse for a new server:

   1. Click **Browse Connection** to add a new server connection. **Select Server Connection** dialog is displayed.
   2. If you want to create a new server connection, click **New** on **Select Server Connection** dialog. 2.4.1 Server Connection Settings dialog is displayed.
   3. Enter all the parameters in **Server Connection Setting** dialog to create a new server connection and click **OK**. The new server connection is displayed in **Select Server Connection** dialog. Refer *Setting Your Preferences* to know more about various parameters.
   4. Click OK on Select Server Connection dialog.

   Figure 12.3 Selecting a server connection



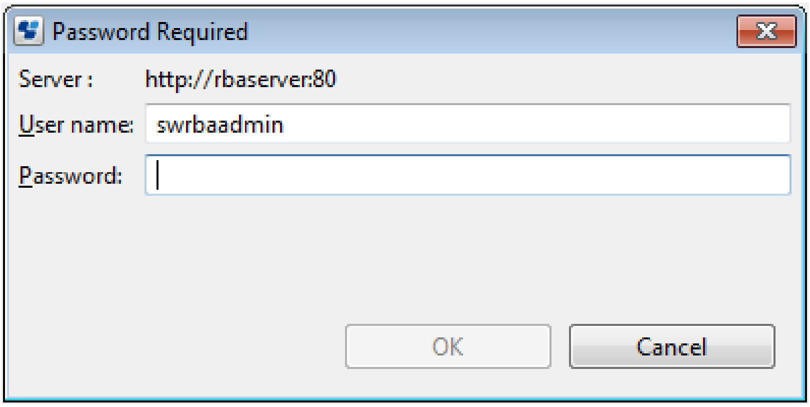
4. In the Gather Default Simulation Values from History dialog, click Get list.

   The **Password Required** dialog is displayed.

5. In the **Password Required** dialog box, type in your password to authenticate against the server.

   The user name is automatically displayed. Click **OK** to confirm your user name and password.

Figure 12.4 Entering your password



Password Required

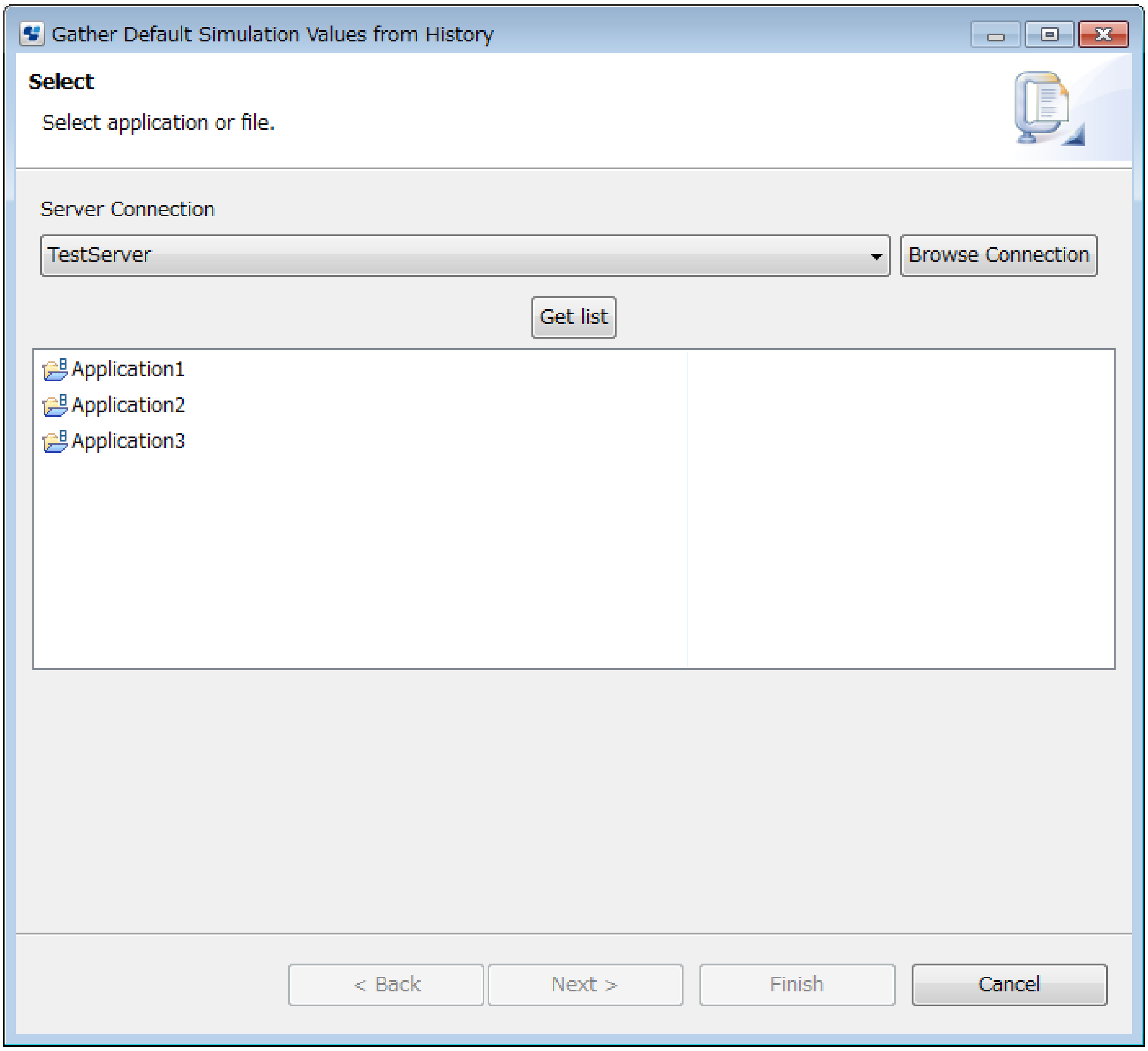
Server : http://rbaserver:80

User name: swrbaadmin

Password:

6. In the **Select** dialog that displays all applications stored on the selected server (remote_server, in the example), in the Application area select an application, and click **Next**.

Figure 12.5 Displaying applications on the selected server



Gather Default Simulation Values from History

Select

Select application or file.

Server Connection

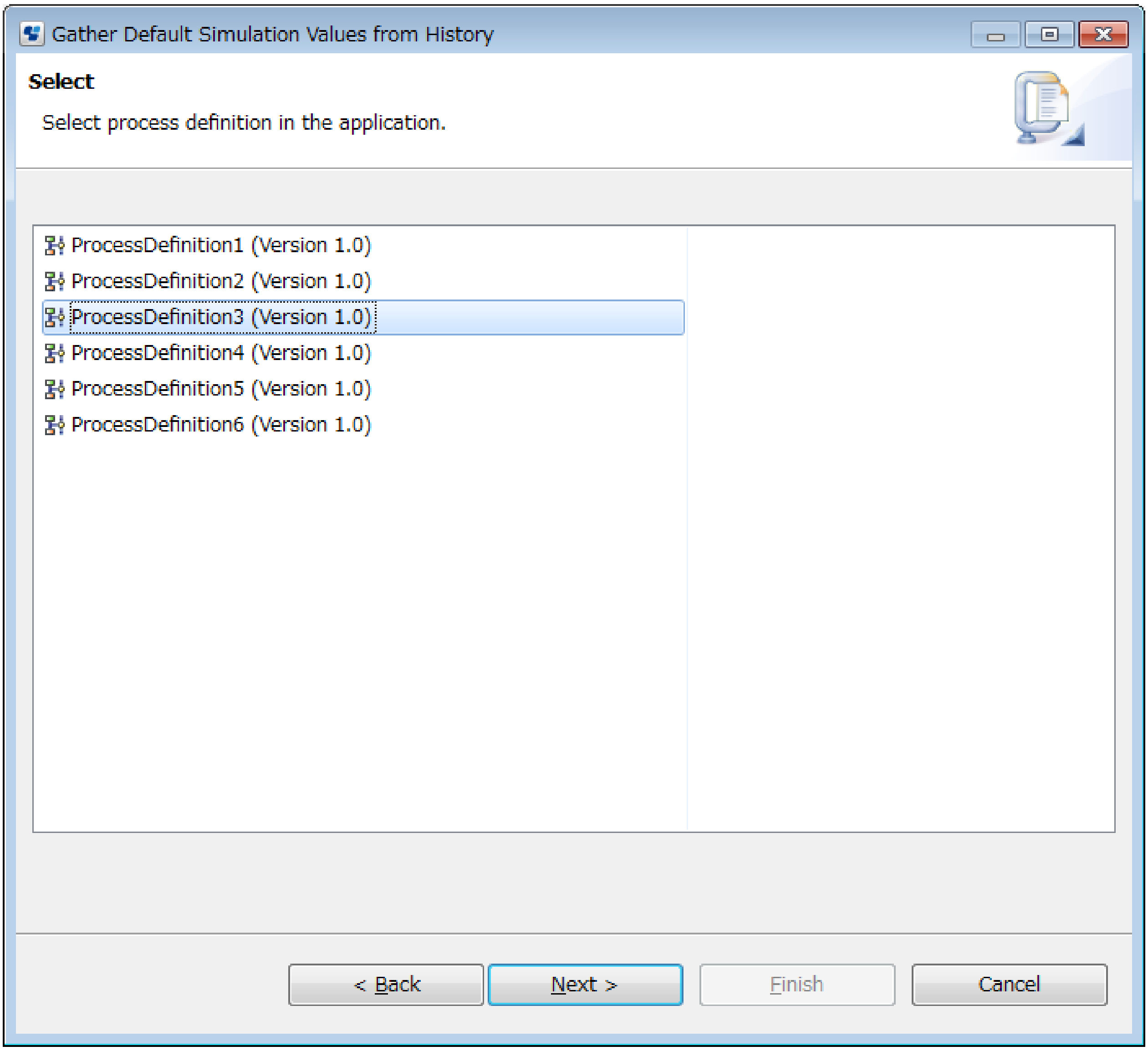
TestServer

Application1
Application2
Application3

The next dialog shows the history events pertaining to the selected application.

7. In the Process Definition area, select a process definition and click **Next**.

Figure 12.6 Displaying process definitions



Gather Default Simulation Values from History

Select

Select process definition in the application.

- ProcessDefinition1 (Version 1.0)
- ProcessDefinition2 (Version 1.0)
- ProcessDefinition3 (Version 1.0)
- ProcessDefinition4 (Version 1.0)
- ProcessDefinition5 (Version 1.0)
- ProcessDefinition6 (Version 1.0)

Note

This dialog appears only if your simulation scenario contains a single process definition. If your scenario contains more than one process definition, this dialog is skipped; all history events related to the selected application are retrieved.

Note

You can update the list of process definitions using the pop-up menu or pressing the F5 key on your keyboard.

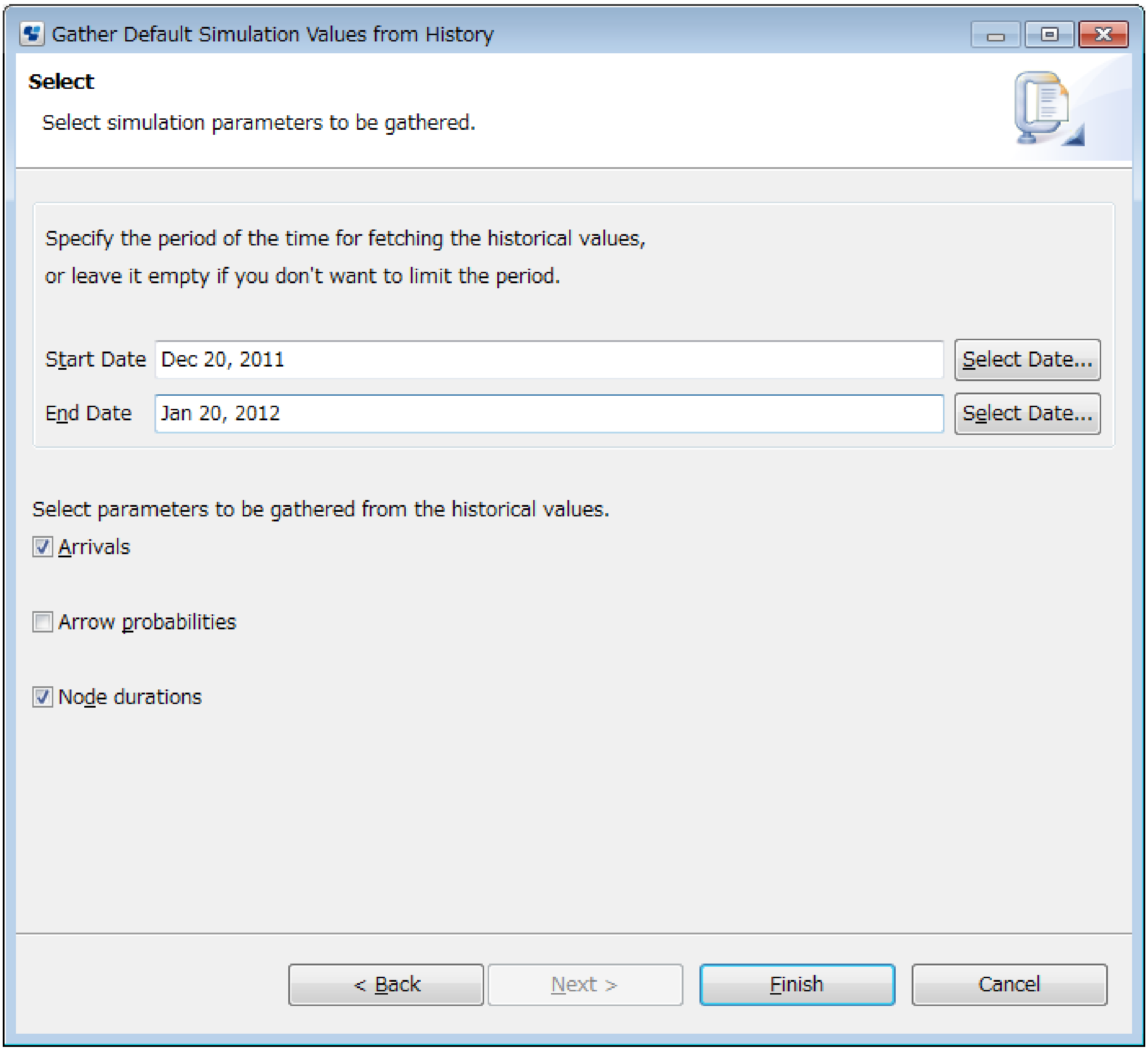
8. In the dialog that appears, specify when the simulation values are to be gathered.

   You can specify the following parameters:

   - **Start Date**: Refers to the date when Systemwalker Runbook Automation Studio starts getting historical simulation values.
   - **End Date**: Refers to the date until which Systemwalker Runbook Automation Studio fetches historical simulation values.

- **Arrivals**: Specifies that process instances will be triggered at the average interval you specified as arrival interval.
- **Arrow probabilities**: Refers to the percentage of the time a particular arrow is taken.
- **Node durations**: Refers to the calculation of average activity duration.

Figure 12.7 Specifying simulation dates



Gather Default Simulation Values from History

Select

Select simulation parameters to be gathered.

Specify the period of the time for fetching the historical values, or leave it empty if you don't want to limit the period.

Start Date: Dec 20, 2011 [Select Date...]

End Date: Jan 20, 2012 [Select Date...]

Select parameters to be gathered from the historical values.

☑ Arrivals

□ Arrow probabilities

☑ Node durations

Note

If you have selected a process definition having Subprocess Nodes or Chained-Process Nodes for simulating its execution, the **Arrivals** checkbox will be disabled.

Note

If you do not select any dates, the time period for fetching historical values will be unlimited.

9. Specify simulation procedure.

   You can specify the following parameter.

   - Start Date: The date when Systemwalker Runbook Automation Studio starts to get historical simulation value.

- End Date: Systemwalker Runbook Automation Studio has got historical simulation value until this date.
- Arrivals: To be triggered process instance by average interval as specified arrivals.
- Transition of arrows: The probability(%) transited with specific arrows
- Work time for node: calculated value for average activity period.

12.2.3 Defining Simulation Properties for a Scenario

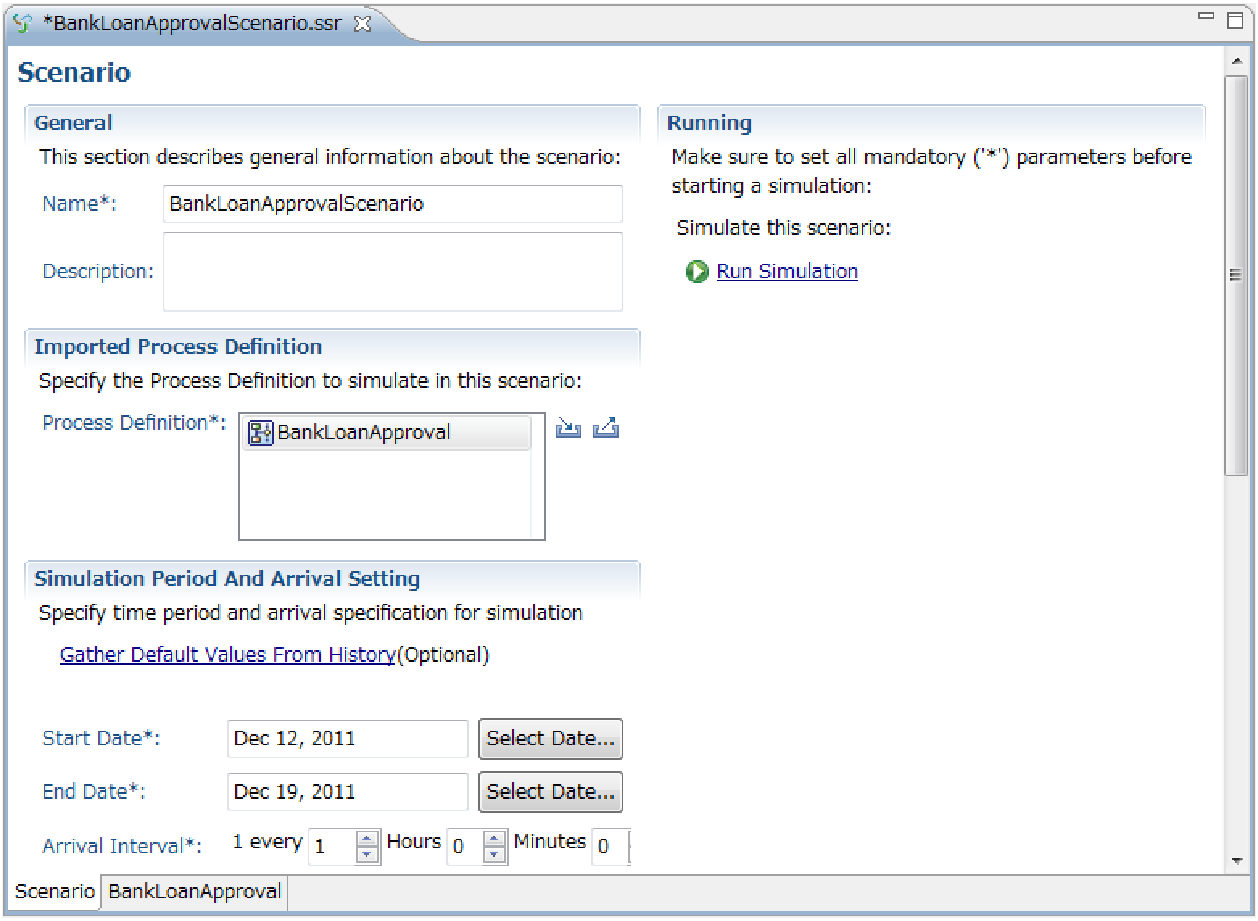
This section explains the procedure for defining simulation properties for scenarios.

Prerequisite:

- You have created a scenario in the **Simulation Scenarios** project or a Workflow Application project. Refer to 12.2.1 Creating a Simulation Scenario for details.

The Scenario editor is either opened automatically after clicking **Finish** in the **New Scenario** dialog or by double-clicking a scenario name in the Navigator view.

Figure 12.8 The Scenario Editor



*BankLoanApprovalScenario.ssr

Scenario

General

This section describes general information about the scenario:

Name*: BankLoanApprovalScenario

Description:

Imported Process Definition

Specify the Process Definition to simulate in this scenario:

Process Definition*: BankLoanApproval

Simulation Period And Arrival Setting

Specify time period and arrival specification for simulation

Gather Default Values From History(Optional)

Start Date*: Dec 12, 2011 Select Date...

End Date*: Dec 19, 2011 Select Date...

Arrival Interval*: 1 every 1 Hours 0 Minutes 0

Running

Make sure to set all mandatory ('*') parameters before starting a simulation:

Simulate this scenario:

Run Simulation

Scenario | BankLoanApproval

To define the simulation properties for a scenario:

1. Open the Scenario editor as described above.
2. In the **General** area, define the following information for the scenario:
    1. **Name**: Name of the scenario. By default, this is the name you entered when creating the scenario.
    2. 2. **Description**: Additional information about the scenario. For example, enter text describing the purpose of the simulation. By default, this is the description you entered when creating the scenario.

3. In the **Imported Process Definition** area, perform the following actions:

    1. **Process Definition**: To import the process definition that is to be used for this scenario, click the **Import** button (). The **Import Process Definition** dialog is opened and lists all available projects with their process definition(s). Select the respective one from the list and click **OK**.

        Note

        - The process definition that you select in **Import Process Definition** dialog is treated as the primary process definition for the simulation.

        - If there are two or more process definitions which have the same name, the process definition selection dialog will be displayed so that you can select a process definition which will be included in the process definition list to be simulated.

        - If you select a process definition having Subprocess Nodes or Chained-Process Nodes for simulating its execution, the list of all its subprocess definitions or chained-process definitions is automatically imported, and then displayed in the **Process Definition** field. These subprocess or chained-process definitions are retrieved from the application project to which the scenario belongs.

        - A process definition with subprocess or chained-process definitions cannot be simulated in the **Simulation Scenarios** project.

        You can also export a process definition used by the current scenario by clicking the Export button (). This will open the **Export Process Definition** dialog and you can select a project in which the process definition is to be stored.

        Note

        When storing a process definition in a server project, its application ID is set to 'System'.

    2. When the importing of the process definition is finished, it is opened in a separate page that becomes active. You can then change or define the simulation properties for the individual nodes contained in the process definition. Proceed as described in section 12.2.4 Defining Simulation Properties for Nodes.

        Note

        When you import a parent process definition that contains subprocesses, the parent process definition is displayed at the top in **Process Definition** field. The subprocess are automatically displayed below the parent process definition. For each subprocess definition, a separate Process Definition tab is opened in the Scenario Editor. You can then change or define the simulation properties for the individual nodes contained in the subprocess definitions. The process definition list is updated if you edit process definitions and add or delete or change the subprocesses.

    3. To return to the scenario and continue with the definition of the simulation properties for the scenario, click the tab labeled with the name of the scenario at the bottom of the Scenario Editor.

4. In the **Simulation Period and Arrival Setting** area, you can perform the following actions:

    - **Start Date**: Date when the simulation is to start. The date format depends on the locale installed on your machine. You may click the **Select Date** button to choose a date from the popup calendar.

    - **End Date**: Date when the simulation is to finish. The date format depends on the locale installed on your machine. You may click the **Select Date** button to choose a date from the popup calendar.

    - **Gather Default Values from History**: Click this link to gather default values from history in order to run a simulation using these values. For more information, see the chapter 12.2.2 Using Simulation Values From History.

    - **Arrival Interval**: Arrivals define how often a process instance will be triggered in the course of the simulation. Define this flow of process instances by specifying the interval as a specific period in time.

    - **Arrival Type**: You can choose between **Regular** and **Random** arrivals. Regular arrivals specify that process instances will be triggered always at exactly the interval you specified as arrival interval. Random arrivals specify that the process instances will be triggered some time in the interval you specified. For example, if six process instances per hour are to be triggered, the amount of time between each trigger is not necessarily the same.

Note

Arrival settings are used only for the primary process definition, because subprocesses will be started appropriately when the subprocess nodes are activated in the parent process instances.

- **Business Calendar**: Business calendars are very important when simulating processes because they allow you to consider only working days and hours in your simulation. A default business calendar is included with your installation of Systemwalker Runbook Automation. The default calendar defines business hours as 8:30 AM to 6:00 PM. The default calendar is stored at the following location:

  <*Systemwalker Runbook Automation Installation Directory*>\ibpm\Data\calendar

  You may create as many Business Calendars as necessary to meet the needs of your organization and use a different Business Calendar for every process definition or process instance. Your Business Calendars are stored in the Workflow Application project directory in the **Calendar** folder.

  To specify a global business calendar for your simulation scenario, select it from the **Business Calendar** drop-down list. You can additionally specify individual business calendars for each human resource allocated in your process definition (see below).

5. In the **Resources** area, define the allocation of resources at scenario level. Human and "non-human" (additional) resources are displayed in a separate table. Human resources are defined based on the roles defined in the imported process definition. Additional resources are defined based on node-level simulation properties defined in the imported process definition.

   The area contains the following information on **Human Resources**:

   - **Currency**: Currency used by this scenario for calculating the cost of resources. Supported currency units are Dollars, Euros, British Pounds, and Japanese Yen. Select the desired currency from the drop-down list.

   - **Role Name**: Role assigned to a particular node in the process definition. You cannot change the role names listed in this area. If required, you need to change the process definition.

   - **Unit of Measurement**: Time unit for calculating the cost of a particular resource. Supported values are "Minute", and "Hour".

   - **Cost Per Unit**: Cost of the resource per unit of measurement.

   - **Count**: Number of allocated resources with this role.

   Except for the **Role Name**, you can change the displayed values by clicking the table cell and entering a value for the corresponding resource parameter.

   The area contains the following information on **Additional Resources**:

   - **Resource Name**: Name of the resource as defined with a node in the process definition. You cannot change the resource names listed in this area. If required, you need to change the process definition.

   - **Unit of Measurement**: Time unit for calculating the cost of a particular resource.

   - **Cost Per Unit**: Cost of the resource per unit of measurement.

   Except for the **Resource Name**, you can change the displayed values by clicking the table cell and entering a value for the corresponding resource parameter.

   Note

   Resources tables show resources of subprocess definitions as well as the parent process definitions.

12.2.4 Defining Simulation Properties for Nodes

This section explains the procedure for setting simulation properties for specific nodes.

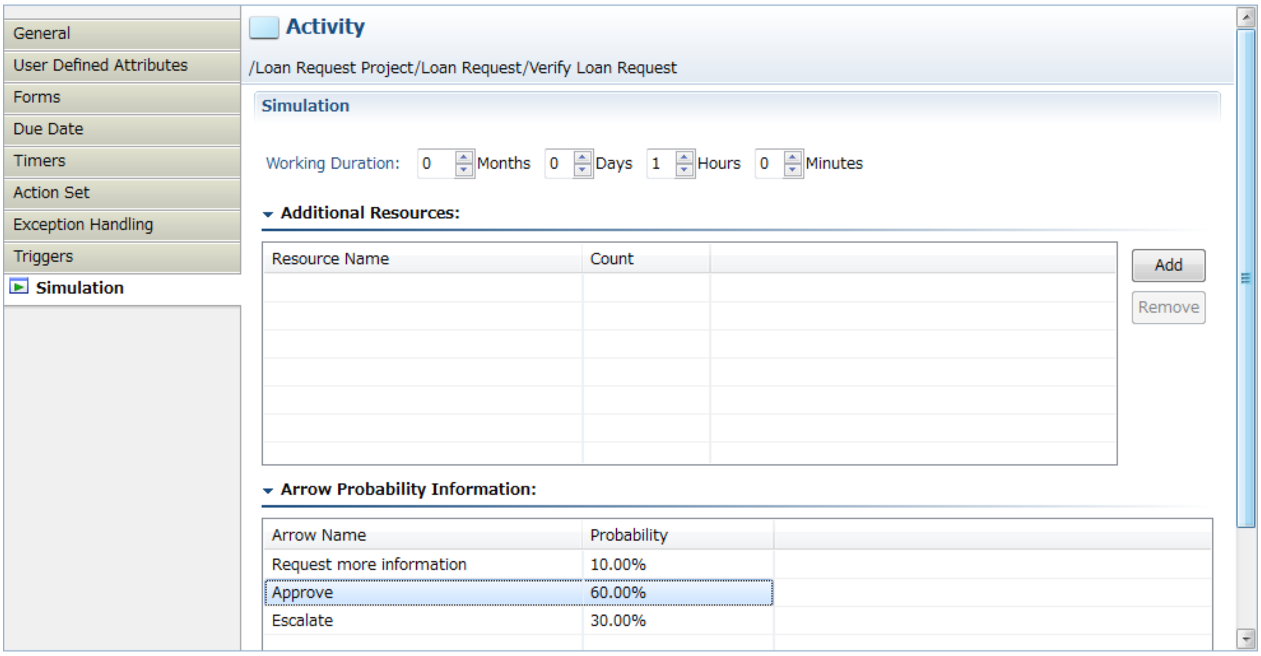
Prerequisite:

- You have created a project with a process definition whose execution you want to simulate. It contains nodes of one or more of the following types:
    - Activity Node
    - Subprocess Node
    - Chained-Process Node
    - Voting Activity Node
    - Conditional Node
    - Complex Conditional Node
    - Compound Activity

To set simulation properties for a particular node:

1. Select the node for which you want to set simulation properties, to display the **Properties** view for it.
2. Select the **Simulation** tab to define the simulation properties for the selected node, for example:

Figure 12.9 Simulation Node Properties view



/Loan Request Project/Loan Request/Verify Loan Request

Simulation

Working Duration: 0 Months 0 Days 1 Hours 0 Minutes

Additional Resources:

Resource Name	Count

Arrow Probability Information:

Arrow Name	Probability
Request more information	10.00%
Approve	60.00%
Escalate	30.00%

3. For Activity Nodes and Voting Activity Nodes: Specify in the two **Working Duration** selection sets, the estimated range of time necessary for a person with the specified Role to complete the task associated with the node. Note that if you specify Months, one month corresponds to 30 days.

   For Conditional and Complex Conditional Nodes, proceed with Step 4.

4. For Activity Nodes and Voting Activity Nodes: In the **Additional Resources** area, any non-human resources required for performing the task associated with the node are listed. The **Resource Name** describes the resource, the **Count** describes the required quantity of this resource.

   To add an additional resource for this node, click the **Add** button. The default values of a new resource are "Resource", and "10". You can change those values by clicking the table cell containing the value and entering a new one.

   To remove a resource, highlight the corresponding row and click **Remove**.

Note

You cannot define simulation properties for Subprocess Nodes and Chained-Process Nodes.

5. In the **Arrow Probability Information** area, all outgoing arrows of the selected node are listed. The default choice probability is as follows: 100% in case of a single choice and for the first arrow, 0% for all additional arrows when there are several outgoing arrows. The same probability distribution is applied when an arrow is newly added to the node.

   The **Probability** value determines the percentage amount applied to making that particular choice. You can change the probability by clicking inside the **Probability** table cell and entering the new value. Ensure that the total always sums up to 100%.

   The following probability distribution is applied when an arrow is deleted from the node: The probability percentage is removed and distributed to each remaining arrow. The distribution factor is decided by the ratio of the remaining arrows. For example:

   ```
   Arrow1     10%
   Arrow2     30%
   Arrow3     60%
   ```

   The probability distribution after removing Arrow3 will be as follows:

   ```
   Arrow1     25%     60% * 10% / (10% + 30%)
   Arrow2     75%     60% * 30% / (10% + 30%)
   ```

The specified probabilities are validated against a total of 100%, and you will receive an error message in the **Problems** view if the probabilities do not add up to 100%.

12.2.5 Saving a Scenario

You can save scenarios that have been modified. If there are no changes, the save functions are not active.

To save a scenario:

1. Do one of the following:

   - To save the scenario that is currently displayed in the Scenario editor, select **File** > **Save**.

   - To save all scenarios that have been modified, select **File** > **Save All**.

2. If a scenario is not valid, for example because the start date is after the end date, a message is displayed telling you so. You cannot save such a scenario. Instead, you must cancel saving, fix the errors and then try to save the scenario again.

   The simulation changes are automatically saved to the Scenario file in the Simulation folder of your project directory. If the process definition imported in a scenario is not valid, you can save the scenario anyway.

   Note

   You can edit simulation results in the Scenario editor. If you save your edits, the Scenario file will be saved and can be reused. The original simulation results, however, will be deleted.

12.3 Executing a Simulation Scenario

This section explains how to execute a simulation scenario, and describes the simulation results.

12.3.1 Preparing a Simulation Result

Once finished with the definition of the simulation properties, you prepare your scenario for the simulation run.

To prepare the simulation result, do one of the following:

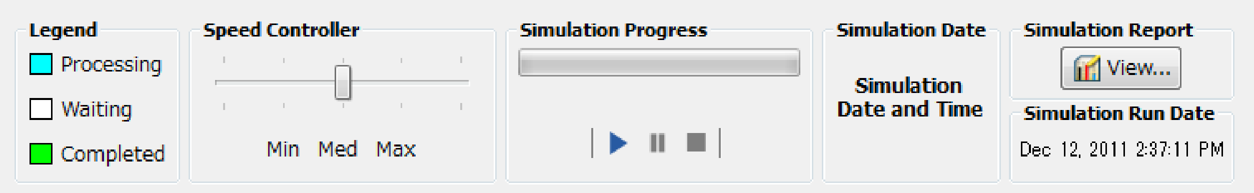
- Right click the scenario in the Navigator view, and select **Run Simulation** from the popup menu.

- Click **Run Simulation** in the Scenario editor in which the scenario is displayed.

Your scenario will be validated. For example, it will be checked whether you imported a valid process definition, whether all units for measurement have been specified, et cetera. If an error occurs, you are returned to the Scenario editor and you need to fix the error(s) before you can proceed with the simulation process.

If no errors occur, the simulation results are automatically added to the Scenario (.ssr) file and stored in the Simulation folder of your project. The Scenario editor displays the process definition you have selected for simulation. The Simulation Controller is displayed below the Scenario editor:

Figure 12.10 Displaying the Simulation Controller



Note

If errors occur while the process definitions in the scenario are being validated, all these errors are displayed in the **Problems** view. The error details such as error description, resource where the error occurred, error location and project are displayed in this view. These details are displayed in **Description**, **Resource**, **Project Pass**, **Location** and **Typ**e columns respectively. In the **Location** column, the location of the error is displayed in the form of the process definition name and the actual location of the error in that process definition. These two are separated by a delimiter (/).

The Simulation Controller is used for starting, stopping, pausing a simulation run, for controlling the speed of a simulation run, and for generating simulation reports. The Simulation Controller consists of the following parts:

- **Legend**: Shows if the simulation run is still waiting to be executed, just processing, or already completed.
- **Speed Controller**: Controls the speed at which activities are processed and animated during the simulation.
- **Simulation Progress**: Shows the progress of a simulation run. Below the progress bar, you find the following buttons:

Button	Action	Description
▶	Play	Starts the execution of the simulation based on the selected scenario.
❚❚	Pause	Pauses the simulation and saves its intermediate history. You can resume the simulation run by clicking the Play button again.
■	Stop	Cancels the execution of the simulation run.

- **Simulation Date**: Displays the date and time of the simulation.
- **Simulation Report**: Displays the **View** tab. Use this tab to generate simulation reports. The reports are displayed in a separate window.

Note

Simulation Report is not displayed in Simulation Controller of the subprocess definition.

For more information on executing a simulation scenario, refer to section 12.3.2 Using the Simulation Controller.

12.3.2 Using the Simulation Controller

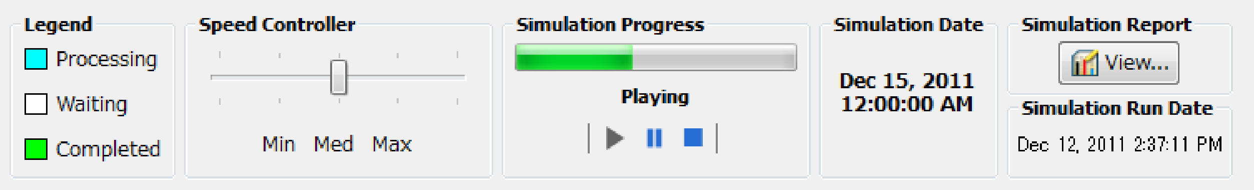
Running a simulation actually displays an animation of the items to be processed arriving at the individual nodes (Activity Node, Subprocess Node, Chained-Process Node, Voting Activity Node, Conditional Node, Complex Conditional Node, Compound Activity) and being sent on to other nodes for processing.

Note

If you simulate a process definition containing a Subprocess Node or Chained-Process Node, you can click the respective tabs of the subprocess definitions in the Scenario Editor and use the Simulation Controller to run the animation of their respective simulations.

You use the Simulation Controller for starting, pausing or stopping a simulation run, and for generating simulation reports:

Figure 12.11 Simulation Controller

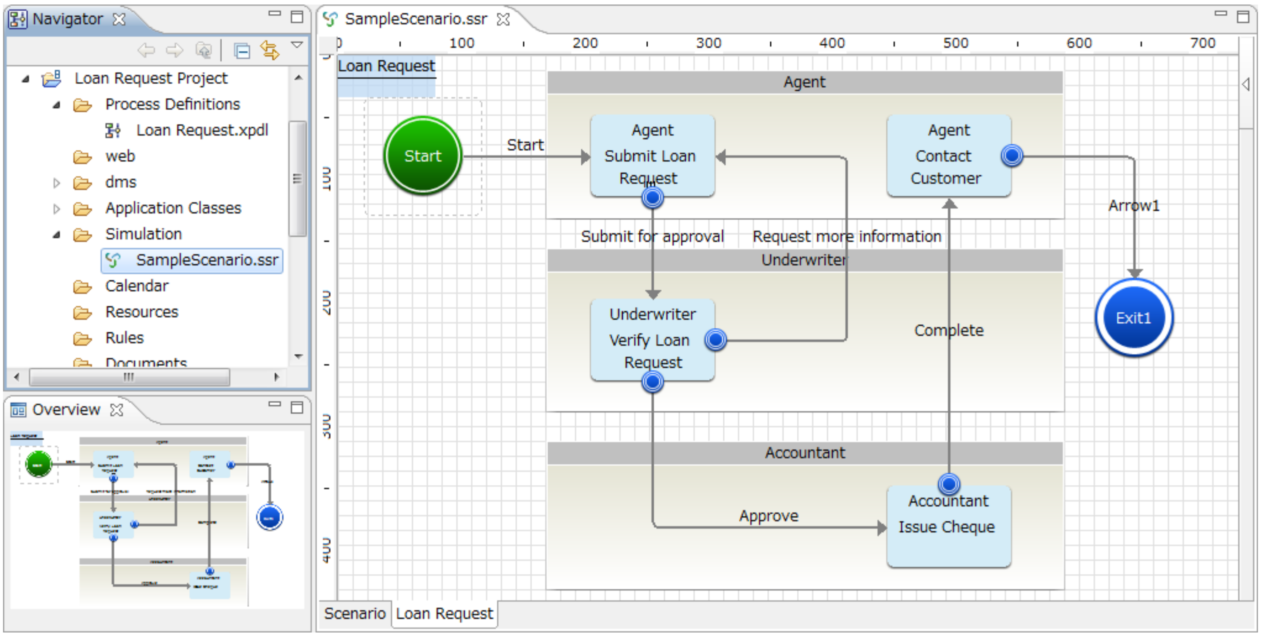


To execute a simulation scenario:

1. Prepare a simulation, as described in section 12.3.1 Preparing a Simulation Result.
2. In the Simulation Controller, define the speed with which the processing of the nodes will be animated and displayed. In this way you can control the pace of the simulation: Adjust the slider of the **Speed Controller** by dragging it to the desired position between **Min** (minimum pace, e.g. hour by hour), **Med** (medium pace, for example day by day) to **Max** (maximum pace, for example an entire week). You can also instantly click **Min**, **Med**, or **Max**.
3. Click the **Play** button.

   The process definition imported in the selected scenario will be opened in the Scenario editor view. In the following example, the process definition (extension "xpdl") has been imported into the scenario file (extension "ssr"):

   Figure 12.12 Simulating a process definition



   In the course of the simulation run, you can observe the following:

   - The **Simulation Date** area in the Simulation Controller shows the virtual date and time of the running simulation. It starts on the start date defined in the simulation scenario and ends on the end date you defined in the scenario.
   - The **Simulation Progress** bar indicates the percentage of the simulation that has been completed so far, and the **Legend** explains the meaning of the Simulation Progress bar colors during simulation animation. The same colors are used for showing the processing status of the individual nodes in the process definition.

- The processing of the process definition is animated, The progress of the simulation run is shown for every node, for example whether an activity has been completed, which outgoing arrow has been chosen, how many process instances have been triggered, et cetera.

- The arrow transition probability that you define is displayed near each arrow in the process definition.

4. To **pause** the execution of the scenario, click the **Pause** button. To resume the processing again, click the **Play** button.

   Note

   To **stop** the execution, click the **Stop** button. This will cancel the entire simulation run. The animation of the processing of the process definition is controlled in the Simulation Controller in each tab.

5. In the Simulation Report area of the Simulation Controller, click **View** to generate a simulation report.

   Refer to section 12.3.3 Generating Simulation Reports for details.

12.3.3 Generating Simulation Reports

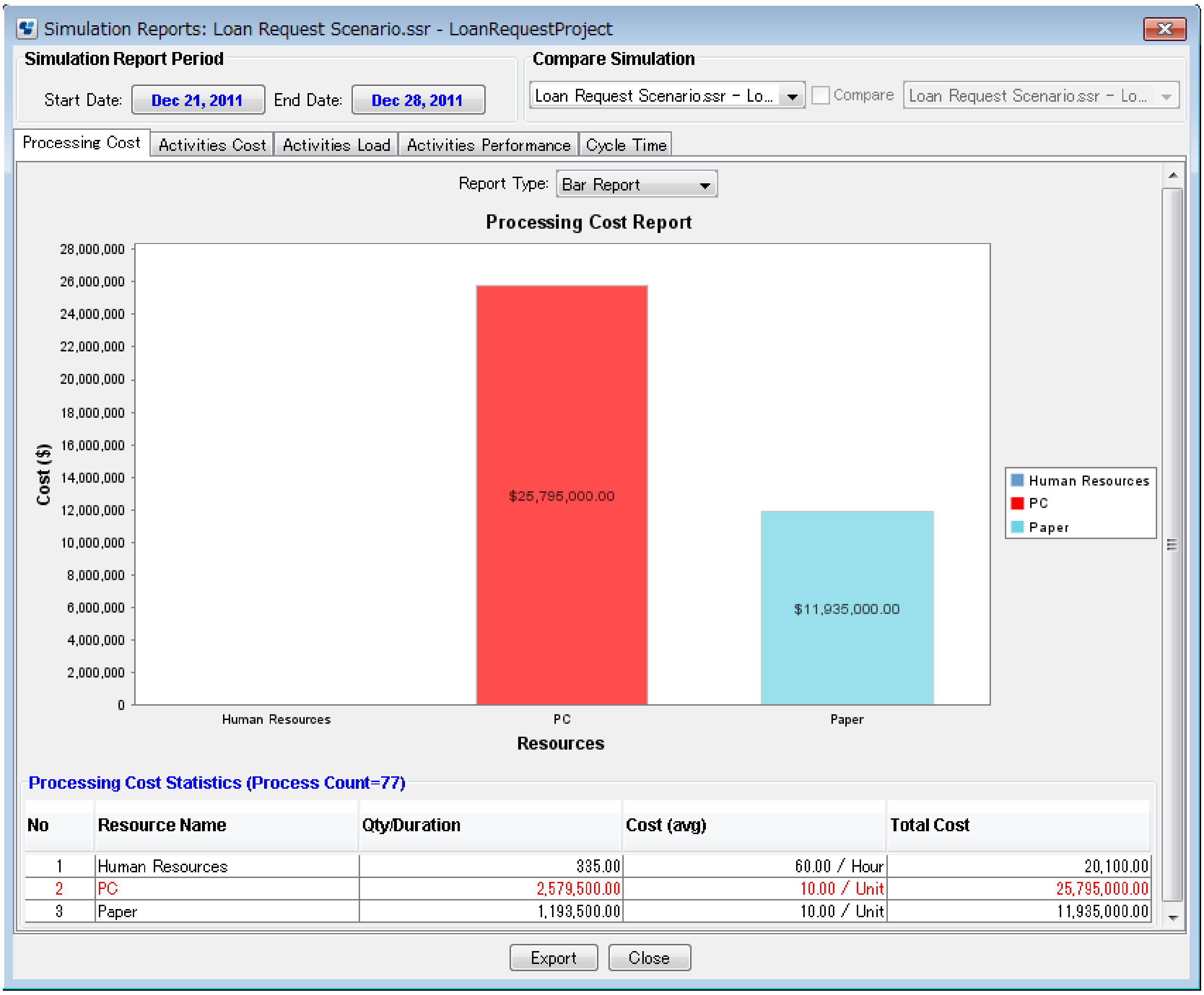
Simulation reports will help you analyze operating costs. Operating costs are associated with process bottlenecks and resource utilization. You can export the report data and charts in HTML, PDF, and CSV format.

Simulation reports also support user personalization for corporate sharing. Reports are created to show one process at a time. The process ID of this process instance is supplied by the simulation process itself.

1. To generate simulation reports for a particular scenario: In the Simulation Controller, click the View button.
   The Simulation Report will open in a new window displaying the name of the scenario and the date when the simulation was executed

in the title bar, and the report start and end date. This date is, by default, the start and end date specified in the scenario that has been executed. For example:

Figure 12.13 Simulation Reports Window



Simulation Reports: Loan Request Scenario.ssr - LoanRequestProject

Simulation Report Period

Start Date: Dec 21, 2011 End Date: Dec 28, 2011

Compare Simulation

Loan Request Scenario.ssr - Lo... □ Compare Loan Request Scenario.ssr - Lo...

Report Type: Bar Report

Processing Cost Statistics (Process Count=77)

No	Resource Name	Qty/Duration	Cost (avg)	Total Cost
1	Human Resources	335.00	60.00 / Hour	20,100.00
2	PC	2,579,500.00	10.00 / Unit	25,795,000.00
3	Paper	1,193,500.00	10.00 / Unit	11,935,000.00

Note

Simulation Report is displayed only in the Simulation Controller of the primary process definition. This is not displayed in the Simulation Controller of the subprocess definition.
This report shows the total of all the process definitions.

2. To review a particular report, click its tab. Refer to section 12.3.4 Simulation Report Types for information on the various report types.

3. By default, the report start and end date is the same as the dates supplied in the scenario you executed. You can change the start and end date of the report as follows:

   In the **Simulation Report Period** area, click the respective date button and select the desired date from the popup calendar. After clicking **OK**, the report will change accordingly.

Note

To view the initial start and end date again, display the tool tip provided for the simulation result in the **Compare Simulation** area.

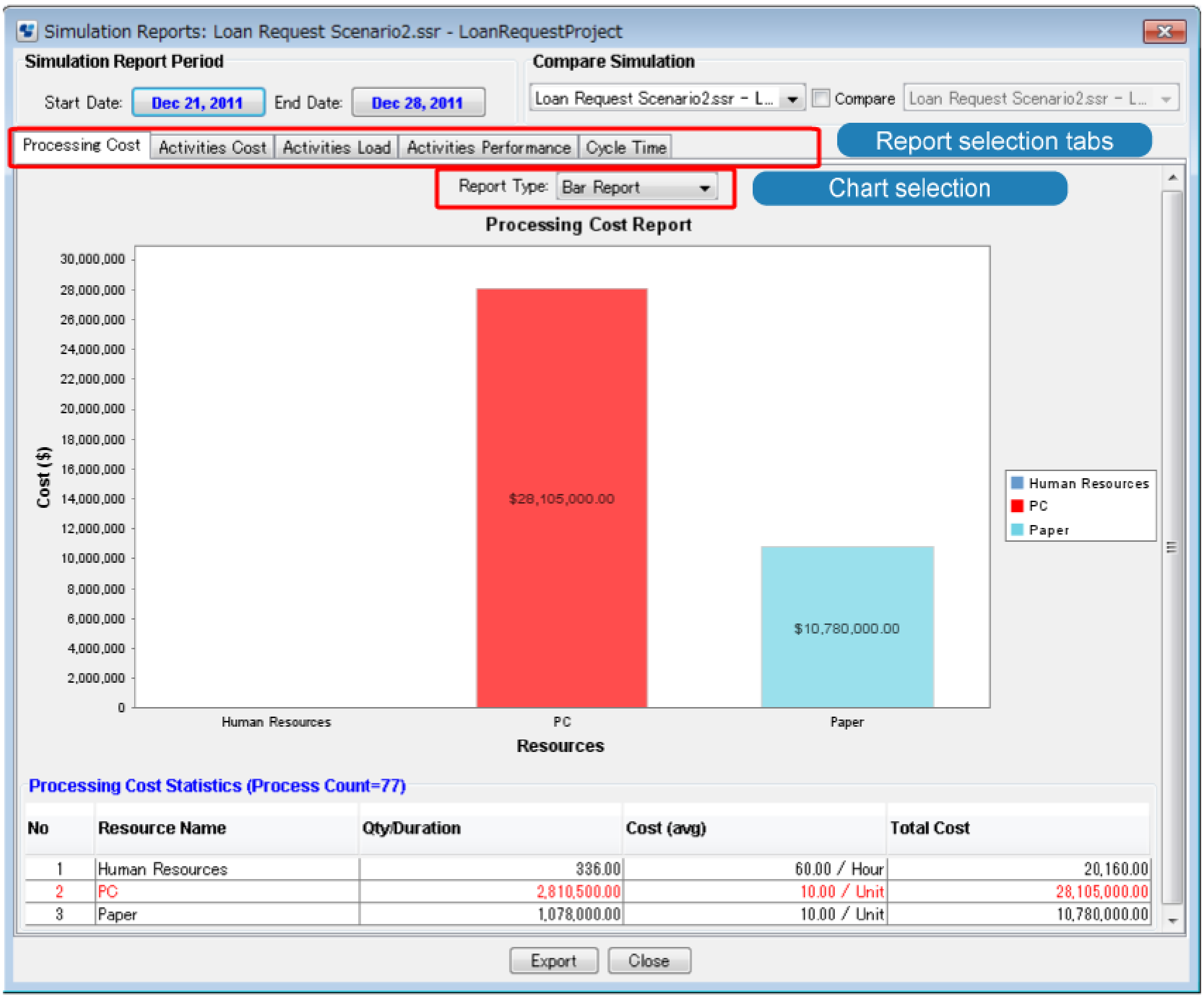
4. Use the **Compare Simulation** area to select the a simulation result from the list and view comparative reports. Refer to section 12.3.6 Comparing Simulations for details.

5. You can export your simulation reports by clicking the **Export** button. Refer to section 12.3.5 Exporting Simulation Reports for details.

12.3.4 Simulation Report Types

Whenever you have executed a simulation run, you can generate simulation reports. A simulation report is displayed in a separate window that consists of various parts. The parts depend on the type of report displayed. For example:

Figure 12.14 Simulation Report Parts



Simulation Reports: Loan Request Scenario2.ssr - LoanRequestProject

Simulation Report Period

Start Date: Dec 21, 2011 End Date: Dec 28, 2011

Compare Simulation

Loan Request Scenario2.ssr - L... □ Compare Loan Request Scenario2.ssr - L...

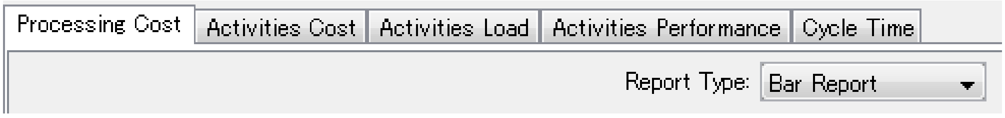
Report Type: Bar Report

Processing Cost Statistics (Process Count=77)

No	Resource Name	Qty/Duration	Cost (avg)	Total Cost
1	Human Resources	336.00	60.00 / Hour	20,160.00
2	PC	2,810,500.00	10.00 / Unit	28,105,000.00
3	Paper	1,078,000.00	10.00 / Unit	10,780,000.00

The various types of report can be displayed by clicking the respective tab:

Figure 12.15 Report Types



Report Type: Bar Report

You can change the layout of a report by selecting the relevant entry from the **Report Type** drop-down list. For example, a Processing Cost report can be displayed as bar report or pie report.

A detailed report for the individual activities, resources, etc. can be opened by double-clicking an entry in the **Statistics** area of the **Simulation Reports** window.

The following types of report are available:

Processing Cost Report

The Processing Cost report shows the total amount of cost (based on the defined resources) of the simulated process activities within the specified period. The cost of human resources is shown in one group named as "Human Resources". The cost of all other resources are shown by a separate bar or pie segment for each resource.

The highest or critical cost flashes and is shown in the color red. The row in the Processing Cost Statistics containing the critical values (highest values among the values in the table) is also shown in red. In addition, the Processing Cost Statistics show the number of process instances completed between the report start date and the report end date.

Double-clicking any particular resource opens the detail report for this resource, and processing cost drilled down to the selected resource only will be shown. Information on the other resources is available by clicking the respective tab.

Activities Cost

The Activities Cost report shows the cost for every performed activity, i.e. for every node that has been processed, calculated on the basis of every resource assigned to an activity. The costs are divided into the total amount of human resource costs and other resource costs.

Double-clicking any particular activity opens the detail report for this activity. Information on other activities is available by clicking the respective tab.

Activities Load

The Activities Load report shows, for each activity, how often a it has been triggered and processed. For each activity, you see how often it has been completed and how many instances are still in a waiting or working state.

Double-clicking any particular activity opens the detail report for this activity. Information on other activities is available by clicking the respective tab.

Activities Performance

The Activities Performance report shows, for every activity, the total processing time. The processing time is combined of waiting hours (hours passed during which the activity could not be processed because another instance was still in progress) and processing hours (hours it took in order to complete the activity instances).

Double-clicking any particular activity opens the detail report for this activity. Information on other activities is available by clicking the respective tab.

Cycle Time

The Cycle Time report shows the number of process instances completed between the report start and end date in three categories: minimum days, maximum days, and average days.

Note

If you are simulating a process definition which contains Subprocess Node or Chained-Process Node, the **Cycle Time** report of the simulation displays the process instance cycle information of only the primary (parent) process definition.

Comparative Reports

Simulation reports can be generated on two sets of simulation data for comparison purposes. Such a comparison results in the generation of comparative reports. You can consider only one process definition at a time and both of the simulations must have a result corresponding to the same process definition, but the simulations may have different scenarios. The above-mentioned reports are available as comparative reports and behave in the same way as reports for single simulations.

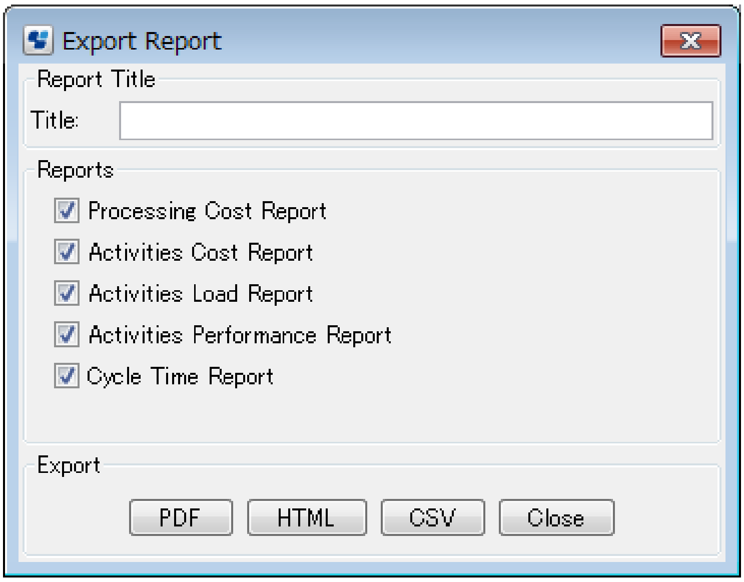
12.3.5 Exporting Simulation Reports

Simulation reports can be exported in the following formats: PDF, HTML, or CSV.

To export your simulation reports:

1. In the Simulation Report view, click the **Export** button to open the **Export Report** dialog box.

   Figure 12.16 Export Report Dialog

   Export Report

   Report Title

   Title:

   Reports

   ☑ Processing Cost Report

   ☑ Activities Cost Report

   ☑ Activities Load Report

   ☑ Activities Performance Report

   ☑ Cycle Time Report

   Export

2. Enter a title for the report.

3. Select the reports that you want to export by activating the appropriate check boxes. Refer to section 12.3.4 Simulation Report Types for information on the available reports.

4. Click the button corresponding to the format that you want to export (**PDF**, **HTML**, or **CSV**). This will open a **Save** dialog box.

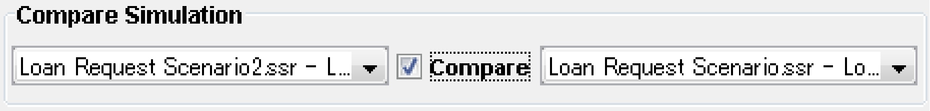
5. Enter a file name and navigate to a location for your report. Then, click **Save**.

   The report will be saved with the name and location specified.

12.3.6 Comparing Simulations

You can compare the results of two simulation runs and view comparative reports. Both simulations must use a scenario that imports the same process definition.

Figure 12.17 Comparing simulation results



Compare Simulation

Loan Request Scenario2.ssr - L... ☑ Compare Loan Request Scenario.ssr - Lo...

To compare two simulation results:

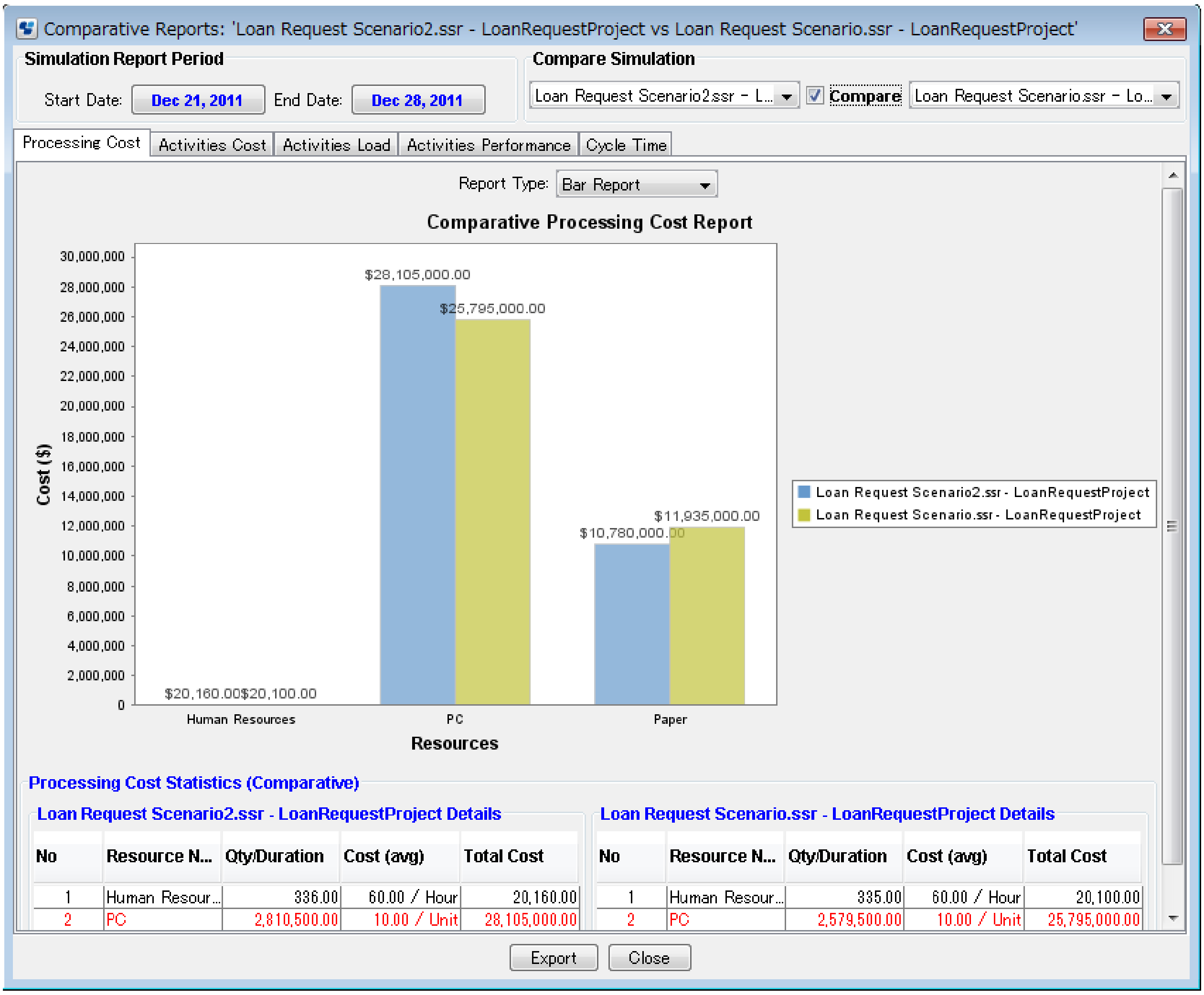
1. In the **Compare Simulation** area of the **Simulation Reports** view, select the two simulations that are to be compared from the drop-down lists.

   By default, the drop-down list on the left displays the scenario that is currently open in the Scenario editor. The drop-down list on the right displays all scenarios simulating the same process definition that is being used in the current scenario. The available entries in the drop-down lists consist of the name of the scenario and scenario file name.

2. Activate the **Compare** check box.

   The content of the **Simulation Reports** window will change and display comparative reports. Refer to section 12.3.4 Simulation Report Types for details. The following example shows a comparative report.

Figure 12.18 Displaying a comparative report



Comparative Reports: 'Loan Request Scenario2.ssr - LoanRequestProject vs Loan Request Scenario.ssr - LoanRequestProject'

Simulation Report Period

Start Date: Dec 21, 2011 End Date: Dec 28, 2011

Compare Simulation

Loan Request Scenario2.ssr - L... ☑ Compare Loan Request Scenario.ssr - Lo...

Report Type: Bar Report

Processing Cost Statistics (Comparative)

Loan Request Scenario2.ssr - LoanRequestProject Details

No	Resource N...	Qty/Duration	Cost (avg)	Total Cost
1	Human Resour...	336.00	60.00 / Hour	20,160.00
2	PC	2,810,500.00	10.00 / Unit	28,105,000.00

Loan Request Scenario.ssr - LoanRequestProject Details

No	Resource N...	Qty/Duration	Cost (avg)	Total Cost
1	Human Resour...	335.00	60.00 / Hour	20,100.00
2	PC	2,579,500.00	10.00 / Unit	25,795,000.00

12.4 Managing Simulation Scenarios

This section explains the functions available for managing simulation scenarios.

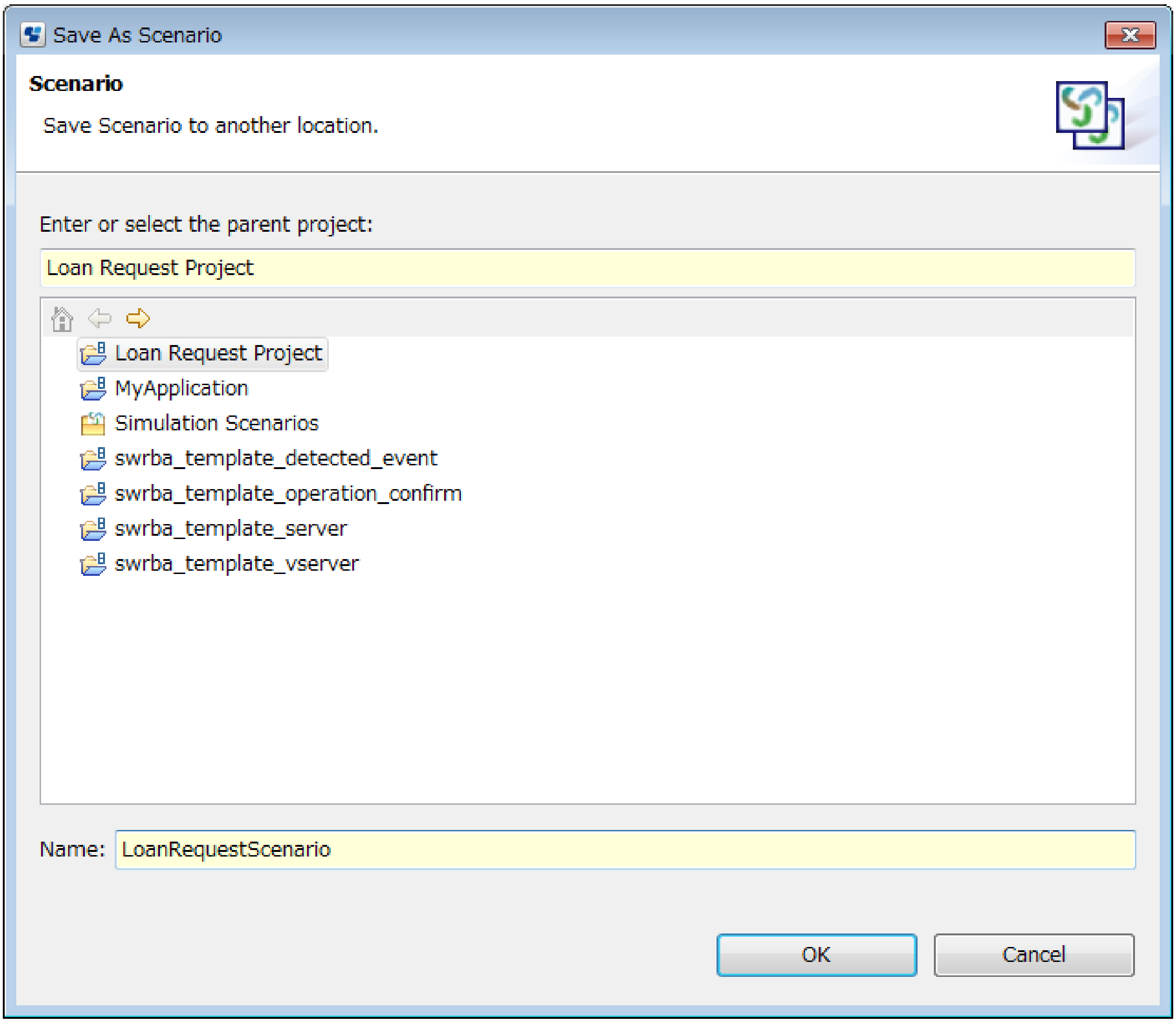
12.4.1 Saving a Scenario to a Different Name or Project

To save a scenario to a different name or project:

1. Click the Scenario editor that displays the scenario to be saved.

2. Select **File** > **Save As**.

Figure 12.19 Saving a Scenario to a Different Name



Scenario

Save Scenario to another location.

Enter or select the parent project:

Loan Request Project

Name: LoanRequestScenario

3. In the **Save As Scenario** dialog, select the project where you want to save the scenario. You can save it to any Workflow Application project or to the "Simulation Scenarios" project. You cannot save it to server projects. When saving it to a Workflow Application project, it will always be stored in the Simulation folder of this project.

4. If you want to save the scenario to a different name, type the new name in the **Name** field.

5. Click **OK**.

6. If a scenario is not valid, for example because the start date is after the end date, a message is displayed telling you so, and you cannot save such a scenario. Instead, you must cancel saving, fix the errors and then try to save the scenario again.

   Note that if the process definition imported in a scenario is not valid, you can save the scenario anyway.

12.4.2 Changing the Scenario Name and Description

The name of a scenario is used to identify it, therefore you must specify a name when creating a scenario. You can use a description to provide additional information on the process.

To change the name and description for a scenario:

1. In the Navigator view, double click the scenario to open it in the Scenario editor.

2. Change the name and description, as desired, in the **Name** and **Description** fields.

3. Save the scenario when finished by selecting **File** > **Save**.

12.4.3 Opening a Scenario

To open a scenario, do one of the following:

- In the Navigator view, double click the scenario.
- In the Navigator view, right click the scenario and select **Open** from the pop-up menu.

12.4.4 Closing a Scenario

To close a scenario:

1. Do one of the following:
    - To close a particular scenario, click the **Close** button of the Scenario editor.
    - To close all scenarios, select **File** > **Close All**.
2. If there are unsaved changes, a message is displayed telling you so. Do one of the following:
    - To close the scenario and save your changes, click **Yes**.
    - To close the scenario without saving your changes, click **No**.

12.4.5 Importing a Scenario

You can import scenarios in XML format into Systemwalker Runbook Automation Studio.

To import a scenario:

1. In the Navigator view, right click the Simulation Scenarios project and select **Import** from the pop-up menu.
2. Navigate to the location where the scenario is stored.
3. Select the scenario and click **Open**.
4. If a scenario with the same file name already exists in the scenario, a dialog is displayed telling you so. Rename the file to a unique name and restart importing.

12.4.6 Exporting a Scenario

You can export scenarios from Systemwalker Runbook Automation Studio to the file system. You can use the exported files, for example, to import them into other systems. The export format is XML.

To export a scenario:

1. In the Navigator view, right click the scenario. Select **Export Scenario**.
2. Navigate to the location where the exported scenario is to be stored.
3. Click **Save**.

12.4.7 Copying a Scenario

You can copy scenarios easily in the Navigator view.

To copy a scenario:

1. In the Navigator view, right click the scenario that you want to copy and select **Copy** from the pop-up menu.
2. Right click the **Simulation Scenarios** folder scenario and select **Paste** from the pop-up menu.
3. Type a new file name and click **OK**.

You can copy several scenarios at once. Hold down the <Shift> key or <Ctrl> key while selecting all scenarios to be copied, and - for each scenario - enter a new name.

12.4.8 Renaming a Scenario

You can specify a new file name for a scenario.

Note

You can use this function only if you set your preferences for the Navigator view to **File name**.

To rename a scenario:

1. Right click the scenario in the Navigator view. Select **Rename** from the pop-up menu.
2. Type the new name for the scenario.
3. Press the <Return> key.
4. If a scenario using the same file name already exists in the Simulation Scenarios project, a dialog is displayed telling you so. Do one of the following:
    - To overwrite the existing scenario, click **Yes**.
    - To keep the existing scenario, click **No**.

      In this case, the scenario is not renamed.

12.4.9 Removing a Scenario

When you remove a scenario, it is removed from Systemwalker Runbook Automation Studio and from the file system as well.

To remove a scenario:

1. Right click the scenario in the Navigator view. Select **Delete** from the pop-up menu.
2. Confirm the removal with **Yes**.

Chapter 13 Process Debugging

This chapter explains process debugging using Systemwalker Runbook Automation Studio.

13.1 Overview of Debugging

This section provides an overview of the debugging operations.

The following main debugging operations are available using Systemwalker Runbook Automation Studio:

- Remote Connection (Connection to the Management Server), Debug Configuration
- Debugging
- Process instance start, suspend, and step motion
- Browse/update variables
- Jump to the Process Definition

Note that the following tasks will be required if it is necessary to update the process definition and operation component while debugging:

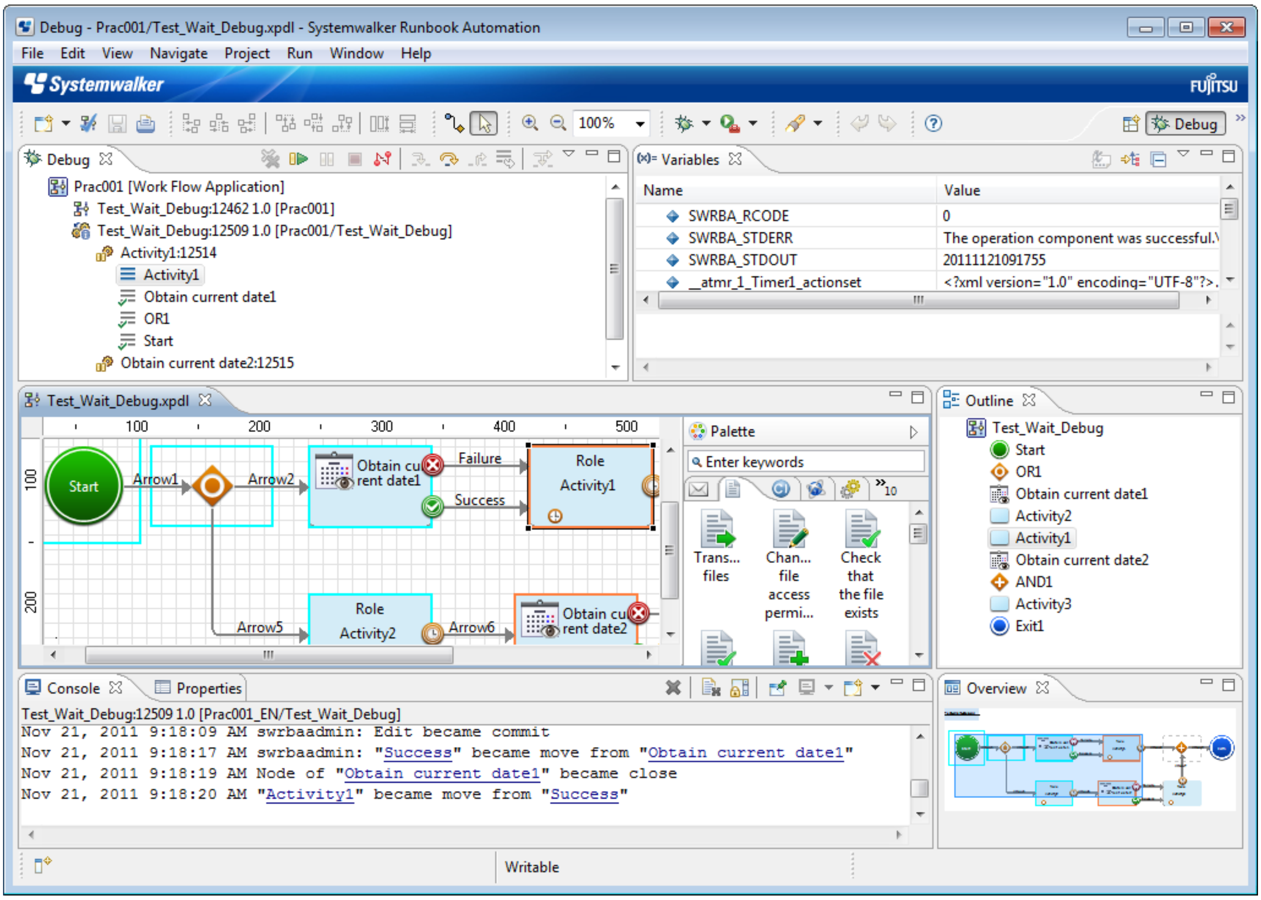
- Upload application project
  Refer to "3.1.15 Uploading Workflow Application Projects to a Server" for details.
- Upload operation component
  Refer to "5.6 Uploading to the Operation Component Server" for details.
- Change Automated Operation Process Group to online status
  Operate from the Web console. Refer to *information on placing Automated Operation Process Groups online* in the *Systemwalker Runbook Automation Operation Guide* for details.

13.2 Window Configuration

The following main windows (perspective, view, and dialog box) are provided as the debugging functions.

13.2.1 Debug Perspective

This is the standard window configuration for debugging, which allows related tool buttons and view operations to be performed quickly.

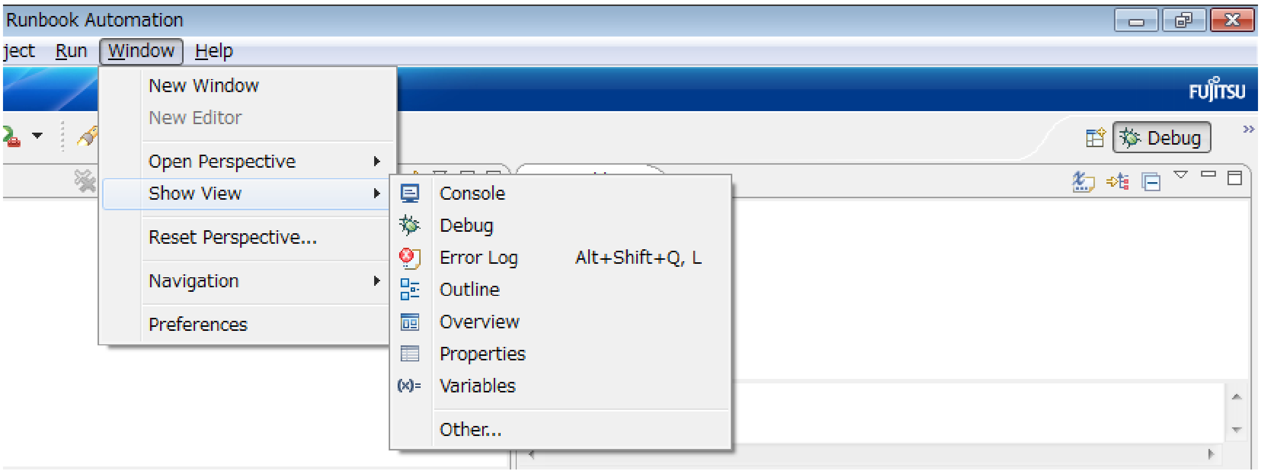


Name	Value
SWRBA_RCODE	0
SWRBA_STDERR	The operation component was successful.\
SWRBA_STDOUT	20111121091755
__atmr_1_Timer1_actionset	<?xml version="1.0" encoding="UTF-8"?>.

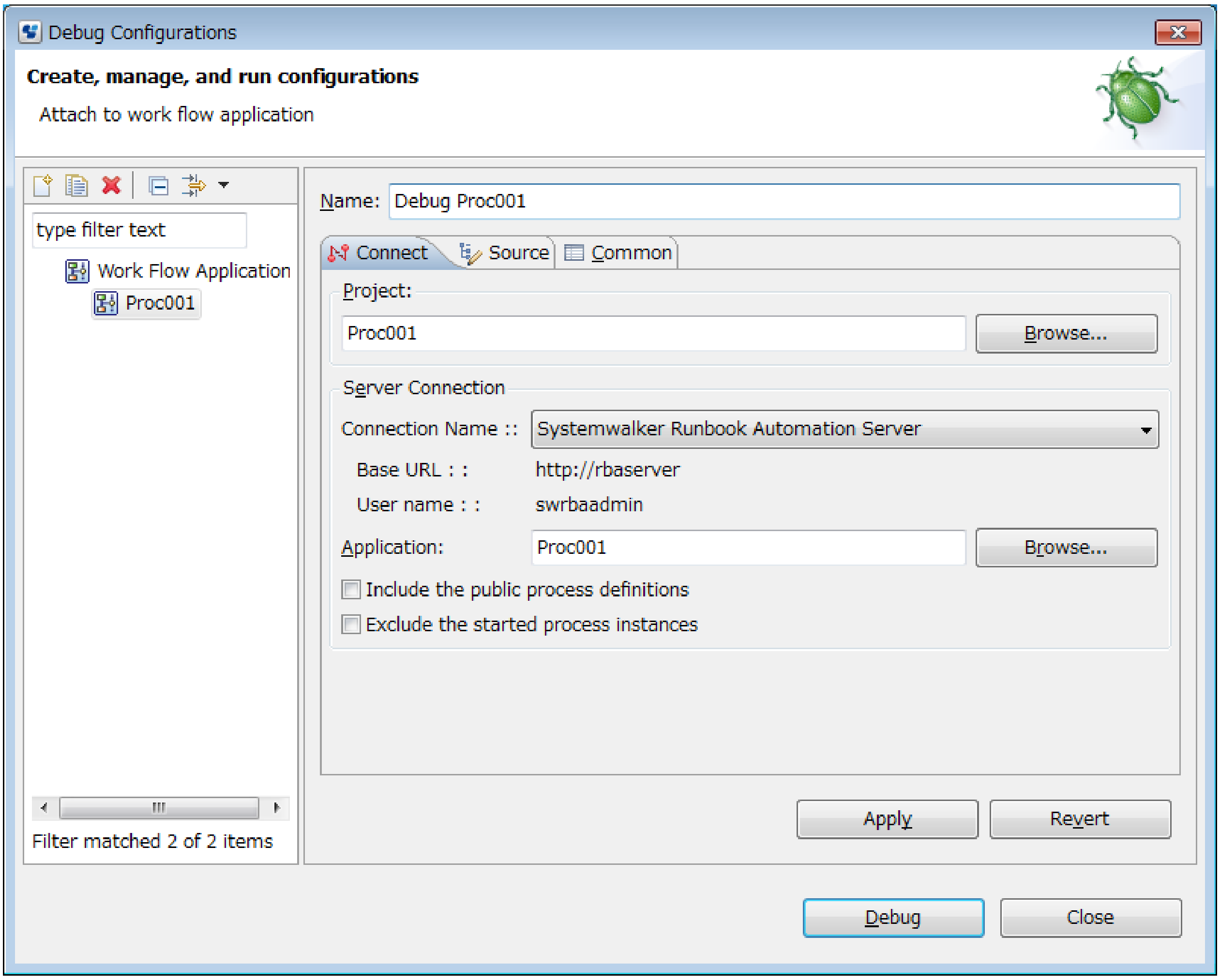
Test_Wait_Debug:12509 1.0 [Prac001_EN/Test_Wait_Debug]

```
Nov 21, 2011 9:18:09 AM swrbaadmin: Edit became commit
Nov 21, 2011 9:18:17 AM swrbaadmin: "Success" became move from "Obtain current date1"
Nov 21, 2011 9:18:19 AM Node of "Obtain current date1" became close
Nov 21, 2011 9:18:20 AM "Activity1" became move from "Success"
```

Additionally, the **Window** >> **Show View** submenu can be used to open each view.



13.2.2 Debug Configurations Dialog



Create, manage, and run configurations

Attach to work flow application

type filter text

- Work Flow Application
  - Proc001

Filter matched 2 of 2 items

Name: Debug Proc001

Project:

Proc001

Server Connection

Connection Name :: Systemwalker Runbook Automation Server

Base URL : : http://rbaserver

User name : : swrbaadmin

Application: Proc001

□ Include the public process definitions

□ Exclude the started process instances

In the **Name** field, specify a unique name for debug configuration. This name cannot contain the /, \, :, &, *, ?, ", <, >, | or @ symbols.

In the **Connect** tab, specify the project to be debugged (this project includes the process definition) and the Management Server to connect to.

- **Project**: Enter the project name directly, or browse the project names and select the project name.
- **Server connection setting**: From the list in **Server connection settings** under **Window** >> **Preferences** >> **Interstage BPM Studio** of the workbench, select the connection information for the server which will be connected to using the debug configuration. In the **Connection name** combo-box, the base URL and user name are displayed as information elements to be confirmed from the Server connection setting that was selected. This information element string can also be selected and then copied. In Application field select the application containing the process definition of the project. If no application is selected. When an application is selected all process definitions on the management server will be selected. You can also Browse and choose applications from Management server.
- **Include the public process definitions**: Select this to make public process definitions the subject of the process.
- **Exclude the started processinstances:** Select this to exclude process instances that had already been started when debugging was started, and thereby reduce the amount of time it will take till debugging can be started.

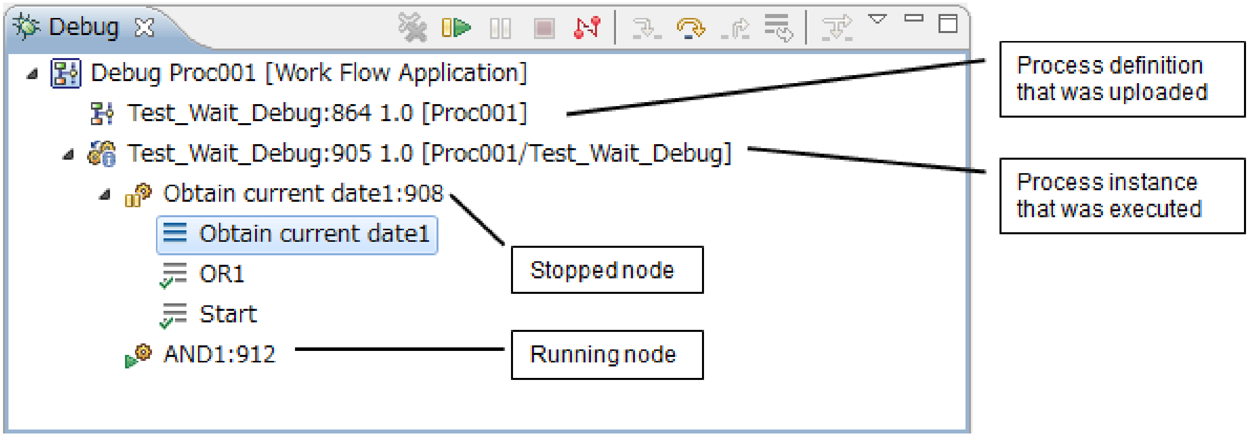
In the **Source** tab, define projects related to the project defined in the **Connect** tab if necessary.
Note: When used in Systemwalker Runbook Automation, related projects do not usually need to be defined in this tab.

In the **Common** tab, define the storage location of the Debug configuration, and advanced options for operations when the debug configurations is running.
Note: When used in Systemwalker Runbook Automation, options do not usually need to be specified explicitly in this tab.

Note

In Systemwalker Runbook Automation you cannot select (button) to open Debug configuration.

13.2.3 Debug View



Debug

- Debug Proc001 [Work Flow Application]
  - Test_Wait_Debug:864 1.0 [Proc001]
  - Test_Wait_Debug:905 1.0 [Proc001/Test_Wait_Debug]
    - Obtain current date1:908
      - Obtain current date1
      - OR1
      - Start
    - AND1:912

Process definition that was uploaded
Process instance that was executed
Stopped node
Running node

Workflow application debugging is managed in the **Debug** view. In this view, the nodes which have been executed or suspended (including operation components, when these exist) will be displayed for each process definition process instance to be debugged (the debugging target). Each process instance node is displayed as a tree node.

Tree Node Display Format

The node on the tree is displayed in the following format. (The debugging target is displayed in italics.)

Type	Display format
Debug configuration	*Name of debug configuration* [Work Flow Application]
Process definition	*Name of process definition Version* [*Name of application*]
Instance	*Name of process definition:Identifier ID Version* [*Name of application*/ *Name of instance*]
Node or activity	*Name of node or activity:Identifier ID*
Stack	☰ *Name of node*

The Process definition icons are displayed according to the status of the Process definition. They are displayed in the following format.

Image	Description
	This is a draft status Process definition.
	This is a public Process definition.

The instance icons are displayed according to the status of the instance. They are displayed in the following format.

Image	Description
⚙	This is an instance in which all the nodes are running.
⚙	This is an instance in which some of the nodes have a no execute status (for example, "suspended").

Node icons show the status of nodes:

Image	Description
▶	This is a node that is running.
⏸	This is a node that is suspended.
▶⏳	This is a node that is running but which has received a request to suspend.
⏸⏳	This is a node that is suspended but which has received a request to resume.
▤	This is a node that is running. The node has a task.
▤⏳	This is a node that is running but which has received a request to suspend. The node has a task.
✓	This is a node that has ended normally.
☒	This is a node that has ended abnormally.
⛔	This is a node that was stopped by force.

The stack displays routes (nodes) executed between the Start Node and the position at which the interruption occurred, in the reverse order of execution times. The node icons displayed for the stack, in accordance with the status of the nodes, are as follows:

Image	Description
≡	This is a node that is suspended.
≡▶	This is a node that is running.
≡⏳	This is a node that is waiting.
≡✓	This is a node that has ended normally.
≡☒	This is a node that has ended abnormally.
≡⛔	This is a node that was stopped by force.

Note

- Stacks are displayed in the reverse order of execution times for each node and not necessarily expressed as the transition from the bottom node to the top node.
- If any of the nodes on the route between the Start and interrupted nodes are currently being executed or currently being interrupted, it will not be possible to display the route for the interrupted node correctly.
- The Iterator (Parallel) Loop node displays the node generated last.
- All nodes generated by the Iterator (Parallel) Loop node must be completed to transition from the Iterator (Parallel) Loop node to the next node.
- The Sequential Loop node displays the last node to be executed.

Tree Node Display Order

The order in which each node under the debug configuration node on the tree is displayed is as follows:

- Process definition: Displayed in ascending order; not case-sensitive
- Instance: Displayed in ascending order of instance identifier ID
- Node: Displayed in ascending order of instance identifier ID

Tree Node Display Range

Nodes are displayed on the tree according to the following conditions:

- Process definition: The root folder of the project defined in the debug configuration is positioned, and only process definitions which have already been uploaded to the Management Server and have draft status are displayed.
  If there are multiple versions of the process definition, only the latest version will be displayed.
- Process instance: The root folder of the project defined in the debug configuration is positioned and only process definitions which were instantiated on the Management Server at the time of the server connection and instances started from the **Debug** view are displayed.

Note

Subprocess nodes or process definitions executed as subprocess by operation components cannot be debugged at the same time. Process definitions that are to be executed as subprocess should be debugged in advance.

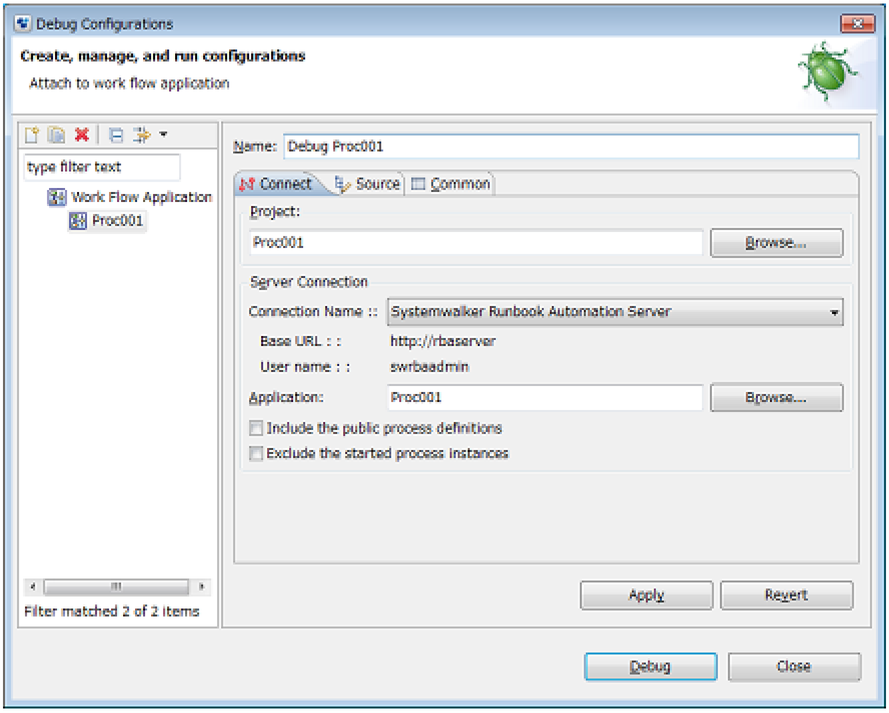
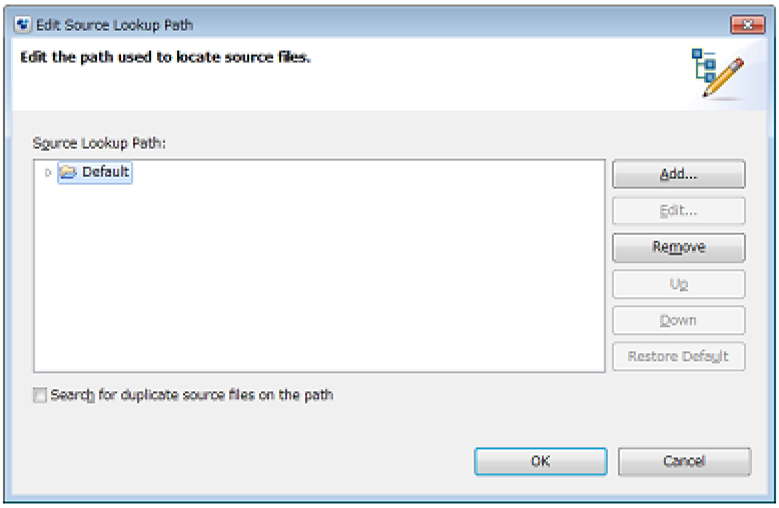
Properties for Tree Nodes

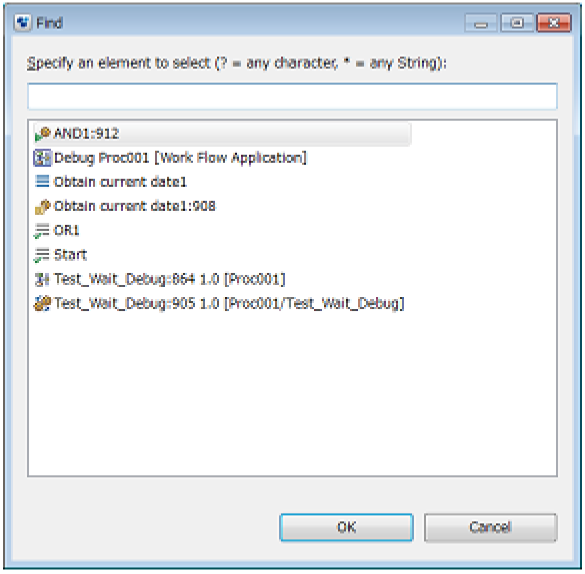
The information for nodes displayed in trees is displayed as properties.

Available Operations

In the **Debug** view, run process instances in step mode and suspend, resume, and disconnect using the following operations.

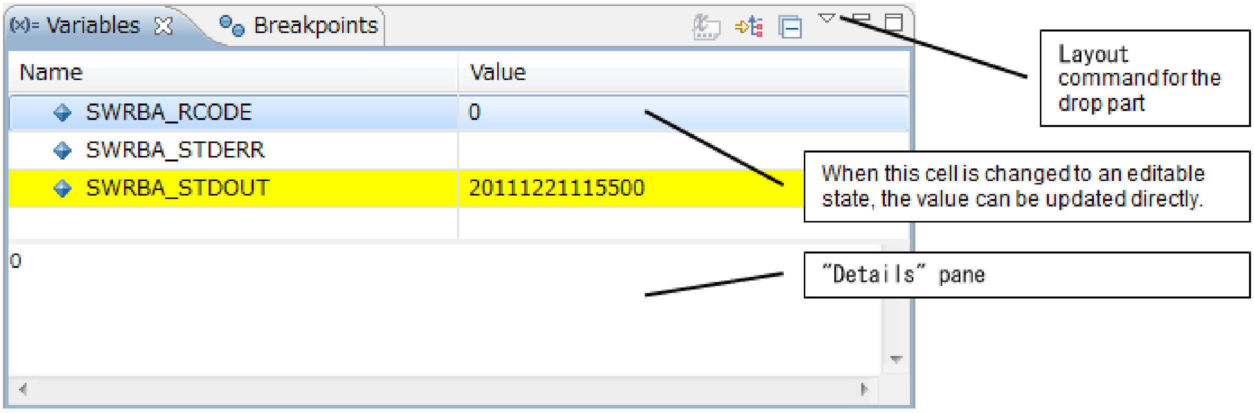
Image/Button	Name	Description	Location of display
	Copy stack	Copies the display string of the selected process instance or node to the Clipboard	Context menu
	Disconnect	Disconnects the selected process instance or debug configuration from debugging	Context menu and Toolbar
	Drop to Frame	Cannot be used in Systemwalker Runbook Automation	Context menu and Toolbar
-	**Edit [Configuration Name]...**	Opens the **Debug Configuration** dialog box of the associated process flow so that changes can be made	Context menu

Image/ Button	Name	Description	Location of display
		 Debug Configurations Create, manage, and run configurations Attach to work flow application type filter text Work Flow Application Proc001 Filter matched 2 of 2 items Name: Debug Proc001 Connect | Source | Common Project: Proc001 Browse... Server Connection Connection Name :: Systemwalker Runbook Automation Server Base URL : : http://rbaserver User name : : swrbaadmin Application: Proc001 Browse... □ Include the public process definitions □ Exclude the started process instances Apply Revert Debug Close	
-	**Edit Step Filter...**	Cannot be used in Systemwalker Runbook Automation	Context menu
	Edit Source Lookup...	Opens the **Edit Source Lookup Path** dialog box so that changes can be made; when **OK** is clicked, the search of the process instance process definition file is executed according to the range/order specified in this dialog box  Edit Source Lookup Path Edit the path used to locate source files. Source Lookup Path: Default Add... Edit... Remove Up Down Restore Default □ Search for duplicate source files on the path OK Cancel	Context menu
-	**Find...**	Opens the **Find** dialog box; clicking **OK** will select the tree of the selected element on the **Debug** view	Context menu

Image/ Button	Name	Description	Location of display
		 Find Specify an element to select (? = any character, * = any String): AND1:912 Debug Proc001 [Work Flow Application] Obtain current date1 Obtain current date1:908 OR1 Start Test_Wait_Debug:864 1.0 [Proc001] Test_Wait_Debug:905 1.0 [Proc001/Test_Wait_Debug] 	
-	**Look Up Source**	Source lookup can be executed forcibly	Context menu
-	**Properties**	Displays the selected element properties; in Systemwalker Runbook Automation, no information is displayed for the properties	Context menu
✖	**Remove All Terminated**	Clears all completed debugging targets from the view display	Context menu and Toolbar
▶	**Resume**	Resumes stopped node; cannot be used on a completed node	Context menu and **Run** menu
↴	**Step Into**	Cannot be used in Systemwalker Runbook Automation	Context menu, **Run** menu, and Toolbar
↷	**Step Over**	Executes stopped node, and pauses automatically at the next node; cannot be used on a completed node	Context menu and Toolbar
↱	**Step Return**	Cannot be used in Systemwalker Runbook Automation	Context menu and Toolbar
⏸	**Suspend**	Suspends the running node so that code reference or change, data inspection, and steps can be executed	Context menu, **Run** menu, and Toolbar

Image/ Button	Name	Description	Location of display
🟥	**Terminate**	Cannot be used in Systemwalker Runbook Automation	Context menu and action display
🟥	**Terminate/ Disconnect All**	Cannot be used in Systemwalker Runbook Automation	Context menu
🟥▶	**Terminate and Relaunch**	Cannot be used in Systemwalker Runbook Automation	Context menu
🟥✖	**Terminate and Remove**	Cannot be used in Systemwalker Runbook Automation	Context menu
⇉	**Use Step Filters**	Cannot be used in Systemwalker Runbook Automation	Context menu and Toolbar
-	**Web console**	Opens the Systemwalker Runbook Automation Web console	Context menu
-	**Show Task**	Opens the Task held by the node	Context menu
-	**Start Instance**	Starts the process instance	Context menu
-	**Re-execute node**	Re-executes an optional node that ended normally	Context menu
-	**Start Instance by Step Motion**	Starts the process instance by step motion	Context menu
-	**Abort instance**	Shuts down the process instance	Context menu
-	**Remove instance**	Clears the stopped or disconnected process instance	Context menu

13.2.4 Variables View



Name	Value
SWRBA_RCODE	0
SWRBA_STDERR	
SWRBA_STDOUT	20111221115500

0

Layout command for the drop part

When this cell is changed to an editable state, the value can be updated directly.

"Details" pane

The **Variables** view displays information about the variables which are associated with the stack selected in the **Debug** view.

Icons for different variables displayed in the **Variables** view are shown below.

- ◆ These are the variables (UDA).
- ◆ This is the operation component input.
- ◆ This is the operation component output result.

These variables are displayed in the order shown for the three groups below. They are displayed in the ascending order and are not case-sensitive.

1. Variables starting with SWRBA (UDA)
2. All other variables (UDA)
3. Operation component variables (input and output results, display name is **Node name: name**)

Variables (UDA) modified between the start of a debug process or restart of the debug process following an interruption and the next interruption will have an emphasized display in the **Variables** view. Moreover, variable (UDA) values modified by editing view operations also have an emphasized display.

Note

The color of the emphasized display for modified variables (UDA) can be changed using the following procedure.

1. From the Windows menu, select **Settings**.
2. Select **Run/Debug** from the tree to the left of the **Settings** dialog box.
3. When selecting **Show Columns**, specify the color using the **Changed value background color** button.
4. If not selecting **Show Columns**, specify the color using the **Changed value color** button.
5. Note that **Show Columns** can be selected from the layout operations in the **Variables** view.

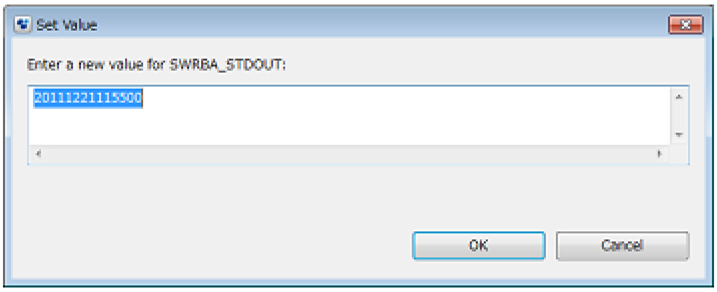
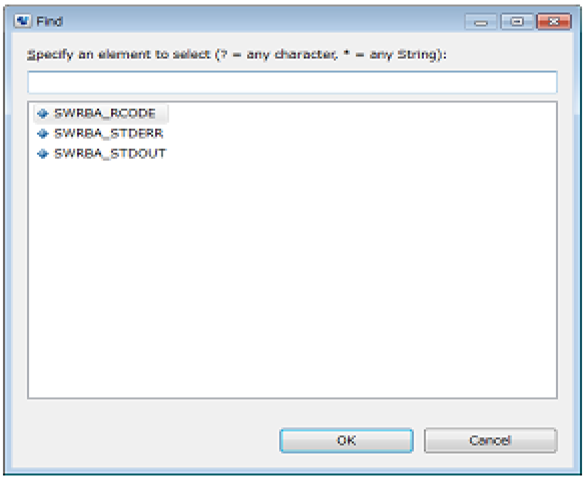
Note

- Variable (UDA) always shows the current value.
- Special characters, such as \, tab, carriage returns, or line breaks, included in the displayed values appear as substitute characters. Special characters are displayed normally, as regular characters without any modification, in the **Details** pane.
- The variable (UDA) cannot be changed if the process has finished.
- If the operations component node is currently interrupted, some of the operations component variables may not display properly.

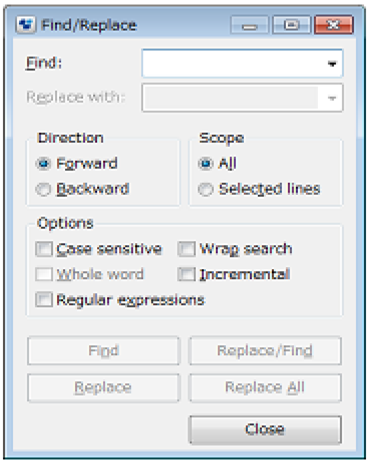
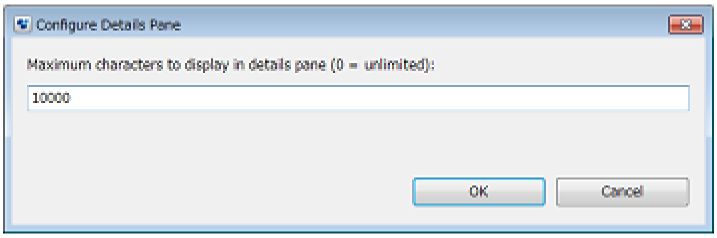
- Variables specific to the operations component node cannot be changed.

Available Operations

In the **Variables** view, the following operations can be used.

Image/ Button	Name	Description	Location of display
	Change value...	The value for the selected variable which will be the basis can be changed using the dialog box.  Set Value Enter a new value for SWRBA_STDOUT: 20111221115500 When **OK** is clicked, the value for the variable will be updated by the value in the dialog box.	Context menu
	Copy variables	Copies the selected variable to the system Clipboard.	Context menu
-	**Find...**	Opens the Find dialog box to search elements in the **Variables** view.  Find Specify an element to select (? = any character, * = any String): SWRBA_RCODE SWRBA_STDERR SWRBA_STDOUT Clicking **OK** will select the tree of the selected element on the **Variables** view.	Context menu
-	**Select All**	Selects all the variables in the view.	Context menu
	Show Type Names	This cannot be used in Systemwalker Runbook Automation.	Toolbar
	Show Logical Structure	This cannot be used in Systemwalker Runbook Automation.	Toolbar
	Collapse All	This cannot be used in Systemwalker Runbook Automation.	Toolbar

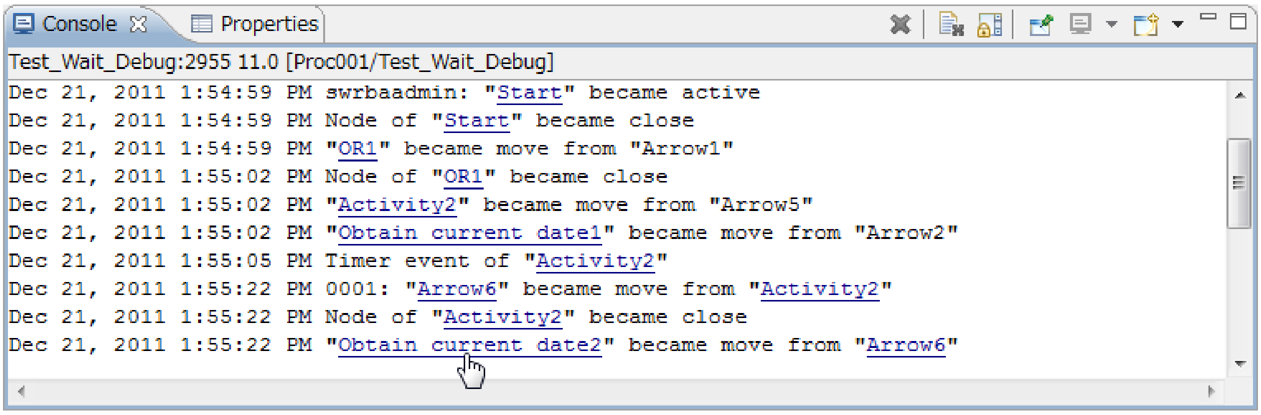
In the **Details** pane, the following operations can be used.

Image/ Button	Name	Description	Location of display
-	**Assign Value**	Evaluates the **Details** pane string and assigns a new value for the currently selected variable.	Context menu
	Content Assist	Opens the **Content Assistance** pop-up dialog box which provides code complete help.	Context menu
-	**Cut**	Copies the selected element to the system Clipboard and deletes it from the **Details** pane.	Context menu
-	**Copy**	Copies the selected element it from the **Details** pane to the system Clipboard.	Context menu
-	**Find/Replace...**	Searches or replaces the specified expression.  Find/Replace Find: Replace with: Direction ● Forward ○ Backward Scope ● All ○ Selected lines Options □ Case sensitive □ Wrap search □ Whole word □ Incremental □ Regular expressions Find Replace/Find Replace Replace All Close When the respective operation button is clicked, the search string is located and changed to a selected state, or replaced. In Systemwalker Runbook Automation **Scope** is not used.	Context menu
-	**Max Length...**	Displays a dialog box for changing the max value for the number of characters displayed in the **Details** pane.  Maximum characters to display in details pane (0 = unlimited): 10000 When **OK** is clicked, the string will be able to be displayed up to the max specified in the dialog box.	Context menu
-	**Paste**	Pastes the system Clipboard content to the **Details** pane.	Context menu
-	**Select All**	Selects all the strings in the **Details** pane.	Context menu
-	**Wrap Text**	Specifies whether to wrap text displayed in the **Details** pane at a predefined width.	Context menu

Using the view layout operation, it is possible to specify the layout and position of the **Details** pane.

Image/ Button	Name	Description	Location of display
▤	**Show Columns**	Specifies whether to display columns in the view. This is valid for the **Variables** view only.	Toolbar
-	**Select Columns...**	When columns are displayed, selects which columns will be displayed.	Toolbar
▥	**Horizontal View Orientation**	Displays the **Details** pane on the right of the view with horizontal sorting available.	Toolbar
▤	**Vertical View Orientation**	Displays the **Details** pane at the bottom of the view with vertical sorting available.	Toolbar
☰	**Variables View only**	Hides the **Details** pane.	Toolbar

13.2.5 Console View



```
Test_Wait_Debug:2955 11.0 [Proc001/Test_Wait_Debug]
Dec 21, 2011 1:54:59 PM swrbaadmin: "Start" became active
Dec 21, 2011 1:54:59 PM Node of "Start" became close
Dec 21, 2011 1:54:59 PM "OR1" became move from "Arrow1"
Dec 21, 2011 1:55:02 PM Node of "OR1" became close
Dec 21, 2011 1:55:02 PM "Activity2" became move from "Arrow5"
Dec 21, 2011 1:55:02 PM "Obtain current date1" became move from "Arrow2"
Dec 21, 2011 1:55:05 PM Timer event of "Activity2"
Dec 21, 2011 1:55:22 PM 0001: "Arrow6" became move from "Activity2"
Dec 21, 2011 1:55:22 PM Node of "Activity2" became close
Dec 21, 2011 1:55:22 PM "Obtain current date2" became move from "Arrow6"
```

In the **Console** view, the process instance execution history is displayed.

The execution history of the process instance or node selected in the **Debug** view will be displayed.

Click the node name displayed in the **Console** to open the Process Definition (editor) in the workbench (if it is already open, the editor moves to the front).

Note

If the execution history contains an error message, the message will be displayed with standard error color.
Underlined node names are displayed in the hyperlink text color.
In Console view there is no limitation of message length by default.

To change the standard error color or limit the message length perform the following steps

1. Select **Preferences** from **Window** menu.
2. Select **Run/Debug** >> **Console** from the tree in **Preferences** dialog.
3. To change the color of standard error click on **Standard Error Text Color**.
4. To change the length limit of messages check **Fixed width console** and click on **Maximum character width**.

To change the color of hyperlink text, perform the following steps:

1. Select Window >> Preferences.
2. Select **General** >> **Appearance** >> **Colors and Fonts** from the left tree in the **Preferences** dialog box.

3. Click on **Hyperlink text color** to specify the color.

Note

The following options from **Run/Debug** >> **Console** of **Preferences** dialog are not effective on Systemwalker Runbook Automation **Console**

- Show when program writes to standard out
- Show when program writes to standard error
- Standard In text color

Operations that can be used

In the **Console** view, the following operations can be used.

Image/ Button	Name	Description	Location of display
✖	**Close Console**	Closes the currently active Console. To open a closed Console, use the **Open Console** command.	Toolbar
	Clear Console	Clears the string displayed on the currently active Console.	Context menu and Toolbar
	Display Selected Console	This cannot be used in Systemwalker Runbook Automation.	Toolbar
	Open Console	Opens a new Console. To open the Console, open the drop-down menu on the right of the button, then select the Console to be opened.	Toolbar
	Pin Console	Pins the current Console, so that it will remain in front of all the other Consoles.	Toolbar
	Scroll Lock	Specify whether the scroll lock can be used in the current Console.	Context menu and Toolbar
	Copy	Copies the selected element from the Console to the system Clipboard.	Context menu
	Cut	This cannot be used in Systemwalker Runbook Automation.	Context menu
-	**Find/replace...**	Search can be executed using the specified expression. **Replace** cannot be used in Systemwalker Runbook Automation.	Context menu
	Paste	This cannot be used in Systemwalker Runbook Automation.	Context menu
-	**Open Link**	This cannot be used in Systemwalker Runbook Automation.	Context menu

13.3 Debugging Flow

The flow for the debugging functions provided is explained below.

13.3.1 Preparing to debug

Following is an outline of tasks necessary to prepare for debugging

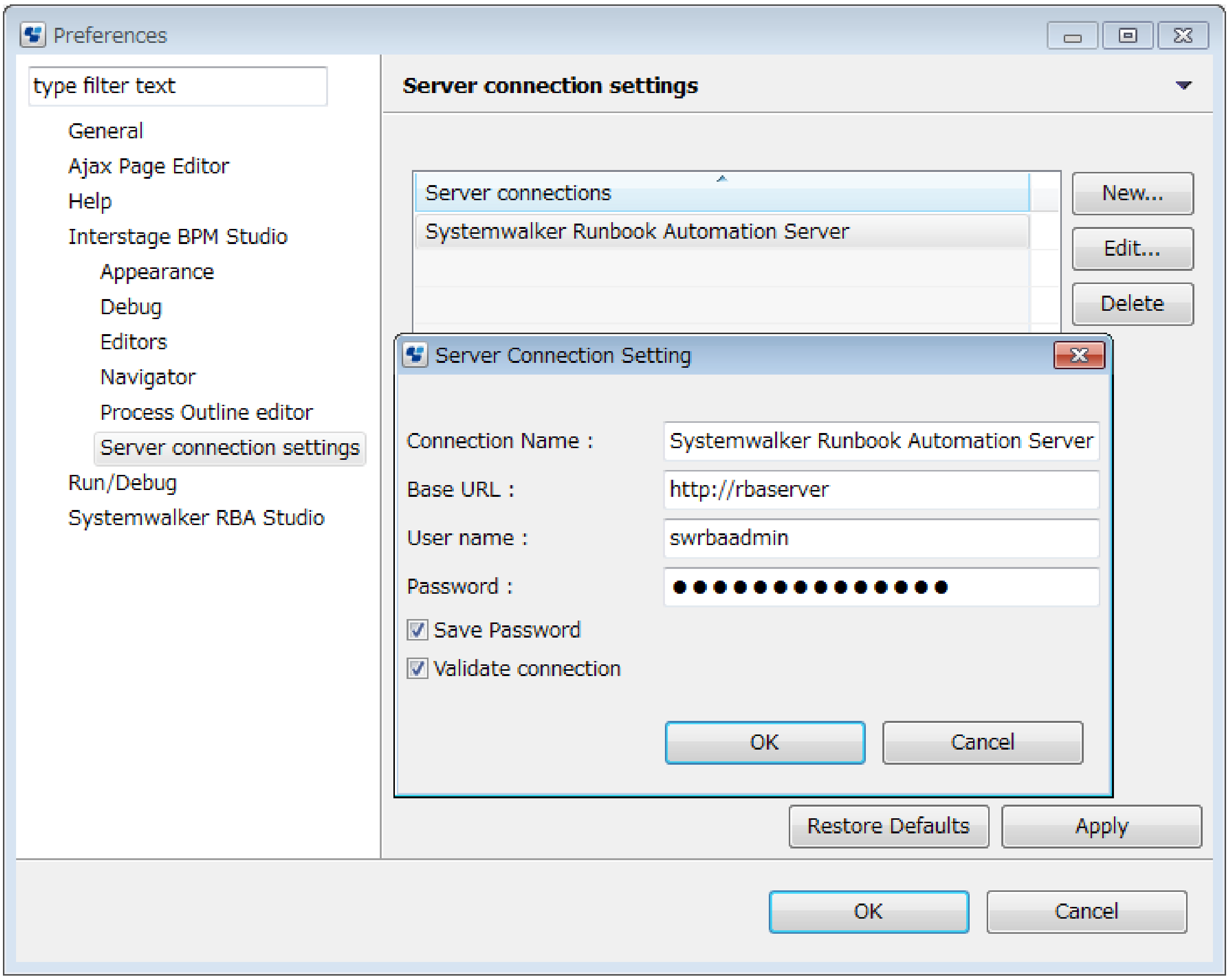
1. Set up Server connection setting
2. Prepare process definition

3. Create debug configuration

Set up Server connection setting

Use the following procedure for the debugging flow.

1. From the menu items **Window** select **Preferences**. Preferences window will be displayed
2. Select **Server connection settings** from **Interstage BPM Studio** tree and define the Management Server to be used.



Systemwalker Runbook Automation Server

Connection Name : Systemwalker Runbook Automation Server

Base URL : http://rbaserver

User name : swrbaadmin

Password : ●●●●●●●●●●●●●●

☑ Save Password

☑ Validate connection

- Connection name: Specify a name to identify the connection. Any name can be specified.
- Base URL: This is the URL used to connect to the Management Server. Specify the base URL using the following format:
  http://<hostname>:<port>
  In the URL example below, "rbaserver" is used for the hostname and "80" is used as the port.
  http://rbaserver:80
- User name: Specify the name of the user with administrator privileges that can execute operations such as the generation of process instances.
- Password, save password: To save a password so that it does not need to be entered at the time of connection, specify the password in this dialog box then select **Save Password**. If the password is not saved, it will need to be entered each time there is a connection.
- **Validate connection**: select to verify the connection when **OK** is clicked.

For details about server connection settings, refer to "2.4.1 Server Connection Settings" in Chapter 2, "Basic Information and Customization for the User Interface".

Prepare process definition

Make the process definition on the workspace identical to the process definition on the Management Server.

- If the debugging target process definition on the workspace is new, or there is no process definition on the Management Server, upload a process definition to the Management Server. Refer to "3.1.15 Uploading Workflow Application Projects to a Server" for information on how to upload the process definition.
- If there is no debugging target process definition on the workspace, or the process definition is old, download a process definition from the Management Server. Refer to "3.1.14 Downloading Workflow Application Projects from a Server" for details.

Make sure that process definition has the following attributes

- Status:
  Draft
- Owner:
  Is the user set in "Set up Server connection setting".

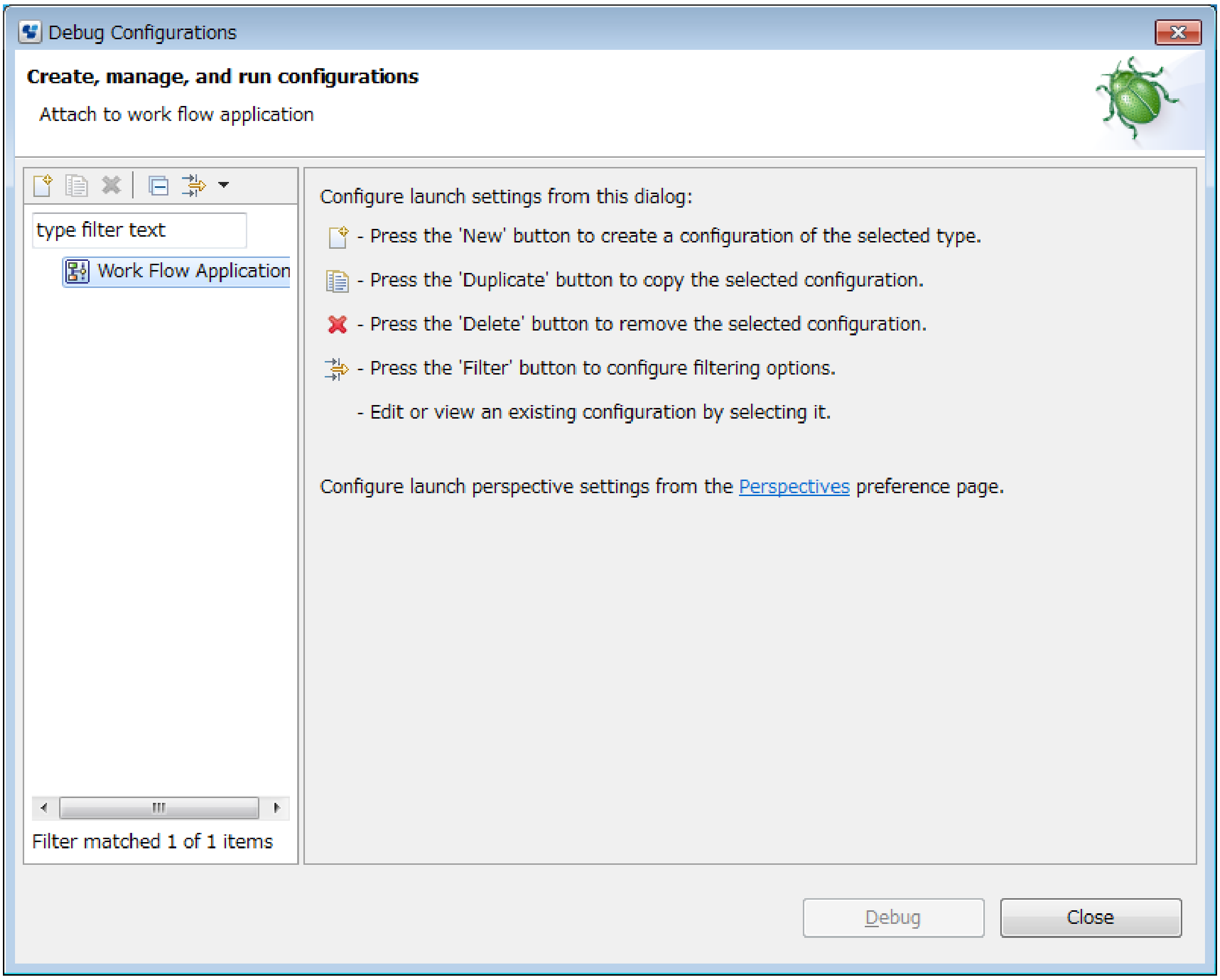
Note

To debug the process definition, the process definition name and process definition file name must match.

Create debug configuration

1. Open the menu items **Run** >> **Debug Configurations**.
   This can also be opened by clicking the ▼on ✱ (the **Debug** button) on the toolbar, then selecting the **Debug Configurations** menu.

2. While **Work Flow Application** is selected on the tree, click the **New launch Configuration** button from the top part of the tree (this button is on the very left). Alternatively, while **Work Flow Application** is selected on the tree, open the context menu then select the **New** menu.



Create, manage, and run configurations

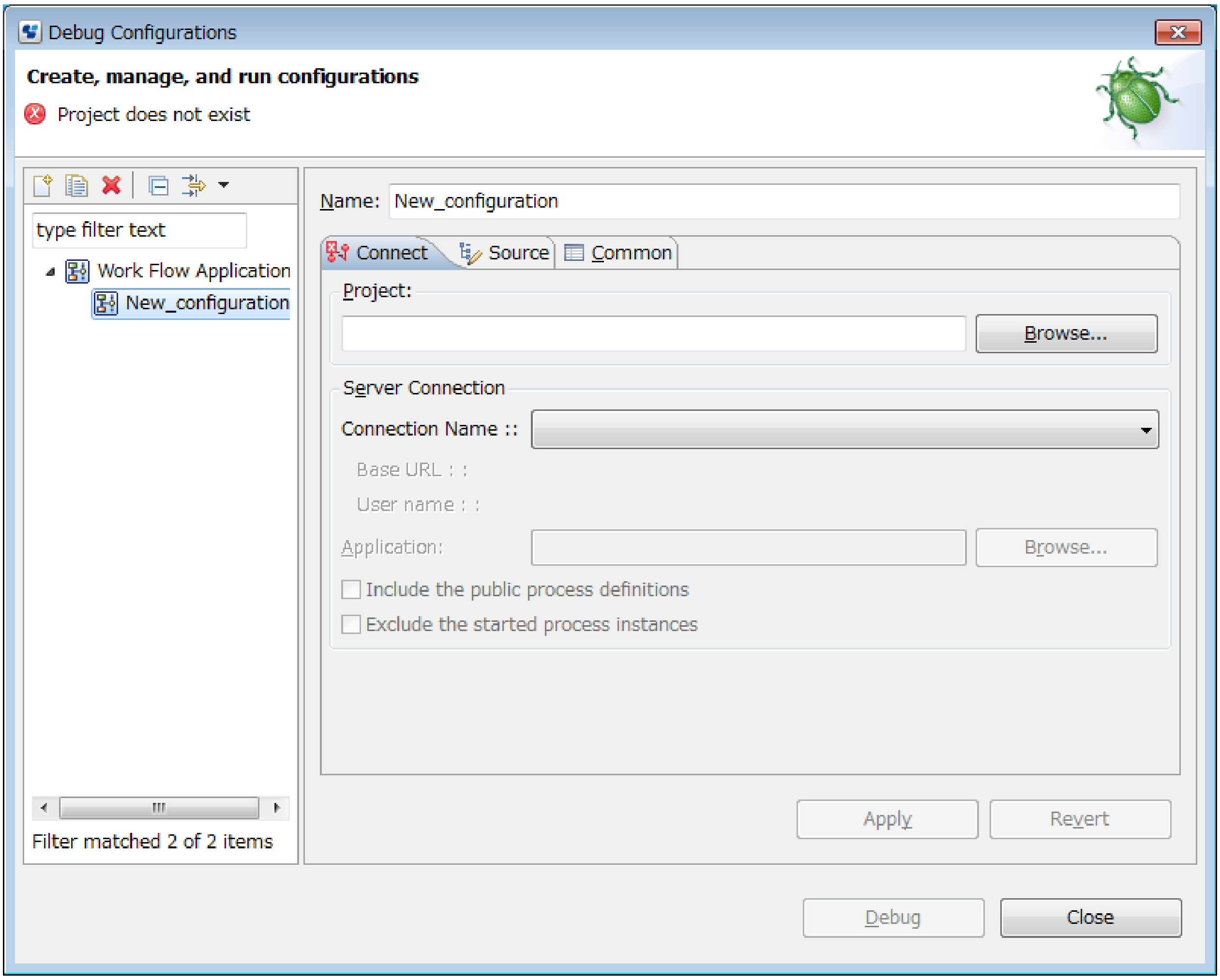
Attach to work flow application

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

3. The new debug configuration used in **Work Flow Application** debug configuration is displayed to the right of the dialog box.



Debug Configurations

Create, manage, and run configurations

Project does not exist

type filter text

- Work Flow Application
  - New_configuration

Filter matched 2 of 2 items

Name: New_configuration

Project:

Server Connection

Connection Name ::

Base URL : :

User name : :

Application:

□ Include the public process definitions

□ Exclude the started process instances

4. In the **Name** field, specify the name which will be attached to the debug configuration. This name must be unique, and should be easy to remember.

5. In the **Project** group, specify the project containing the process definition to be debugged.
   This can also be selected and specified using the **Browse** button.

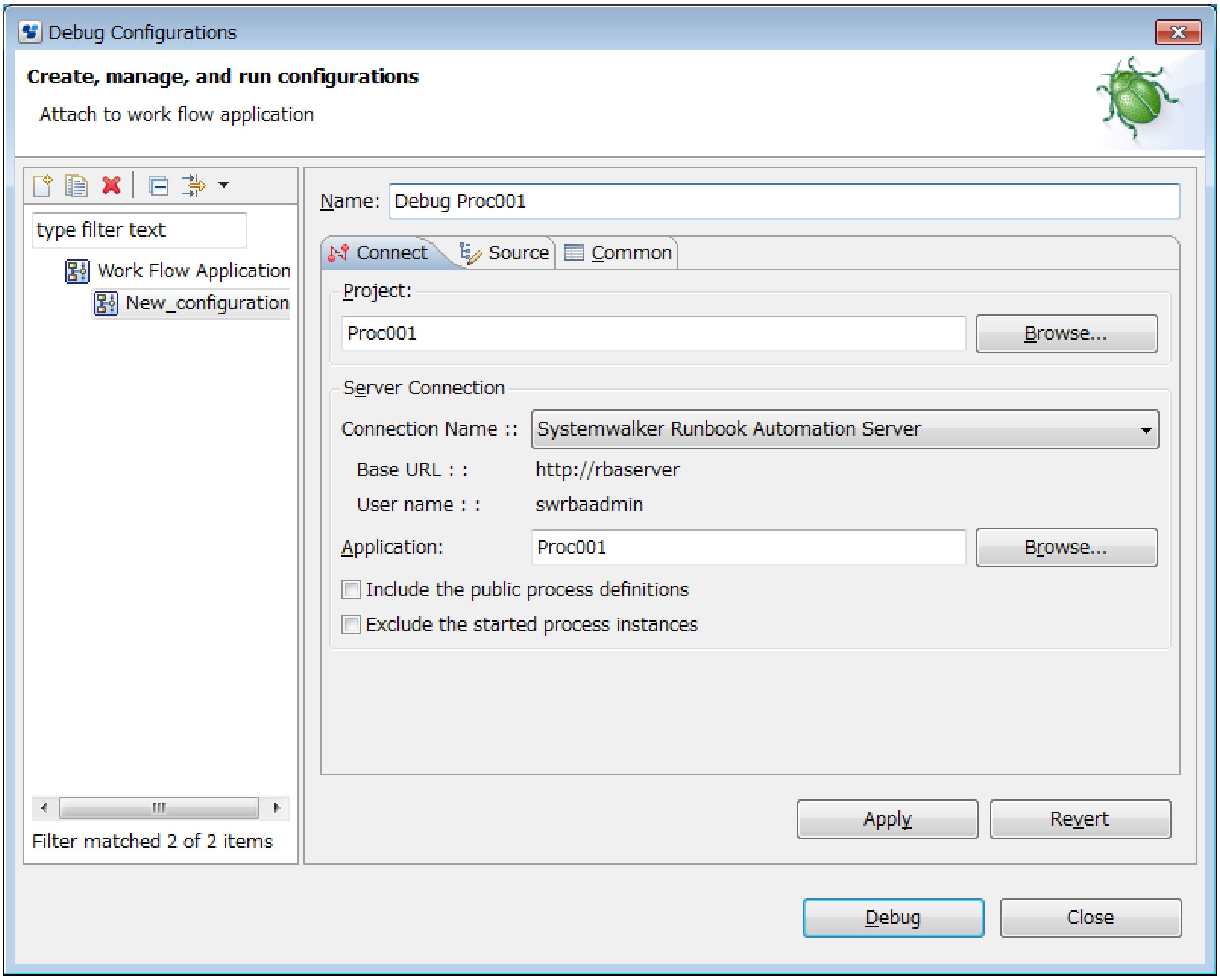
6. From the **Connection Name** combo-box, select the connection name which was defined in "Set up Server connection setting".

   Application field refers to application on management server containing the process definition to be debugged. Selecting **Application** is optional. Press **Browse** to select Application.

Point

If **Application** is not set, searching for process definitions may take time since all the process definitions on management server will be searched. To narrow down the search and shortening the search time you can set the **Application**. If Application is set, searching for only process definitions which is contained in Application. Included. You can shorten the amount of time to make debug possible by setting Application.

The time until debugging becomes possible can be further reduced by selecting the **Exclude the started process instances** checkbox.



Debug Configurations

Create, manage, and run configurations

Attach to work flow application

Name: Debug Proc001

Project: Proc001

Server Connection

Connection Name :: Systemwalker Runbook Automation Server

Base URL : : http://rbaserver

User name : : swrbaadmin

Application: Proc001

□ Include the public process definitions

□ Exclude the started process instances

7. Click the **Debug** button to connect to the server and start the debugging.
   The **Debug Configurations** dialog box closes, and the workbench window configuration switches to the **Debug** perspective.

13.3.2 Starting to debug

Starting to debug after creating a debug configuration

1. Click on **Debug** button.
   Connection to management server will be established, **Debug Configuration** window will be closed and workbench window configuration will switch to **Debug** perspective.

Starting to debug using an already registered debug configuration

One of the following procedures can be used to start debugging:

Operating from the Menu

1. Open the menu item **Run** >> **Debug Configuration**.

2. From **Work Flow Application** on the tree, select a debug configuration which has already been created.

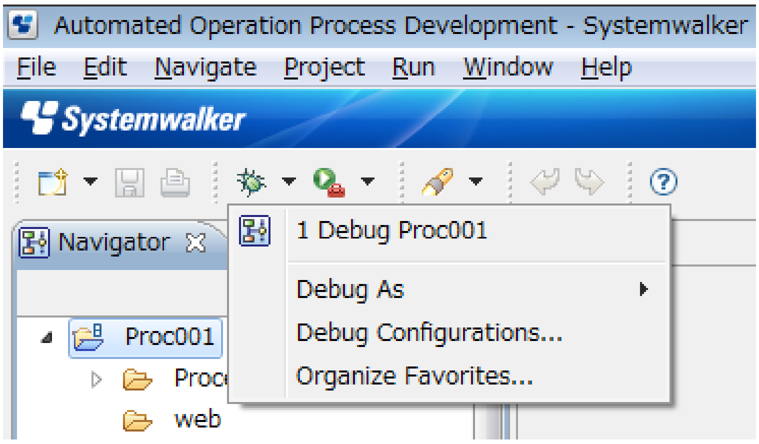
3. Click the **Debug** button to connect to the server and start the debugging.
   The **Debug Configuration** dialog box closes, and the workbench window configuration switches to the **Debug** perspective.

Operating from the Toolbar

1. Open the menu item **Run** >> **Debug Configuration**.

   Click the ▼ on (the **Debug** button) on the toolbar, then select the debug configuration which has already been created.

   The workbench window configuration switches to the **Debug** perspective.

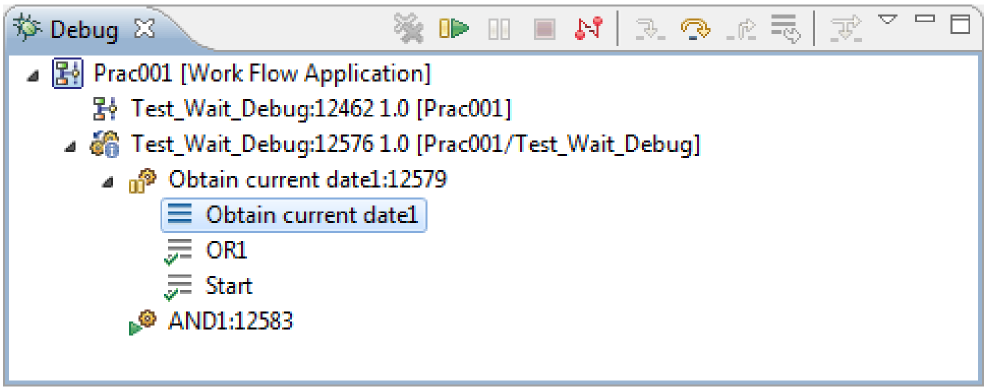


13.3.3 Process Instance Start, Suspend, Step Motion

Debugging after creating a new process instance

1. Select the process definition displayed on the **Debug** view, then select **Start instance by step motion** from context menu.

   A new process instance is generated on the Management Server and displayed on the **Debug** view.


Debug
Prac001 [Work Flow Application]
Test_Wait_Debug:12462 1.0 [Prac001]
Test_Wait_Debug:12576 1.0 [Prac001/Test_Wait_Debug]
Obtain current date1:12579
Obtain current date1
OR1
Start
AND1:12583

2. Node under the new process instance will be shown by Suspend icons. Confirm the status of the node (such as the variable value).

3. Select the node or suspended stack ☰ (element at the beginning) in the **Debug** view.

4. Select the menu item **Run** >> **Step Over** or click (the **Step Over** button) on the toolbar of the **Debug** view to run the process instance node in step mode.

5. Whenever a node completes, the process pauses by **Suspended status** just before running the next node. Continue debugging the process definition by repeating the selection of **Step Over** and confirming the status of the node.

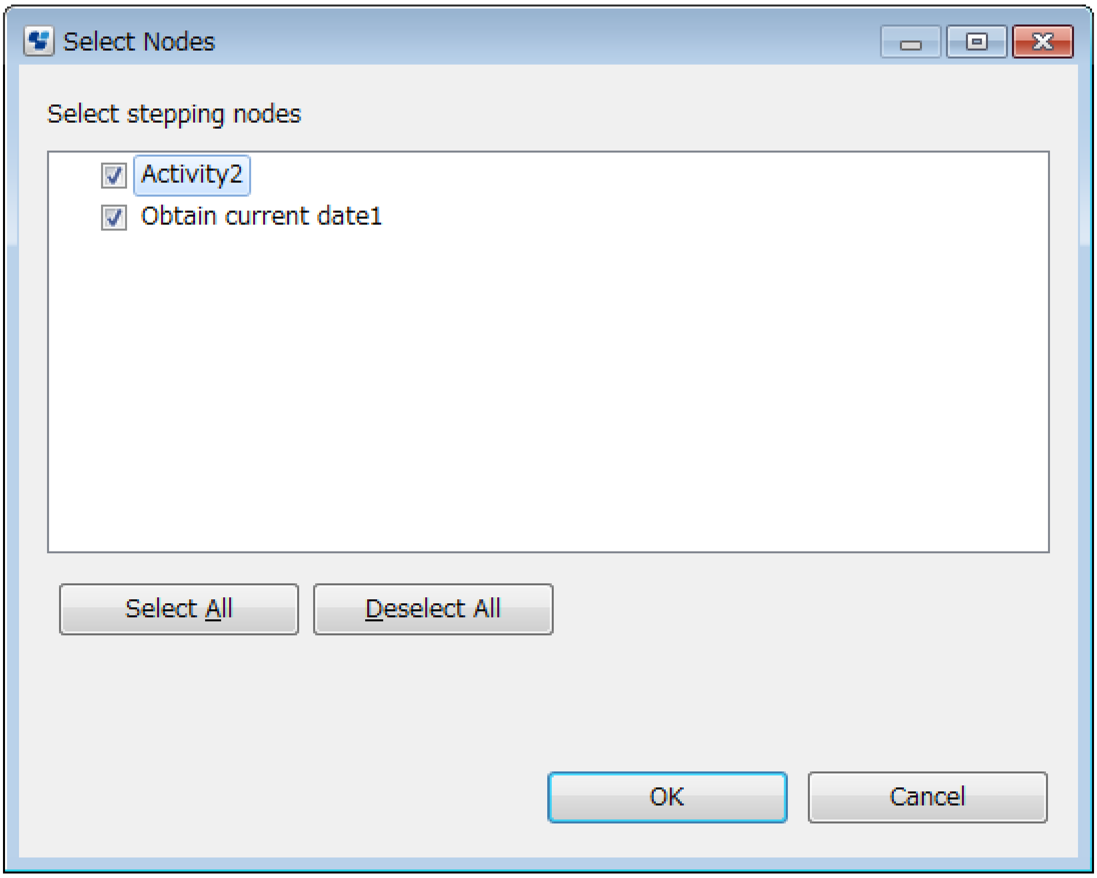
Note

In this version, when the step over is complete, the process pauses at the position of the next node to be executed In branch nodes, such as "OR" nodes, when there is more than one node on the next step, the process interrupts at the position of each node. The process cannot be interrupted at the Start and Exit nodes.

Note

Re-starting or stepping over the nodes that have Tasks, the icon will switch to ▤ or ▤. In this case finish the Task to move to next node. For information on finishing Task refer to "Displaying the Task of an activity which is running". In this version, you cannot debug by creating Start Node Task. You can create an activity after the Start Node with a Task and execute the Task from that activity.

Stepping over can also be performed by selecting a process instance. If there are multiple suspended nodes, select the node from the dialog box and execute the step over.


Select Nodes

Select stepping nodes

☑ Activity2
☑ Obtain current date1

Debugging while a process instance is running

1. Select the process instance displayed in the **Debug** view or the node under it, then either select the menu item **Run** >> **Suspend**, or click ⏸ (the **Suspend** button) of the **Debug** view on the toolbar to suspend the process instance node.

2. After confirming the status of the node, either select the menu item **Run** >> **Step Over** or click the **Step Over** button on the toolbar of the **Debug** view to run the process instance node in step mode.

3. Whenever a node completes using step motion, the process pauses in **Suspended** status just before the next node. Continue debugging the process definition by repeating the selection of **Step Over** and confirming the status of the node.

Note

Suspending the node doesn't make the process to stop right away. In this case the process shifts to the next node then it will be suspended.

You cannot suspend a node if the execution of node has already finished. Make sure to pause only running nodes. In this version process cannot be suspended on "AND" nodes.

Stopping the debug of a process instance which is running, and Aborting the process instance

Select the process instance displayed on the **Debug** view, and then select **Abort Instance** from the context menu.
If the shutdown was successful, a message will be displayed on the node.

Re-executing an optional node that ended normally

It is possible to re-execute nodes that have ended normally, as well as the node types indicated below, for process instances that are currently being executed. To re-execute a node, select the node that ended normally under the process instance displayed in the **Debug** view, and then, select **Re-execute node** from the Context menu. The node will re-execute before being interrupted at the position of the next node to be executed.

- Activity Node
- Voting Activity Node
- Subprocess Node
- Operation component

Note

- When re-executing an optional node that ended normally, the node transition may not occur as intended. This prevents the process instance from ending normally. End the process instance forcibly if it does not end normally.
- If any of the nodes on the route between the Start and interrupted nodes are currently being executed or currently being interrupted, it will not be possible to display the route for the interrupted node correctly.

Disconnecting a process instance which is running

Select the process instance displayed on the **Debug** view, then select **Disconnect** from the context menu.
If the disconnection was successful, "disconnected" will be displayed on the node.

Clearing aborted or disconnected process instances

Select an aborted or disconnected process instance displayed on the **Debug** view, then select **Remove Instance** from the context menu.
The selected process instance will be cleared from the **Debug** view.

Debugging and editing process definition

If during debugging process definition has been changed, debug must be restarted. Following are the steps necessary to restart debugging:

1. Run the process instance to end or abort the process instance.
2. Disconnect the debug configuration in "Debug" window.
3. Upload the changed process definition to management server.
4. Use the process definition to connect to management server and start debugging the new process definition.

Debugging from Web console

The operations below are possible from the Web console for process instances, nodes, and tasks that have not been debugged using Studio:

- Pause
- Abort
- Delete
- Resume
- Activate
- Not Activate

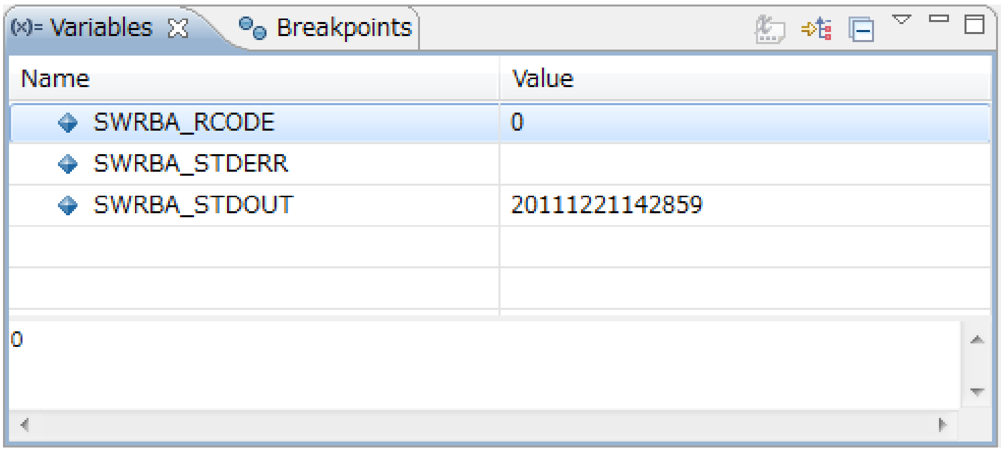
13.3.4 Browsing/Updating Variables

Variables can be browsed or updated when the node is in a paused state.

To browse a variable value, select the stack from the tree displayed on the **Debug** view.

When a paused stack is selected, the list of variables and the values for each variable will be displayed on the **Variables** view.

The output results particular to a node can also be shown in the **Variables** view by selecting a stack that has completed.



Name	Value
SWRBA_RCODE	0
SWRBA_STDERR	
SWRBA_STDOUT	20111221142859

0

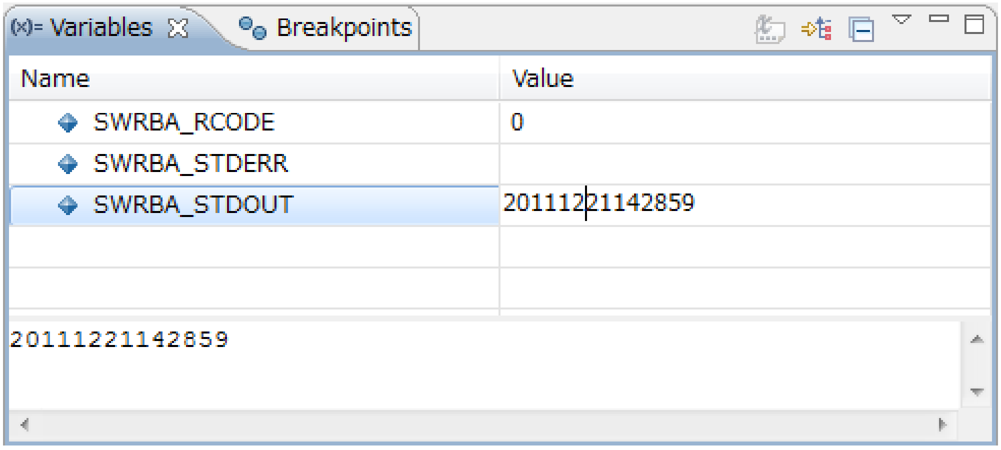
You can change the variable directly in the view or by invoking the dialog box.

If the variable was changed successfully, the corresponding **Value** field on the view will change.

Note

The output results specific to a node cannot be changed.

Changing the variable from the view directly



Name	Value
SWRBA_RCODE	0
SWRBA_STDERR	
SWRBA_STDOUT	20111221142859

20111221142859

When the corresponding **Value** field on the **Variables** view is clicked and the status is changed to editable (where | which indicates 'input', is displayed), overwrite the value directly then press the Enter key.

Note

When overwriting values on the **Variables** view, type substitute characters for the special characters listed below.

- For \, type \\.
- For carriage returns, type \r.

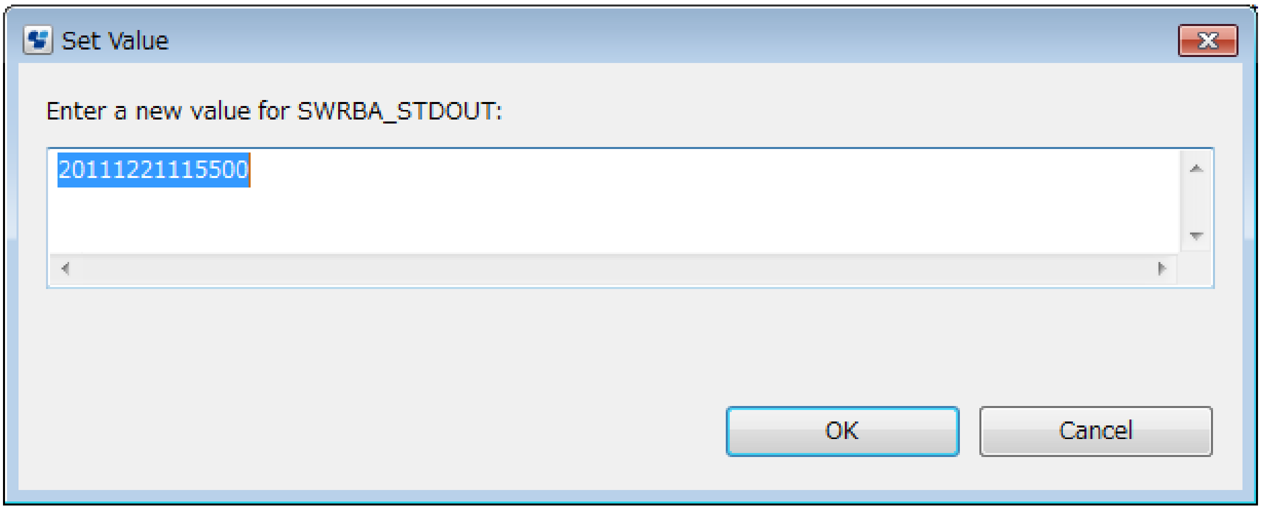
- For line breaks, type \n.
- For tabs, type \t.

Use the following procedure to change from the **Details** pane:

1. Update the value displayed on the **Details** pane.
2. Open the context menu on the **Details** pane, then select **Assign Value** menu.

Displaying the dialog box then changing the variable

Select the corresponding variable name on the **Variables** view, then select the **Change Value**... menu from the context menu.



Enter a new value for SWRBA_STDOUT:

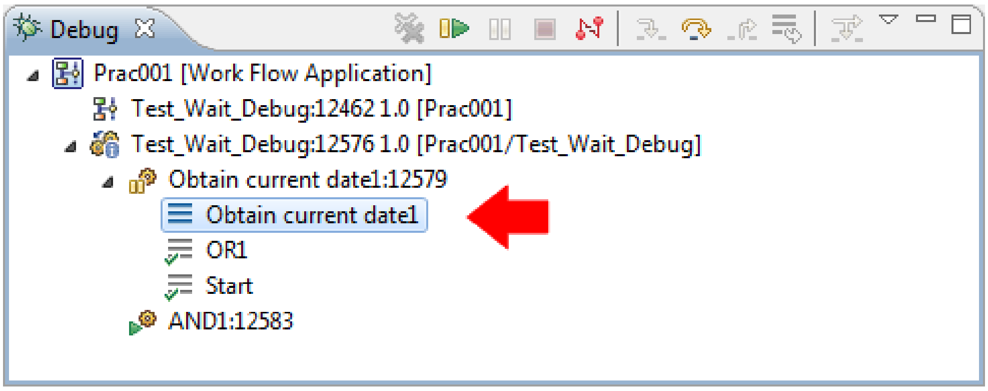
20111221115500

Overwrite the value in the displayed dialog box then click the **OK** button.

13.3.5 Jumping to the Process Definition

The process definition (Editor) can be called from the suspended location.

Select the stack displayed under the suspended node from the tree displayed on the **Debug** view.


Debug
Prac001 [Work Flow Application]
Test_Wait_Debug:12462 1.0 [Prac001]
Test_Wait_Debug:12576 1.0 [Prac001/Test_Wait_Debug]
Obtain current date1:12579
Obtain current date1
OR1
Start
AND1:12583

When the suspended stack is selected, the process definition (Editor) in the workbench will open (if it is already open, the Editor will be displayed in front) and the location of the node corresponding to the stack will be selected.

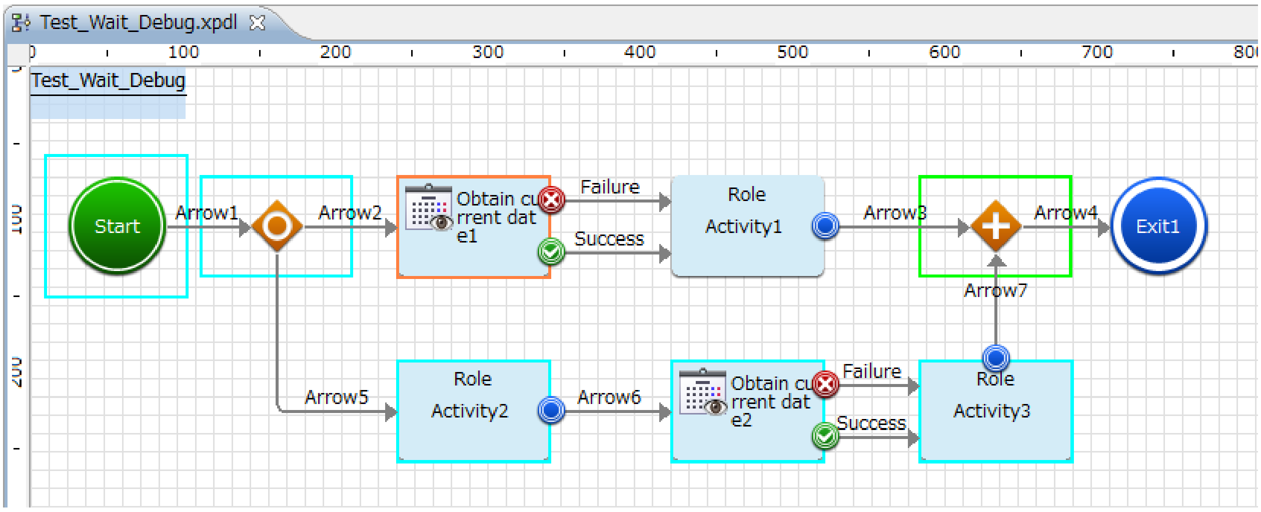
Note

Using node names in process definition you have to select the nodes. You cannot choose multiple nodes with the same name. Make sure to assign unique names to nodes.

Process definitions cannot be opened when a stack is selected from the tree of finished process instances displayed in the **Debug** view. Moreover, it is not possible to select nodes that correspond to those in the open process definitions.

Further, by clicking the node name displayed in the console showing the execution history of process instances, the process definition (editor) is opened in the workbench and the node location can be selected.

The status of nodes can also be emphasized for display in the opened process definition.



Point

To emphasize the display of node statuses, select the **Window** >> **Preferences** menu of the workbench, then select **Interstage BPM Studio** >> **Debug** >> **Display the highlighted execute node**.

To change the colors of the statuses of the nodes, change the colors corresponding to the statuses.

Note

The emphasized display on process definitions is removed when process instances end.

13.3.6 Other Operations

Starting the Systemwalker Runbook Automation Web console

The Systemwalker Runbook Automation Web console can be opened from the **Debug** view.

To open the Web console, select debug configuration node displayed on the **Debug** view and click on **Web console** from the context menu.

Point

The Web console opens in an external browser of the workbench. For confirming external browser, select the **Window** >> **Preferences** menu of the workbench, select **General** >> **Web Browser** of the displayed settings dialog box. The selected browser in **External Web Browser** will be used.

Displaying the Task of an activity which is running

The Task of an activity which is running can be opened from the **Debug** view.

To open Task, select **Show Task** from the context menu while the running activity (node with the or icon) is selected in the **Debug** view. If multiple tasks are associated with the activity, a dialog box for selecting tasks will be displayed. To identify tasks they are listed as [Assignee: Task ID], select the required task and the task will be displayed. A task can be performed by selecting on the possible choices in **Make Choice** section of **Details** tab of the task window. Close the browser after making choices for the task.

Point

Show Task opens in the external browser of the workbench. For confirming external browser, select the **Window** >> **Preferences** menu of the workbench, select **General** >> **Web Browser** of the displayed settings dialog box. The selected browser in **External Web Browser** will be used.

When **Show Task** is selected, the external browser opens and the page for connecting to task is displayed. If you have allowed browser to execute JavaScript, transition to the task server will occur automatically. If you have not allowed the execution of JavaScript, use one of the following methods to move to task window.

- Select the permissions for the enquiry displayed on the browser as to whether the execution of JavaScript is allowed.
- Click the **Connect** button displayed in the connection page.

Note

Before opening tasks close all the open external browsers. If there are open external browser login to server may fail.

The "To display the webpage again, Internet Explorer needs to resend the information you've previously submitted." message may be displayed. In this case click on **Retry** button.

13.4 Debugging techniques

Techniques to effective debugging are explained bellow.

13.4.1 Running specific node using conditional branches

Debugging process definitions with branched nodes such as conditional nodes, you can select to **Start instance by step motion** and browse/update variables to run a specific node. Take the following steps to run a specific node

1. Start the debugging by selecting **Start instance by step motion**.
2. Step over the nodes before the branch node.
3. Select the stack and open the process definition. Confirm the branch properties
   For conditional branches confirm the condition.
4. Confirm the branch variables in **Variables** view.
5. Change the variables in **Variables** view so that the condition of arrow to run the specific node is satisfied.
6. Step over to next node.

13.4.2 Opening process definition while debugging

To open process definition while debugging in Debug perspective take the following steps.

1. Select **Show view** >> **Other**... from **Window** menu.
2. Select **Navigator** under the **Interstage BPM Studio** tree.
3. From **Navigator** view select the required process definition file and select **Open** in the context menu or double click on the process definition file.

Chapter 14 Maintenance for Systemwalker Runbook Automation Studio

This chapter explains the following three items regarding the maintenance for Systemwalker Runbook Automation Studio.

- Exporting Settings
- Importing Settings
- Collecting Maintenance Information

Point

- Settings information includes definition information on the palette in the Process Definition Editor.
- If you reuse work space directory after Systemwalker Runbook Automation Studio has been reinstalled, export setting information before Studio has been uninstalled Import setting information which exported, after Studio has been reinstalled.
- Note that you can not reuse work space directory which made in previous version(14.0)
- Maintenance information includes various log files and configuration files that are required for investigating errors when they occur.

14.1 Exporting Settings

To export settings for Systemwalker Runbook Automation Studio to a file, use the swrbaexport command (the settings export command).

You must specify the output file.

Example

To export the settings to C:\work\swrbaconfig.zip:

```
swrbaexport C:\work\swrbaconfig.zip
```

Refer to the *Systemwalker Runbook Automation Reference Guide* for details.

14.2 Importing Settings

To import settings for Systemwalker Runbook Automation Studio, use the swrbaimport command (the settings import command).

You must specify the source file containing the exported settings.

Example

To import the settings from C:\work\swrbaconfig.zip

```
swrbaimport C:\work\swrbaconfig.zip
```

Refer to the *Systemwalker Runbook Automation Reference Guide* for details.

14.3 Collecting Maintenance Information

To collect maintenance information for Systemwalker Runbook Automation Studio when a problem occurs, use the swrbacolinf command (the maintenance information collection command).

You must specify the folder where the maintenance information file is to be output - the file will be named swrbacolinf*yyyy-MM-dd-HH-mm-ss*.dat.

Example

To collect the maintenance information in C:\work

```
swrbacolinf C:\work
```

Refer to the *Systemwalker Runbook Automation Reference Guide* for details.

Appendix A Supported JavaScript Functions

Systemwalker Runbook Automation provides a set of JavaScript functions that can be used with Java Actions, triggers, Complex Conditional Nodes, and Sequential Loop Nodes. This appendix covers these functions in detail.

Apart from using functions listed in this appendix, you can use the JavaScript functionality defined in the ECMA Standard. For information about the ECMA Standard, refer to *ecma-262.pdf* provided with the Systemwalker Runbook Automation installation file.

Note

The size of a method used in JavaScript is limited by the Java Virtual Machine (JVM). Currently, the method byte code size is limited to 64 KB. If you use a method with a larger size, the JVM throws an error, requiring you to reduce the size of the method before executing the JavaScript again.

A.1 General JavaScript Functions

You can use the JavaScript functions explained in this section with Java Actions, triggers, Complex Conditional Nodes, and Sequential Loop Nodes.

Note

As numeric values are treated as values of the Number type, you can only use the following range of integers in JavaScript:

- Minimum: -9007199254740992
- Maximum : 9007199254740992

Note that the following minimum and maximum values of the Long type are not supported:

- Minimum: Packages.java.lang.Long.MIN_VALUE
- Maximum: Packages.java.lang.Long.MAX_VALUE

new Packages.java.util.Date()

For a description, refer to the Javadoc provided with the J2SE Development Kit (JDK).

Date DateAdd(Date or Number date, Number offset, String field)

Returns a JavaScript Date object containing a date and time that is the result of adding the offset to the date

The String field determines the unit of time for the offset. Valid field values are:

Field Value	Description
"ss"	Seconds
"mi"	Minutes
"hh"	Hours
"dd"	Days

Example:

```
var now = Packages.java.util.Date();
uda.Date = DateAdd (now, 1, "dd");
```

Note

In the above example, the User Defined Attribute (UDA), Date, is of the DATE type.

If now has the following value:

Tue Jul 01 2006 14:02:59 GMT-0800 (PST)

Then, tomorrow (DateAdd (now, 1, "dd")) has the following value:

Wed Jul 02 2006 14:02:59 GMT-0800 (PST)

If date is of the Number type, its value is interpreted in milliseconds since January 1, 1970.

Boolean DateCompare(Date or Number date1, String operator, Date or Number date2)

Compares two Date values, and returns TRUE or FALSE as the result of the comparison

Valid operators are:

Operator	Description
">"	Greater than
"<"	Less than
">="	Greater than or equal to
"<="	Less than or equal to
"=="	Equal to
"!="	Not equal to

Example:

```
var now = Packages.java.util.Date();
var tomorrow = DateAdd (now, 1, "dd");
if (DateCompare (now,"<", tomorrow)) ...; //... Executes because DateCompare evaluates to true.
```

If date1 and date2 are of the Number type, their values are interpreted in milliseconds since January 1, 1970.

Number DateDiff(Date or Number date1, Date or Number date2, String field)

Subtracts date2 from date1, and returns the difference between these date/times in days, hours, minutes, or seconds depending on the field value

Valid field values are:

Field Value	Description
"ss"	Seconds
"mi"	Minutes
"hh"	Hours
"dd"	Days

Example:

```
var now = Packages.java.util.Date();
var tomorrow = DateAdd (now, 1, "dd");
var diff = DateDiff (tomorrow, now, "dd")); //diff has a value of 1.
```

If date1 and date2 are of the Number type, their values are interpreted in milliseconds since January 1, 1970.

Example:

```
var date = DateDiff (20000000,10000000,"ss");
```

BigDecimal DecimalAdd(BigDecimal value1, BigDecimal value2)

Returns a JavaScript BigDecimal object that is the sum of the specified parameters; the result inherits the precision from the parameter with the most significant figures

If you want to assign the result to a UDA, ensure that the UDA is of the BIGDECIMAL type.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal value.

Example:

```
int x = 39;
var z = DecimalAdd ("3.1416",x);
```

If z is of the BigDecimal type, its value is 42.1416.

Boolean DecimalCompare(BigDecimal value1, String operator, BigDecimal value2)

Compares two BigDecimal values, and returns TRUE or FALSE as the result of the comparison

Valid operators are:

Operator	Description
">"	Greater than
"<"	Less than
">="	Greater than or equal to
"<="	Less than or equal to
"=="	Equal to
"!="	Not equal to

If you want to assign the result to a UDA, ensure that the UDA is of the BIGDECIMAL type.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal value.

Example:

```
var smallDecimal = "1.11";
var largeDecimal = "22.22";
if (DecimalCompare (smallDecimal,"<", largeDecimal)) ...; //Executes because DecimalCompare evaluates
to true.
```

BigDecimal DecimalDivide(BigDecimal value1, BigDecimal value2, Number scale)

Divides value1 by value2, and returns the result of the division

The scale field determines the number of significant digits for rounding. Any number can be used as the scale value; default is 2.

The rounding mode is always to round half up; rounds towards the "nearest neighbor" unless both neighbors are equidistant. In that case, it rounds up.

If you want to assign the result to a UDA, ensure that the UDA is of the BIGDECIMAL type.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal value.

BigDecimal DecimalMultiply(BigDecimal value1, BigDecimal value2, Number scale)

Multiplies value1 by value2, and returns the result of the multiplication

The scale field determines the number of significant digits for rounding. Any number can be used as the scale value; default is 2.

The rounding mode is always to round half up; it rounds towards the "nearest neighbor" unless both neighbors are equidistant. In that case, it rounds up.

If you want to assign the result to a UDA, ensure that the UDA is of the BIGDECIMAL type.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal value.

BigDecimal DecimalSubtract(BigDecimal value1, BigDecimal value2)

Subtracts value2 from value1, and returns the difference as a BigDecimal object

boolean toBoolean(String or Number value)

Converts value to a JavaScript Boolean

The value can either be a String containing "TRUE" or "FALSE", or a Number containing zero (for TRUE) or non-zero (for FALSE). The function returns TRUE or FALSE depending on the parameter passed. If the value cannot be converted to Boolean, FALSE is returned.

BigDecimal toDecimal(BigDecimal value, Number scale)

Converts value to a BigDecimal object, and returns the result of the conversion to the number of significant figures specified with scale

Any number can be used as the scale value; default is 2.

If you want to assign the result to a UDA, ensure that the UDA is of the BIGDECIMAL type.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal value.

Packages.java.lang.Float.parseFloat

For a description, refer to the Javadoc provided with the JDK.

Packages.java.lang.Integer.parseInt

For a description, refer to the Javadoc provided with the JDK.

Packages.java.lang.String.valueOf

For a description, refer to the Javadoc provided with the JDK.

Note

As numeric values are of the Number type in JavaScript, only the following range can be treated as Integer:

- Minimum value: -9007199254740992
- Maximum value: 9007199254740992

Note that the following minimum and maximum values of the Long type cannot be used in JavaScript expressions:

- Minimum value: Packages.java.lang.Long.MIN_VALUE
- Maximum value: Packages.java.lang.Long.MAX_VALUE

A.2 JavaScript Functions Supported with Java Actions

With Java Actions, you can use the functions explained in "A.1 General JavaScript Functions" and the functions listed below.

The functions listed below use the Server Enactment Context API, com.fujitsu.iflow.server.intf.ServerEnactmentContext, to provide access to workflow information. For a description, refer to the *API Javadoc*.

void sec.addAttachment(String attachmentName, String attachmentPath)

void sec.addProcessXMLAttributeSubstructure(String udaName, String xPath, String value)

void sec.addProcessXMLAttributeSubstructureByIdentifier(String identifier, String xPath, String value)

void sec.deleteAttachment(String attachmentName)

void sec.deleteProcessXMLAttributeSubStructure(String udaName, String xPath) void sec.deleteProcessXMLAttributeSubStructureByIdentifier(String identifier, String xPath)

void sec.escalateActivity(String assignees)

String sec.getActivityActor(String activityName)

Array sec.getActivityAssignees()

String sec.getActivityName()

String sec.getActor()

Array sec.getAllAttachmentNames()

Array sec.getAllAttributeNames()

String sec.getAttachment(String attachmentName)

Number sec.getCurrentActivityId()

Number sec.getCurrentProcessId()

Array sec.getGroupMembers(String groupName)

String sec.getProcessAttribute(String attName)

String sec.getProcessAttributeByIdentifier(String identifier)

String sec.getProcessAttributeStringType(String udaName)

String sec.getProcessDefinitionId()

String sec.getProcessDefinitionName()

String sec.getProcessDescription()

String sec.getProcessInitiator()

String sec.getProcessName()

Array sec.getProcessOwners()

Number sec.getProcessPriority()

Number sec.getActivityPriority()String sec.getProcessTitle()

String sec.getProcessXMLAttributeElementValue(String udaName, String xPath)

String sec.joinString(Array)

Array sec.resolveRelationship(String relationship, String sourceValue)

void sec.sendEmail(String to, String from, String cc, String bcc, String subject, String body, String mimeType)
void sec.setActivityAssignees(Array assignees)

void sec.setOwners(Array users)

void sec.setProcessAttribute(String name, String value)

void sec.setProcessAttributeByIdentifier(String identifier, String value)

void sec.setProcessDescription(String description)

void sec.setProcessName(String name)

void sec.setProcessOwners(Array users)

void sec.setProcessPriority(Number priority)

void sec.setActivityPriority(Number priority)

void sec.setProcessTitle(String title)

void sec.setProcessXMLAttributeElementValue(String udaName, String xPath, String value)

void sec.setProcessXMLAttributeElementValueByIdentifier(String identifier, String xPath, String value)

void sec.setProcessXMLAttributeSubstructure(String udaName, String xPath, String value) void sec.setProcessXMLAttributeSubstructureByIdentifier(String identifier, String xPath, String value)

void sec.validateProcessXMLAttributeValue(String udaName)

void sec.validateProcessXMLAttributeValueByIdentifier(String identifier)

```
Array sec.splitString(String commaSeparatedList)
```

Using User Defined Attributes (UDAs)

You can use UDAs that have been added to the process definition in your JavaScripts by using the following syntax:

uda.<udaIdentifier>.

Note that you must use the identifier and not the name of the UDA, because multibyte characters are not allowed for variable names in JavaScript.

The following example creates a variable and initializes it to the value of a UDA.

var someVariable = uda.Price;

The following example shows how to assign the value of a variable to a UDA.

var lastName = "Jones";

```
uda.udaIdentifier = lastName;
```

- The methods uda.get and uda.set allow you to access UDAs by their names.
- uda.get returns the value of the specified UDA

  var value = uda.get("<udaName>");
- uda.set sets the UDA to the specified value

  uda.set("<udaName>", "<udaValue>");

When assigning a JavaScript return value to a UDA, ensure that their data types match.

Otherwise, the assignment fails.

Note

If the assignment of a value to a target UDA fails due to conversion or any other kind of error, error details are logged in the IBPMServer.log. The target UDA is not updated and holds its earlier value.

Interstage BPM maps UDA data types to the following Java data types:

- UDAs of the BIGDECIMAL type are mapped to Packages.java.math.BigDecimal objects
- UDAs of the DATE type are mapped to Packages.java.util.Date objects

For details about Java objects, refer to the Javadoc provided with the JDK.

A.3 JavaScript Functions Supported with Triggers

With triggers, you can use JavaScript expressions to specify control conditions. Control conditions narrow down when the trigger fires.

You can use the functions explained in "A.1 General JavaScript Functions" and the function explained below.

String eventData.getXMLData(String xpath)

Returns the text value of the XML element specified in the xpath expression

Example:

Consider the following XML fragment:

```
<Customer>
 <Data>
```

```
  <Name>John</Name>
</Data>
</Customer>
```

The following statement assigns the text value "John" of the <Name> XML element to the name variable.

```
var name = eventData.getXMLData ( "/Customer/Data/Name/text()");
```

Appendix B Project Components

Systemwalker Runbook Automation supports a set of files that you need for running a specific process solution:

This appendix provides a complete list of files supported by Systemwalker Runbook Automation, and contains a brief description of all these files:

Folder Name	Description of Files	Location in the Application Project Directory
Process Definitions	The **Process Definitions** folder contains all process definition (.xpdl) files.	<Project Root>/Process Definitions
web	The default **web** folder contains files that will be exposed to the Internet, such as various image (.gif, .jpeg) files, calendar setup (.js) files, design (.css) files, and QuickForm (.jsp, .qf) files.	<Project Root>/web
dms	The **dms** folder contains all sorts of attachments of Workflow Application projects in **Attachments** folder, such as image (.gif) files, preview (.wmf) files, and .html files. **Attachments** folder also contains Blaze and new ILOG rules files.	<Project Root>/dms and <Project Root>/dms/Attachments
Application Classes/ engine	The **engine** folder contains the **classes**, **lib**, and **js** subfolders. The **classes** folder contains the class (.class) files loaded by the application at runtime. The **lib** folder contains the jar (.jar) files used by your application. The **js** folder contains the Java Script (.js) files used by your application.	<Project Root>/Application Classes/ engine/classes and <Project Root>/Application Classes/engine/lib
Simulation	The **Simulation** folder contains the scenario files storing simulation scenarios (.ssr files) and, after running a simulation, the scenario results.	<Project Root>/Simulation
Calendar	Calendar (.cal) file used for configuring new business calendars. Calendars are similar to property or .ini files. They consist of name-value pairs. To indicate a period of time, commas are used. One project can have several calendar files.	<Project Root>/Calendar
Resources	Resources folder contains the following: - FTP Agent files - HTTP Agent files - Custom Config files	<Project Root>/Resources

Folder Name	Description of Files	Location in the Application Project Directory
	- Configuration file for Java Agents used by the application (agentsConfig.xml) - File Listener file (fileListenerConf.xml)	
Rules	The **Rules** folder contains **Rules Set** sub-folders, which contains the Decision Table (DTM files).	<Project Root>/Rules
Documents	The **Documents** folder contains report and document files.	<Project Root>/Documents
Analytics	The default **Analytics** folder is Analytics empty. After installing the Interstage Analytics plugin, this folder contains the process definitions (.xpdl files) defining the Analytics setting.	<Project Root>/

Appendix C Synopsis of Files

This appendix describes the synopsis of files.

C.1 Java Agent File (agentsConfig.xml)

Agent assigned to an activity node in agentsConfig.xml can be executed within a process instance as an activity.

File format

The <DMSRoot>/apps/<*application ID*>/agentsConfig.xml file is used to configure an agent and is part of a workflow application project.

A sample file is displayed below:

```
<ActionAgentList>
   <ActionAgent>
      <Name>@TestFrameAgent</Name>
      <Description>For tests. Returns the first arrow. </Description>
      <RetryInterval>20</RetryInterval>
      <EscalationInterval>1</EscalationInterval>
      <ClassName>com.fujitsu.iflowqa.testframe.TestFrameAgent
      </ClassName>
      <ClassPath>IBPMROOT/classes</ClassPath>
      <ConfigFile></ConfigFile>
   </ActionAgent>
</ActionAgentList>
```

The table below describes the tags:

Item	Description
<ActionAgentList>	List of agents.
<ActionAgent>	Definitions of a single agent. Separate definitions for each agent must be coded within this tag.
<Name>	Name of the agent. Value must start with a "@" to indicate that it is an agent. This name must be specified as the agent in charge of an activity node, and this activity must be set as an "Agent" activity. This means that the activity functions as an agent rather than as an ordinary activity.
<Description>	Simple description of the agent, generally explaining its function.
<RetryInterval>	Number of seconds between each invocation of an external service.
<EscalationInterval>	Number of failures after which a mail is to be sent to the system administrator. Mail is sent every time the next escalation is reached (for example, if the value is 1, the system administrator is notified every time the agent fails). Mail is sent to the address specified in the ServerEmailAddress parameter for the Admin Server.
<ClassName>	Name of the Java class associated with the agent - the class is the agent's function part.
<ClassPath>	Classpath of the Java class associated with the agent. This tag is ignored in SaaS mode.
<ConfigFile>	Configuration file used by the agent.

Agent class

The class specified in <ClassName> is the procedure used by the agent to access external services - it will be hereafter referred to as the agent class. The agent class must implement the following interface:

```
package com.fujitsu.iflow.server.intf;
public interface ActionAgentInvoke
   {
      public String invokeService(ServerEnactmentContext sec, String configFile) throws Exception;

}
```

The agent class uses invokeService() to invoke external services and to accept data from those services. The agent then returns STRING type data to Systemwalker Runbook Automation which gives instructions for service actions.

Execution of an agent class results in one of the following:

- NULL STRING type or empty STRING type data:
  The agent repeatedly attempts to execute the service. The retry interval is specified in <RetryInterval> and the agent attempts execution up to the number of times specified in <EscalationInterval> - each attempt sends mail to the address specified in the Admin Server ServerEmailAddress parameter.

- Other STRING type data:
  The agent checks whether the data matches one of the outgoing arrow names used as the start point for agent activities. If it does, then the process instance proceeds in accordance with the matching arrow, otherwise the agent sends a "cannot-find-arrow exception" and an error occurs for the process instance.

- Exception:
  The agent sends the exception to Systemwalker Runbook Automation and an error occurs for the process instance.

C.2 FTP Agent File (ftp.xml)

As with all agents, FTP agents are configured in agentsConfig.xml - its default settings are specified in the <ActionAgent> section and generally do not need to be changed.

FTP agents use the ServiceAgent class and the ftp.xml configuration file, which defines the FTP host address and other FTP settings. Modify this file as needed to suit the environment where it is used.

File format

The <DMSRoot>/apps/<*application ID*>.ftp.xml file is located in the agent directory in the Admin Server installation environment.

A sample file is displayed below:

```
<Services>
   <Service>
      <ServiceType>FTP</ServiceType>
      <ServiceStatusUDA>AgentServiceStatus</ServiceStatusUDA>
      <ServiceResultUDA>AgentServiceResult</ServiceResultUDA>
      <ServiceSpecificInfo>
         <FTPHost><HOSTNAME or IP Address></FTPHost>
         <FTPPort></FTPPort>
         <FTPUser>anonymous</FTPUser>
         <FTPPassword></FTPPassword>
         <FTPType>ASCII</FTPType>
         <FTPAppend>FALSE</FTPAppend>
         <FTPDirectory>%%REMOTE_FTP_DIRECTORY%%
            </FTPDirectory>
         <FTPFileNames>%%REMOTE_FTP_FILES%%</FTPFileNames>
         <Documents>%%OUTGOING_FILES%%</Documents>
      </ServiceSpecificInfo>
   </Service>
</Services>
```

Note that some tags are used by all agents that use the ServiceAgent class, while others are specific to the FTP agent.

The table below describes the tags:

Item	Description
<ServiceType>	Literal "FTP". If this tag is not specified or the specification is invalid, a process instance error occurs. This tag is used by all agents that use the ServiceAgent class.
<ServiceStatusUDA>	Name of the UDA that saves the FTP agent activity status. Possible values are: "Done" and "Failed". The FTP agent is not affected if this tag is deleted, but users will no longer be able to check its status.
<ServiceResultUDA>	Name of the UDA that saves error messages, and is used when the FTP agent sends an exception. The FTP agent is not affected if this tag is deleted but, if the FTP agent fails, users will no longer be able to check for issued errors.
<FTPHost>	Server name or IP address of the local computer that receives transferred files. The default value is 127.0.0.1 (local computer). If no value or an invalid value is specified, then files are sent to this IP address, provided that the FTP server is usable. This tag is specific for FTP agents.
<FTPPort>	Port used by the FTP server. The default value is 21. If no value is specified, then the default is used. This tag is specific for FTP agents.
<FTPUser>	FTP user transferring files. If an invalid value is specified, then file transfer fails and a process instance error occurs. This tag is specific for FTP agents.
<FTPPassword>	Password of the FTP user transferring files. This tag is specific for FTP agents.
<FTPType>	Type of file to be transferred. Possible values are: "ASCII" and "BINARY" (default). Specify "ASCII" to transfer text files, or "BINARY" to transfer binary files. The FTP agent sends an exception if an invalid value is specified. This tag is specific for FTP agents.
<FTPAppend>	Specifies whether to append the content of the transferred file to a remote file. Possible values are: "TRUE" and "FALSE" (default). Specify "TRUE" to append the content of the local file content to existing remote file, or "FALSE" to overwrite it. The FTP agent sends an exception if an invalid value is specified. This tag is specific for FTP agents.
<FTPDirectory>	UDA that specifies the file transfer target directory (the UDA must contain the relative path in the FTP server). For example, if value of the specified REMOTE_FTP_DIRECTORY UDA is "/iflow" and the user is working on a Windows system, then the file is transferred to C:\Inetpub\ftproot\iflow. If an invalid directory is specified, a process instance error occurs. This tag is specific for FTP agents.
<FTPFileNames>	UDA that specifies the name of the remote file. For example, if a file called LocalFile.txt is attached to a process instance and the value of the specified REMOTE_FTP_FILES UDA is "RemoteFile.txt", then LocalFile.txt is sent to the FTP directory and its filename is changed to RemoteFile.txt. Separate multiple filenames by semicolons (;).

Item	Description
	If the specified UDA does not contain a filename, then the original filename is used.
<Documents>	UDA that specifies the files to be transferred to the FTP directory. Separate OUTGOING_FILES multiple filenames by semicolons (;) If the specified UDA does not contain a filename, then all files attached to the process instance are transferred.

Note

Files that have a semicolon (;) included in the filename cannot be transferred to the FTP directory, since it is treated as a delimiter between filenames.

C.3 HTTP Agent File (HTTPagent.xml)

As with all agents, HTTP agents are configured in agentsConfig.xml - its default settings are specified in the <ActionAgent> section and generally do not need to be changed.

File format

Create the <DMSRoot>/apps/<*application ID*>/ HTTPagent.xml file in the Admin Server agent directory.

A sample file is displayed below

```
<HTTPAgent>
   <HTTPBaseURL method="POST"
      followRedirects="true">{{Field HTTP_URL}}</HTTPBaseURL>
   <va name="Content-Type">text/xml;
      charset=UTF-8</HTTPHeaderParams>
   <HTTPHeaderParams name="User-Agent">I- HTTP Agent
   </HTTPHeaderParams>
   <HTTPHeaderParams name="accept-charset">UTF-8
   </HTTPHeaderParams>
   <QueryParams name= "ParamName"> Param value</QueryParams>
   <HTTPRequestUDA>HTTP_REQUEST</HTTPRequestUDA>
   <HTTPResponseUDA>HTTP_RESPONSE</HTTPResponseUDA>
   <HTTPAgentStatusUDA>HTTP_STATUS</HTTPAgentStatusUDA>
</HTTPAgent>
```

The table below describes the tags:

Item	Description
<HTTPBaseURL>	URL end point invoked by the HTTP agent. This tag specifies a servlet or other external service. Attributes: - HTTP method - Possible values are: "method="POST"" and "method="GET"". This attributes specifies the request method used for the HTTP request. If this attribute is not specified, a process instance error occurs. If an invalid value is specified, the HTTP agent fails. - HTTP redirects follow - Possible values are: "followRedirects="true"" and "followRedirects="false"". Since this attribute's default is "followRedirects="false"", this attribute must be specified.
<HTTPHeaderParams>	Specifies HTTP header properties such as the encoding style and contents type.

Item	Description
	HTTP headers are used to define various Web object properties, such as request and response object properties for the current HTTP session, for example. Attributes: - Name - Possible values are currently not restricted, that is, all values valid in HTTP headers can be specified. Among the possible values for this attributes are: - "name="content-type" - Defines the type of contents requested. If this attribute is specified, the tag's possible values are: "text/xml" and "charset=UTF-8". - "name="User-Agent"" - Defines information regarding the client (user agent) that sent the request.
<QueryParams>	Value passed to external services as the query string (this tag is required if the HTTP agent uses the "GET" method). Note that the UDA value specified in <HTTPRequestUDA> is not used if the "GET" method is used. This tag's syntax is as follows: - <QueryParams name= "ParamName"> Param value</QueryParams> During execution of the HTTP agent, this is converted to: - URL?ParamName= Param value Any valid string can be specified as the name attribute.
<HTTPRequestUDA>	UDA which value will be part of the HTTP request query string. Note that the value of the specified UDA is not used if the "GET" method is used.
<HTTPAgentStatusUDA>	UDA that saves the status of the HTTP service.
<HTTPResponseUDA>	UDA that saves HTTP responses.

C.4 File Listener File (fileListenerConf.xml)

A file listener monitors the files in a specified directory. When a file handler detects a new or modified file, it notifies the Systemwalker Runbook Automation file handler so that automatic functions (such as starting a process instance or selecting an activity) can be executed. A file listener is generally used to integrate Systemwalker Runbook Automation with other enterprise applications.

Create process definitions and add to the file listener the triggers that start process instances. Create a trigger event data file (XML file) and save it to the filelistener directory.

File format

The file name is <DMSRoot>/apps/<application ID>/fileListenerConf.xml.

A sample file is displayed below:

```
<FileListener>
   <Directory>
      <ScanInterval>60000</ScanInterval>
      <StabilizationPeriod>2000</StabilizationPeriod>
      <PostProcessing>
         <onSuccess>
            <Delete></Delete>
         </onSuccess>
               <onError>
            <Move>
            </Move>
         </onError>
      </PostProcessing>
   </Directory>
</FileListener>
```

The table below describes the tags:

Item	Description
<Directory>	Configuration of the directory that monitors new and modified files.
<Path>	Path of the directory that monitors files. In SaaS mode or non-SaaS mode, if the path is not specified, then <ServerSharedRoot>/tenants/<*tenant name*>/apps/<*application ID*>/filelistener is used.
<ScanInterval>	Number of milliseconds between each check for new files in the directory.
<StabilizationPeriod>	This setting is for monitoring file sizes to check for changes before a file is processed (the unit is milliseconds).
<PostProcessing><onSuccess>	Action to take when a file process succeeds. Possible values are: "<Delete>" and "<Move>".
<PostProcessing><onError>	Action to take when a file process fails. Possible values are "<Delete>" and "<Move>".
<Delete>	Determines that the file will be deleted.
<Move>	Determines that the file will be moved to the specified directory. On success: - In non-SaaS mode, the file is moved to the specified directory. - If the path is not specified in SaaS mode or in non-SaaS mode, the file is moved to <ServerSharedRoot>/tenants/<*tenant name*>/apps/<*application ID*>/filelistener/success. On error: - In non-SaaS mode, the file is moved to the specified directory. - If the path is not specified in SaaS mode or in non-SaaS mode, the file is moved to <ServerSharedRoot>/tenants/<*tenant name*>/apps/<*application ID*>/filelistener/error.

C.5 Custom-Built Files

For details, refer to C.1 Java Agent File (agentsConfig.xml).

Appendix D Naming Conventions

This appendix explains various name definitions.

Refer to "Naming Convention"below for a brief explanation of conventions 1 through 4 used in the "Applicable Naming Conventions" column in the following table.

Name Definitions

Name Type	Name Length	Extension	Applicable Naming Convention	Scope of Uniqueness
Workspace	No limit (*1)		Convention 1	Unique to the Development Computer
Workflow application project	128 characters		Convention 2	Unique to a workspace
Application variables	256 characters		Convention 3	Unique to a project
Server project name	255 characters (*1)		Convention 2	Unique to a workspace
Owner group	256 characters		-	
Server connection information name	No limit (*1)		-	Unique to a workspace
Folder name	255 characters (*1)		Convention 2	Unique to a project
File name	255 characters (*1)		Convention 2	Unique to a project
Scenario name	64 characters		-	
Scenario file name	No limit (*1)	ssr	Convention 1	Unique to a simulation within project
Calendar name	64 characters	cal	Convention 1	Unique to a calendar within the project
Resource name (FTP agent)	64 characters	xml	Convention 1	Unique to resources within the project
Resource name (HTTP agent)	64 characters	xml	Convention 1	Unique to resources within the project
Custom configuration name	64 characters	xml	Convention 1	Unique to resources within the project
Rule set name	255 characters		Convention 2	Unique to rules within the project
Decision rule table name	64 characters	dt	Convention 1	Unique to a rule set
Process definition name	64 characters		Convention 2	
Process definition file name	255 characters (*1)	xpdl	Convention 2	Unique to a project
Process instance owner	200 characters		-	
Process starter privileges	200 characters		-	
Priority level	9999999999		Numerals only	
Node name	64 characters		-	
Arrow name	64 characters		-	
Swimlane name	30 characters		-	
Group name	30 characters		-	
QuickForm name	64 characters (*1)	jsp	Convention 1	Unique to a project

Name Type	Name Length	Extension	Applicable Naming Convention	Scope of Uniqueness
User definition attribute name	64 characters		-	Unique to a process definition
User definition attribute ID	32 characters		Convention 3	Unique to a process definition
Filter name	64 characters		-	Unique to operation component activity
Action set action name	64 characters		-	Unique to owner action, default action, completed action
Action set note	256 characters		-	
Operation component project name	256 characters		Convention 1	Unique to a workspace
Operation component name	61 characters		-	Unique to a workspace
First half of operation component ID	5 characters		Convention 5	Together, the first half and second half are unique to a workspace
Second half of operation component ID	55 characters		Convention 3	
Input information name	64 characters		Convention 3	Unique to operation components
Specified value of maximum number of characters for input information	2147483647		Numerals only	
Component tooltip	256 characters		-	
Script file name	61 characters	pl or rb	Convention 4	Up to 1 operation component
Debug configuration name	248 characters (*1)		Convention 1	Unique to a workspace

*1: Dependent upon file system threshold limit as well as entry length restrictions.

Naming Conventions

- : No naming convention applicable.

Convention 1:The characters \ | / : * ? " < > must not be used.

Convention 2: In addition to Convention 1, must not conclude with a period.

Convention 3: Only half-width alphanumeric characters and underscores may be used.

Convention 4: Only half-width alphanumeric characters, underscores, hyphens, and periods may be used.

Convention 5: Only half-width alphanumeric characters and underscores may be used. Also, "SWRBA" cannot be used.