Interstage

FUJITSU

Interstage Business Process Manager V11.2

Studio User's Guide

J2U3-0063-03ENZ0(00)
July 2011

Publication Date	July 2011
Revision	2

High Risk Activity

Table of Contents

About this Manual

This manual describes how to model business processes using the Interstage Business Process Manager Studio.

Intended Audience

This manual is intended for process designers who define their business process models using the Interstage Business Process Manager Studio. It assumes that the reader has experience on working with graphical user interfaces.

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.

This Manual Contains

Chapter	Title	Description
1	Introduction	Introduction to Interstage BPM, and to process definitions and their components.
2	Getting Started	Explains how to start and exit Interstage BPM Studio, and how to model your first process.
3	Understanding and Customizing the User Interface	Introduces the user interface components and explains how to customize the user interface.
4	Managing Projects	Describes how to create and work with Workflow Application and server projects.
5	Managing Process Definitions	Describes how to create and work with process definitions.
6	Modeling Processes	Explains how to add nodes, swimlanes, groups, and arrows and how to define their properties.
7	Using Process Fragments	Explains how to use reusable process fragments to quickly define processes.
8	Using Forms	Explains how to create QuickForms that are presented to end-users during process execution.
9	UDAs of Type CUSTOM	Explains how to use UDA of type CUSTOM.

Chapter	Title	Description
10	Modeling Subprocesses, Remote Subprocesses and Chained-Processes	Describes how to model processes that need to be split in multiple process definitions.
11	Decision Tables	Explains how Decision Tables allow you to design advanced rules for decision making without programming.
12	Advanced Process Modeling	Explains how to work with Java Actions, Triggers, and JavaScript Expressions.
13	Simulating Processes	Describes how to create and use simulation scenarios for simulating the execution of a process.
Appendix A	Supported JavaScript Functions	Describes the JavaScript functions supported by Interstage BPM.
Appendix B	Project Components	Explains all components that are part of a project
	Glossary	Glossary of terms.

Typographical Conventions

The following conventions are used throughout this manual:

Example	Meaning
`command`	Text you are required to type at a command line is identified by `Courier` font.
screen text	Text that is visible in the user interface is **bold**.
Reference	Reference material is in *italics*.
`Parameter`	A command parameter is identified by `Courier` font.

Other References

The following references for Interstage Business Process Manager are also available:

- *QuickForm UI Widgets Reference*

  Describes how to use UI widgets of the QuickForm in the Interstage BPM Studio. This guide is stored to the following folder:

  `<Studio install dir>\share\docs\en\QuickFormWidgetsReference.pdf`

  where `<Studio install dir>` is the directory to which you have installed the Interstage BPM Studio, for example: `C:\fujitsu\InterstageBPM_studio`.

- *Release Notes*

Contains an overview of Interstage Business Process Manager, installation tips, and late-breaking information that could not make it into the manuals.

- *Interstage Business Process Manager Server and Console Installation Guide*

  Describes software and hardware requirements, installation procedure for Interstage Business Process Manager Server and Console
- *Interstage Business Process Manager Server Administration Guide*

  Explains how to configure and administrate Interstage Business Process Manager Server. This guide also describes the configuration parameters of the Interstage BPM Server.
- *Interstage Business Process Manager Developer's Guide*

  Describes how to use the Interstage Business Process Manager API to customize and extend Interstage BPM to fit the unique needs of your organization.
- *Interstage Business Process Manager Tenant Management Console Online Help*

  Explains how to use the Interstage Business Process Manager Tenant Management Console user interface.
- *Interstage Business Process Manager Console Online Help*

  Explains how to use the Interstage Business Process Manager Console user interface.
- *Interstage Business Process Manager ARIS Process Performance Manager Integration Guide*

  Describes how to install and configure the PPM adapter and the PPM autoConfig tool. With these two programs, process data can be transferred from Interstage Business Process Manager to ARIS Process Performance Manager.
- *API Javadoc*

  This HTML documentation provides the API and syntax of the packages, interfaces and classes for developing custom applications or embedding Interstage Business Process Manager into other products.

Abbreviations

The products described in this manual are abbreviated as follows:

- "Interstage Business Process Manager" is abbreviated as "Interstage BPM".
- "Interstage Business Process Manager Studio" is abbreviated as "Interstage BPM Studio".
- "Oracle Solaris" might be described as "Solaris", "Solaris Operating System", and "Solaris OS" in this document.

1 Introduction

This chapter gives an overview of Interstage Business Process Manager and Interstage Business Process Manager Studio. It also explains basic concepts you need to know before getting started.

1.1 About Interstage Business Process Manager

Interstage Business Process Manager empowers business groups to collaboratively plan, automate, track, and improve business processes. It allows to design automated processes that comprise the sequence of activities, human and IT resources as well as the data to be handled.

Interstage BPM has an open, flexible, scalable architecture that integrates seamlessly into existing environments. Its Application Programming Interfaces (APIs) empower developers or system engineers to tailor Interstage BPM to an organizations' specific needs. Interstage BPM is made up of several components.

The **Interstage BPM Studio** is an easy-to-use business process modeling tool. It enables process designers and system integrators to graphically model the business processes of an enterprise. Entire workflow applications including all artefacts required to run the application can be created using the Interstage BPM Studio. Such an application can be developed offline and later be deployed on an Interstage BPM Server that can be accessed by end-users. To facilitate developing and deploying an application, Interstage BPM Studio provides a wide range of upload and download functions, as well as import and export functions.

Interstage BPM Studio is also available as an Eclipse-plugin. When integrating BPM Studio into the Eclipse software framework, you can use Eclipse as a development environment for Java technology. Eclipse has many features that make for quick and painless Java programming and debugging. For more information, refer to section *Using Interstage BPM Studio as Eclipse Plug-In* on page 18.

Interstage BPM Server is a Web-enabled workflow engine. It provides the run-time environment for process instances created from process definitions.

The Browser-based **Interstage BPM Console** enables users to start processes, respond to work items and administer the Interstage BPM Server.

The following figure shows the Interstage BPM components.

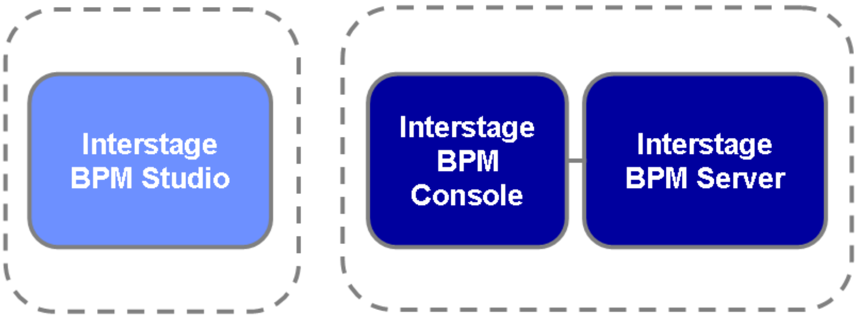


Figure 1: Interstage BPM Components

For details on the Interstage BPM architecture, refer to the *Interstage Business Process Manager Server Administration Guide*.

1.2 Interstage BPM Studio Features

Interstage BPM Studio is an intuitive and easy to use process modeling tool and web application development tool. It is based on the latest user interface technologies and provides the following features:

- **Offline Editing**

  Interstage BPM Studio is a standalone tool that can be used independently of an Interstage BPM Server. Individual process definitions can be exported from Interstage BPM Studio and imported to an Interstage BPM Server.

  In addition, Interstage BPM Studio allows you to design entire **workflow applications** offline. Later you can deploy such an application on an Interstage BPM server that can be accessed by Interstage BPM clients. A Workflow Application project contains everything necessary for the running of a process solution, for example process definitions, forms, simulation scenarios, attachments, etc. When creating a Workflow Application project, you are working locally and are thus independent of an Interstage BPM Server. You can easily upload entire applications to a remote server and download applications from a server. You can also package applications to a file and thus export a project to a file, as well as import a project from the local file system.

  Interstage BPM Studio supports accessing multiple tenants on Interstage BPM Server.

  In SaaS mode:

  - A default tenant as well as some other tenants exist.
  - Each tenant can have multiple applications.
  - Each tenant has a BaseURL. When you upload or download the application, you access this BaseURL.

  In non-SaaS mode:

  - Only the default tenant exists.
  - The default tenant can have multiple applications.
  - The default tenant has a BaseURL. When you upload or download the application, you access this BaseURL.

- **Online Editing: Server Projects**

  In addition to working offline in the Interstage BPM Studio, using it as a standalone tool, you can work directly on a connected Interstage BPM Server. Process definitions that you create with Interstage BPM Studio are stored directly in the database located on the Interstage BPM Server.

  Using a Server Project, you can access only the `System` application in the `default` tenant .

- **Process Simulation**

  Interstage BPM Studio allows for simulating the execution of business processes. Simulation scenarios can be defined, and, based on the scenarios, cost and time can be calculated and reports can be generated for analysis and optimization purposes.

- **Swimlanes and Groups**

  Interstage BPM Studio supports swimlanes and groups for organizing activities into visual categories. Swimlanes are used to group activities by the same Role to illustrate different functional capabilities or responsibilities. Groups are used to group activites by a specific category.

  Both, swimlanes and groups do not affect the sequence flow of the activities within.

- **Flexible Arrows**

Flexible multi-point arrows are shown with editable names. The user interface allows easy source and target node selection for drawing arrows. To draw arrows, you have to drag from the source node to the target node.

- **Alignment**

  Rulers, grids and toolbar buttons allow for easy horizontal and vertical alignment of nodes, swimlanes, and groups.
- **Selection of Multiple Elements**

  Multiple nodes, swimlanes, and groups can be selected for moving and alignment.
- **Zoom Support and Navigation**

  You can zoom in or out on process definitions. An Outline view and an Overview view facilitate navigating large process definitions.

1.3 Using Interstage BPM Studio as Eclipse Plug-In

Interstage BPM Studio can be used as a standalone application or as a plug-in to the Eclipse software development environment.

- **Using BPM Studio as application**: This is the default use of Interstage BPM Studio.
- **Using BPM Studio as Eclipse plug-in**: The Eclipse software development kit provides a Java development environment. BPM Studio can be easily used as an Eclipse-based application. It seamlessly integrates with the Eclipse platform.

Using Interstage BPM Studio as an Eclipse plug-in requires the following:

- A complete Interstage BPM Studio installation: The Eclipse plug-in module automatically comes with the Interstage BPM Studio installation. Note that even when you use Interstage BPM Studio as Eclipse plug-in, you need an installation directory of Interstage BPM Studio.
- The Eclipse Integrated Development Environment (IDE) and the Eclipse Web Tools Platform (WTP): The Eclipse IDE is the default form of the Eclipse software framework. The Eclipse WTP provides the Eclipse platform along with the required tools for developing Web and Java EE applications and building rich, interactive user interfaces.

Note: Interstage BPM supports Eclipse IDE 3.4.1 and Eclipse WTP 3.0.2.

For more information on developing and using Eclipse plug-ins, refer to the Eclipse documentation.

To use Interstage BPM Studio as Eclipse plug-in:

1. From the Eclipse website, download the **Eclipse IDE for Java Developers** package of **Eclipse Ganymede SR1 Packages (v3.4.1)** to download all the plug-ins required for the Eclipse WTP. After this, download the Eclipse WTP from the Eclipse website.

Note: If you use the **Eclipse IDE for Java EE Developers** package, you will not need to download WTP and related plug-ins separately.

Note: To download the **Eclipse IDE for Java Developers** package of **Eclipse Ganymede SR1 Packages (v3.4.1)**, use the website *http://www.eclipse.org/downloads/packages/release/ganymede/sr1/*.

To download the Eclipse WTP, use the website *http://archive.eclipse.org/webtools/downloads/drops/R3.0/R-3.0.2-20080921203356/* .

Note that these websites for these downloads might change in future.

2. Execute the plug-in installation script (WSH script) located at `<Interstage BPM Studio install directory>\etc\install_plugins.js` and specify the Eclipse installed directory. To execute this script, you need to double-click it in Windows Explorer.

Note: If the workspace is created with the V11.0 or earlier versions of Studio, reset the perspective by clicking **Reset Perspective** in the **Window** menu. This ensures that views and menus related to Interstage BPM Studio display correctly.

Note: If you want to **uninstall** the Eclipse plug-in, execute the Eclipse plug-in uninstall script (WSH script) located at `<Interstage BPM Studio install directory>\etc\uninstall_plugins.js`. To execute this script, you need to double-click it in Windows Explorer.

1.4 Process Definitions

Process definitions represent a business process in a form that supports automated manipulation. They define the behavior and properties of the process instances created from them including the flow of control within the process, based on the preceding activity.

A process definition consists of activities and their relationships, criteria to indicate the start and end of the process, and information about the individual activities, such as participants and the data to be handled.

The following figure shows a sample process definition. It describes the basic activities that are accomplished with a bank loan approval:

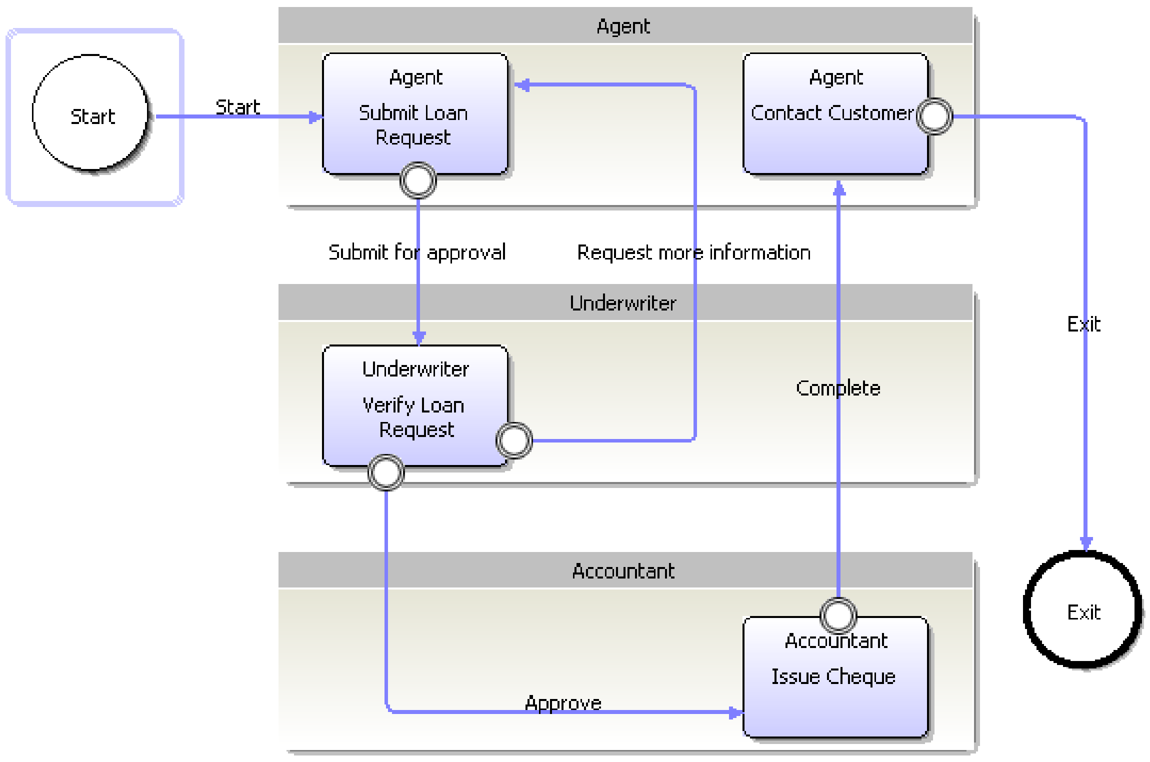


Figure 2: Sample Process Definition

1.5 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.

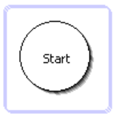
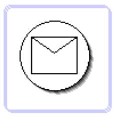
Interstage BPM supports different node types. Each node type is represented by its own graphical symbol.

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

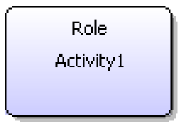
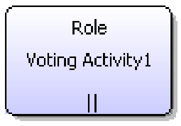
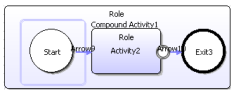
Start Node

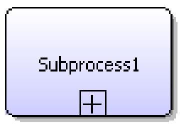
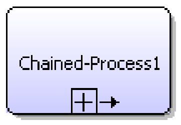
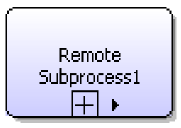
The Start Node identifies the beginning of a process. Every process definition has one and only one Start Node, which is created automatically whenever you create a new process definition. Depending on whether a trigger has been defined for the process definition, the Start Node is displayed as follows:

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

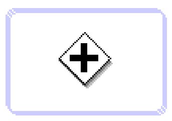
These are nodes that usually involve human activity in order to complete them:

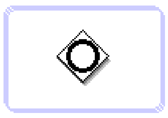
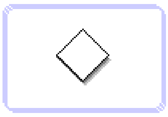
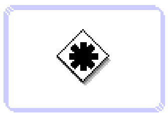
Node Type	Symbol	Description
Activity Node		Activity Nodes are the main building elements of a process definition. They model tasks that require user actions or decision-making. They define the tasks, the forms associated with the task and the users assigned to perform the task.
Voting Activity Node		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 Role Activity2 Start Arrow9 Arrow10 Exit3	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.

Node Type	Symbol	Description
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.
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.
Remote Subprocess Node		A Remote Subprocess Node represents a subprocess that runs on a remote workflow server. A remote workflow server might be, for example, another Interstage BPM Server. When a Remote Subprocess Node is reached, control is passed to the remote subprocess. The parent process waits for the remote subprocess to complete and the results to come back.

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.

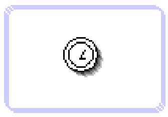
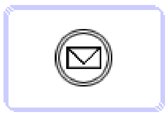
Node Type	Symbol	Description
AND Node		An AND Node synchronizes multiple branches in a process. When an AND Node is reached, the process does not continue until all activities that lead to this node are completed.

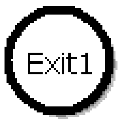
Node Type	Symbol	Description
OR Node		An OR Node splits process flow into multiple parallel branches. When an OR Node is reached, it activates all subsequent nodes connected to it simultaneously.
Conditional Node		A Conditional Node directs process flow along one of several branches, depending on a specified criteria. 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 is similar to a Conditional Node. The only difference is the type of criteria that is specified to direct process flow. For a Conditional Node, a value is compared to another value. For a Complex Conditional Node, advanced criteria can be specified using a JavaScript expression.

Note: It is necessary to use the **OR** node and the **AND** node to execute the parallel processing. When neither the OR node nor the AND node is used, the execution of the parallel processing is not guaranteed.

Event Nodes

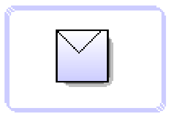
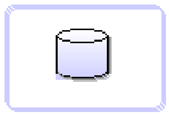
This area includes nodes that involve some kind of event related to the process flow. They are all automatic nodes that do not involve human interaction.

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.
Trigger Node		A Trigger Node represents a step in the process which is driven by an external event. Each Trigger Node has a defined trigger that is supposed to fire when data, typically XML files, comes in from an external system. Once the data arrives, the trigger moves the data into User Defined Attributes (UDAs) according to the trigger's definition. The trigger then makes a choice and process execution moves to the next node.

Node Type	Symbol	Description
Exit Node		An Exit Node identifies the end of a process branch. Every 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.

Node Type	Symbol	Description
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.
DB Node		A DB Node (Database Node) accesses an external database using JDBC. The SQL statements to be executed are specified with the DB Node. After accessing the external database, the DB Node activates all subsequent nodes connected to it simultaneously. No user action is required with this node.
Web Service Node		A Web Service Node retrieves data from a Web Service and makes it available for further processing. After retrieving data, the Web Service Node activates all subsequent nodes connected to it simultaneously. No user action is required with this node.

1.6 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. Interstage BPM Studio displays these properties in the Process Definition editor (for more information, refer to section *Displaying Properties* on page 69).

You can turn off or on the display of the properties in the Process Definition editor by 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 *Using Triggers* on page 353. Triggers can be used with process definitions, Activity Nodes and Trigger 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 *Using Due Dates and Timers* on page 169. Timers can be used with process definitions, Activity Nodes, Voting Activity Nodes, Compound Activity Nodes and Trigger Nodes.
Action	⚙	Java Actions are 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 *Using Java Actions* on page 265. 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, OR Nodes, and Trigger Nodes.
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 *Defining Exception Handling* on page 348. Error Actions can be used with process definitions and all nodes, except Start Nodes and Exit Nodes.
Form	▤	Forms can take data from a process and present it to the user. The Form symbol is displayed when forms have been defined. For more information on forms, refer to chapter *Using Forms* on page 191. Forms can be used with Activity, Voting Activity, Compound Activity, and Start Nodes. They 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 *Defining Java Agents* on page 362. 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.

Property Type	Symbol	Description
Iterator (Parallel) Loop Node	⫼	Iterator (Parallel) Loop node is used to create two or more node instances concurrently. You can configure a node as an Iterator Node only on Activity Node, Subprocess Node and Chained-Process Node. Refer *Configuring Node as Iterator (Parallel) Loop Node* on page 162 for more details.
Sequential Loop Node	↻	In Sequential Loop node, multiple node instances are generated sequentially 'while' a certain condition is being satisfied. You can configure an Activity Node and the Subprocess Node as Sequential Loop node. Refer *Configuring Node as Sequential Loop Node* on page 164 for more details.

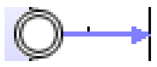
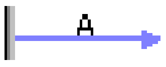
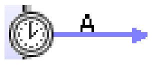
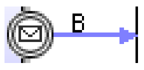
Refer to section *Arrows* on page 26 for a description of the symbols used with arrows.

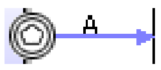
1.7 Arrows

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, Compound Activity Node, or Remote Subprocess Node. This expression means that only one arrow is executed at runtime.
Outgoing arrow from a node except Activity, Voting Activity, Compound Activity, or Remote Subprocess Node.		The arrow originates from a node except Activity, Voting Activity, Compound Activity, or Remote Subprocess Node. 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 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.

Property Type	Symbol	Description
Multiple events		The arrow originates from an Activity Node for which one or several triggers and timers are defined.

1.8 Forms

Most business processes involve decisions and actions that are based on information. In Interstage BPM, "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 *Using Forms* on page 191 for details.

1.9 Roles

In a typical business process, several people are performing tasks. With Interstage BPM, 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.

With Interstage BPM, you assign activities (Activity Nodes, Compound Activity Nodes, or Voting Activity Nodes) to Roles according to who is responsible for completing the activity. The advantage of using Roles rather than individual users is that if personnel changes occur, only the Role definition needs to be updated and not all the process definitions that use the Role.

1.10 Swimlanes and Groups

Many process modeling methods use swimlanes or groups for organizing activities into visual categories. These visual categories illustrate different functional capabilities or responsibilities. With Interstage BPM Studio, you can use swimlanes to visually group activities performed by the same Role. Groups are irrespective of Roles.

The grouping is purely visual; nodes placed on swimlanes or in groups are not processed in a special way.

Refer to section *Process Definitions* on page 19 for an example of a process definition that uses swimlanes.

1.11 Annotations

Interstage BPM Studio allows you to add annotations to your process definition. Using annotations, you can make comments and explain important aspects of your process design.

In addition to user-defined annotations, Interstage BPM Studio shows the list of Java Actions (if defined for a node) in the same way. This list of Java Actions is called "Action Annotations".

An Action Annotation is available for and contains information of the following Java Actions:

- SendEmail
- Web Service Call
- Delete SQL Java Action

- Insert SQL Java Action
- Select SQL Java Action
- Update SQL Java Action
- Decision Tables
- Fair Isaac Blaze Advisor
- ILOG JRules

Action Annotations are attached to the following types of nodes:

- Activity Node
- Voting Activity Node
- Compound Activity Node
- Subprocess Node
- Chained-Process Node
- Remote Subprocess Node

1.12 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. Interstage BPM Studio allows you to design Workflow application projects offline. Later you can deploy such an application on an Interstage BPM Server that can be accessed by Interstage BPM clients.

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, Interstage BPM 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 *Project Components* on page 411 for a complete overview), for example:

- JSP files of QuickForm
- HTML files that are generated from the layout definitions
- Required images or required HTML files
- Blaze files
- ILOG rules files
- Java Script files
- Simulation scenarios and simulation result files
- Calendar files
- Configuration file for Java Agents used by the application
- Configuration files for new data sources used by Interstage BPM Studio
- FTP Agent files
- HTTP Agent files
- Custom Config files
- File Listener files

- Adapter classes
- Rule files
- Document files
- Help files
- XML Schema

Also, refer *Menu Bar* on page 51 for more information about these components.

1.13 Offline Editing and Transfer to Server

Interstage BPM Studio is a standalone tool. It allows you to model your Workflow Application projects together with process definitions offline, that is without connection to an Interstage BPM Server. All artefacts that you create with Interstage BPM Studio are stored in a file system folder of your choice.

You can export individual process definitions or entire Workflow Application projects from the Interstage BPM Studio and import or deploy them on an Interstage BPM Server.

Transferring Individual Process Definitions

On the Interstage BPM Server, you can publish process definitions and start process instances from it. Vice versa, you can export process definitions from an Interstage BPM Server and import them into the Interstage BPM Studio. After completing your modifications, you transfer the modified process definition again to the Interstage BPM Server.

XPDL is used to exchange data between Interstage BPM Studio and Interstage BPM 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 Interstage BPM Studio and Interstage BPM Server.

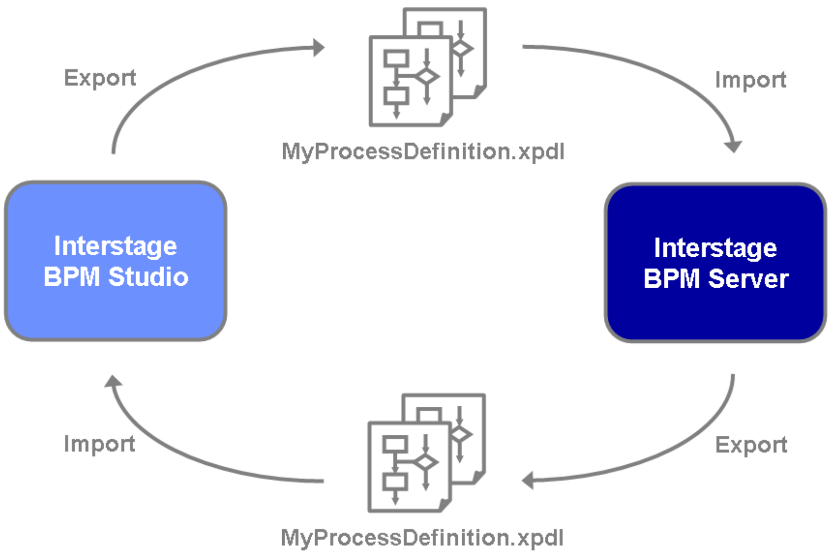


Figure 3: Transferring Process Definitions

Transferring Workflow Application Projects

The Interstage BPM Studio allows you to transfer entire Workflow Application projects between your workspace, the local file system, and the Interstage BPM Server. To facilitate the transfer, projects are packaged into `.bar` files. 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 Interstage BPM Console for deploying the application on the Interstage BPM Server and thus make the application available for public use.

The name of a Workflow Application project must clearly identify the application.

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

- **Download projects from an Interstage BPM 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 *Downloading Workflow Application Projects From a Server* on page 103.
- **Upload projects to an Interstage BPM Server**: This function allows you to upload a local Workflow Application project from your workspace to a remote server. Similarly to the **Download** function, you can upload entire projects or selected components. For more information, refer to section *Uploading Workflow Application Projects to a Server* on page 109.
- **Import projects from the local file system**: 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 *Importing Workflow Application Projects* on page 97.
- **Export projects to a file**: 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 Interstage BPM Server. For more information, refer to section *Exporting Workflow Application Projects* on page 100.

The following figure shows how Workflow Application projects can be transferred between Interstage BPM Studio and Interstage BPM Server.

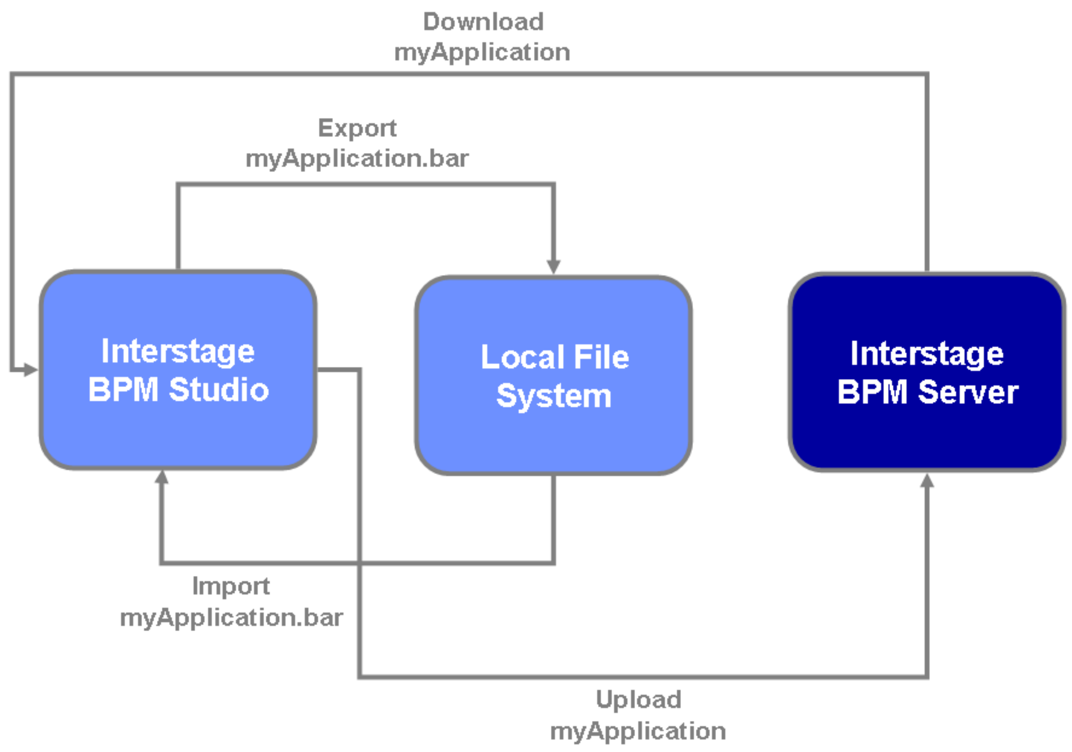


Figure 4: Transferring Workflow Application Projects

1.14 Online Editing: Server Projects

In addition to the possibility of working offline in the Interstage BPM Studio as a standalone tool, you can work directly on a connected Interstage BPM Server. Process definitions that you create with Interstage BPM Studio are stored directly in the database located on the Interstage BPM Server. You can also transfer process definitions created locally directly to the server.

You can either directly connect to an Interstage BPM Server through default tenant in non-SaaS mode or connect to a tenant on that server in SaaS (Software as a Service) mode. Refer *Interstage BPM Studio Features* on page 17 for more information about SaaS and non-SaaS modes.

Tenants can have multiple applications. You can access the applications for a particular tenant using the specific Base URLs. Refer *Creating Server Projects* on page 117 to know more about server connection and setting for multiple tenants.

1.15 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 estimated cost 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 *Defining a Simulation Scenario* on page 378.

You can simulate process definitions that you are currently using, as well as process definitions that you created some time ago and transferred to the server. This process of simulating old data from a server is called "historical simulation". Refer to section *Using Simulation Values From History* on page 380 for more details.

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

2 Getting Started

This chapter explains how to start Interstage BPM Studio, model your first process and exit Interstage BPM Studio.

2.1 Starting Interstage BPM Studio

To start Interstage BPM Studio:

1. Click **Start** on the Windows Task Bar.
2. Select **Programs** or **All Programs** > **Interstage Business Process Manager Studio** > **Studio**.
3. In the **Workspace Launcher** dialog, select a location for your workspace.

   The workspace is a file system folder where your work will be stored.
4. Click **OK**.

The **Workspace Launcher** dialog has an option to turn off the prompting for a workspace (**Use this as the default and do not ask again** check box). To turn it back on, select **Window** > **Preferences** and then **Interstage BPM Studio** > **Startup and Shutdown**. Select **Prompt for workspace on startup**.

Note: Only one instance of Interstage BPM Studio can run at any one time. You cannot start multiple instances.

Note: If you have upgraded from a previous version of Interstage BPM Studio and have installed this version in the same directory as the previous version, you might find that, for example, the online help provided with the Interstage BPM Studio is outdated. Proceed as follows to update all data used by Interstage BPM Studio:

1. Exit Interstage BPM Studio.
2. Click **Start** on the Windows Task Bar and open **Programs** or **All Programs** > **Interstage Business Process Manager Studio** > **Studio**.
3. Right click the **Studio** menu option and select **Properties**.
4. In the **Studio Properties** dialog, add the `-clean` option at the end of the entry in the **Target** field, for example:

   `C:\fujitsu\InterstageBPM_studio\bin\IBPMStudio.exe -clean.`
5. Click **OK**.
6. Start the Interstage BPM Studio as described in this section.

Note: In Interstage BPM Studio, if you use Process Simulation while editing QuickForms using the Ajax Page Editor, you need to do the following setting:

1. Exit Interstage BPM Studio.
2. Edit the file `<Interstage_Studio_Installation_Directory>\bin\IBPMStudio.ini` to add the following option at the end of the command variable: `-XX:MaxPermSize=256m`
3. Start Interstage BPM Studio.

Note: To configure Studio to access HTTPS secured console, set up the Java Virtual Machine (JVM) used by Studio as follows:

1. Verify the file `<Interstage_Studio_Installation_Directory>\bin\IBPMStudio.ini`to find JVM.
2. Add certificates needed to connect web server to the key store of the JVM using the command keytool. For example, `keytool -import -trustcacerts -file C:\Temp\ServerCertificate.cer -keystore C:\PRG\Java\jre1.5.0_11\lib\security\cacerts`. This will ensure that the web server certificate gets added to the JVM's key store.

2.2 Modeling Your First Process

This section provides a quick walk-through of how to model a simple process. As an example, you will be modeling the basics of a bank loan approval. In this process, an agent gets a loan request and passes this on to an underwriter for verification. Once the underwriter approves the request, an accountant issues the check and notifies the agent to contact the customer.

To model your first process:

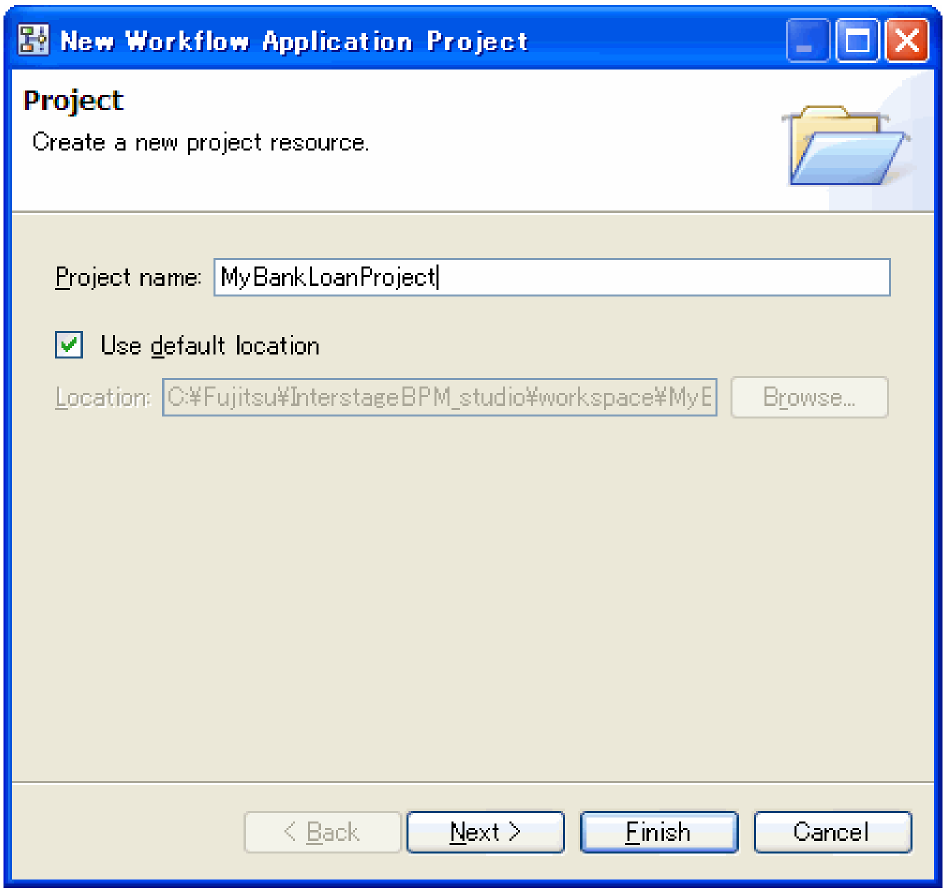
1. Make sure that you have started the Interstage BPM Studio.

   First, you create a project where your work will be stored.

2. To create a project:

   a) Select **File** > **New** > **Project** > **Application**.

b) In the **Project name** field, type `MyBankLoanProject` as the name.



Project name: MyBankLoanProject

✓ Use default location

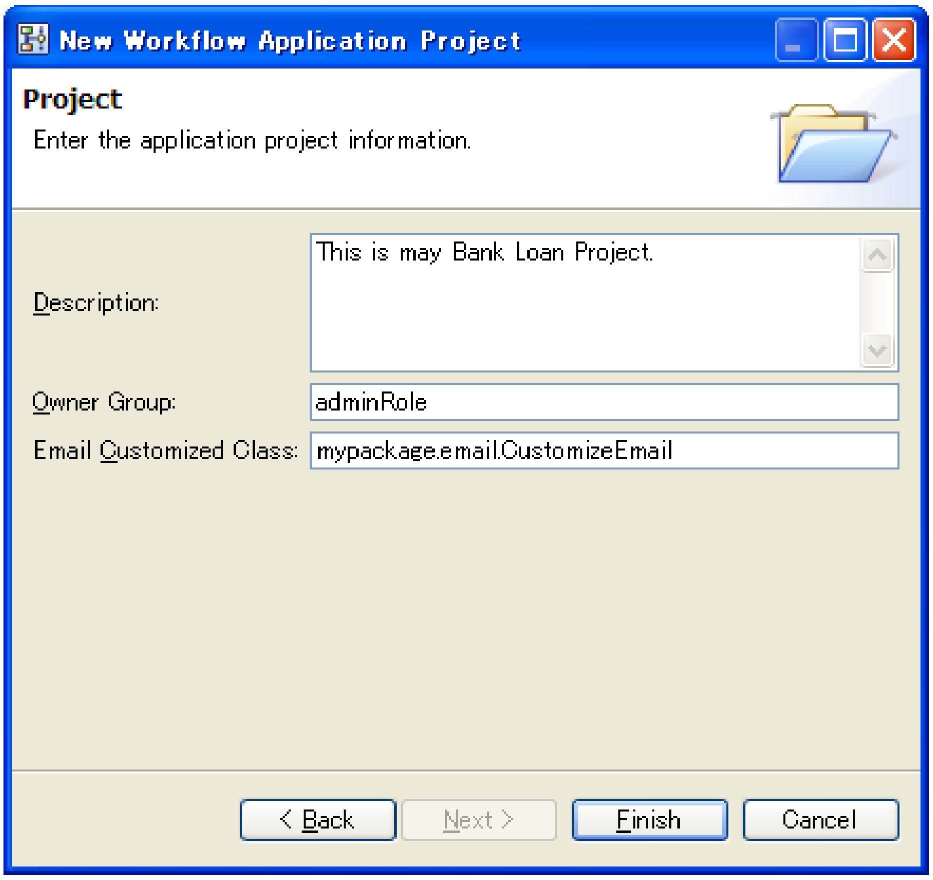
Location: C:¥Fujitsu¥InterstageBPM_studio¥workspace¥MyE

Figure 5: Creating a Project

c) Click **Next**.

d) In the **Project** dialog, type in a description for the new project and the name of the project owner.

The following figure shows the **Project** dialog with the specified information:



Project

Enter the application project information.

Description: This is may Bank Loan Project.

Owner Group: adminRole

Email Customized Class: mypackage.email.CustomizeEmail

Figure 6: Providing project information

e) Click **Finish**.

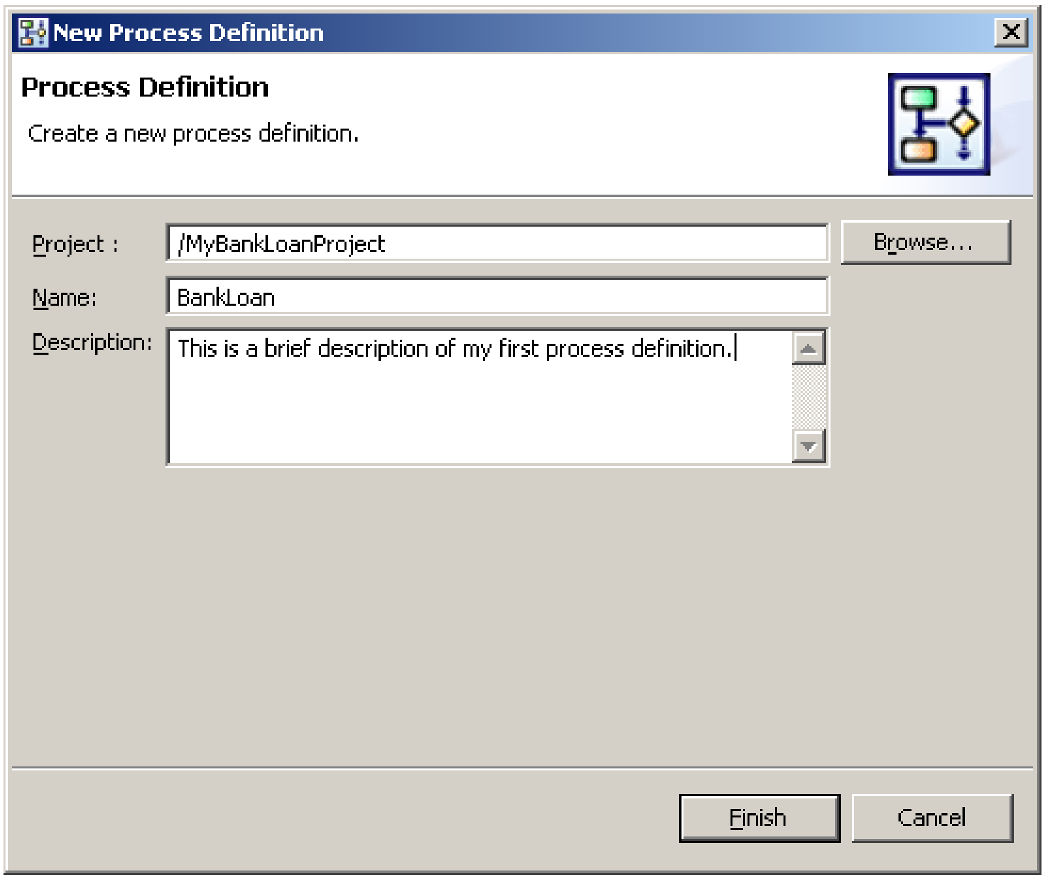
Next, you create a process definition.

3. To create a process definition:

   a) Select **File** > **New** > **Process Definition**.

   b) In the **New Process Definition** dialog, click **Browse**. Select the project that you created, and click **OK**.

c) Type `BankLoan` as the name for your process definition in the **Name** field and enter a brief description in the **Description** field.



Project : /MyBankLoanProject

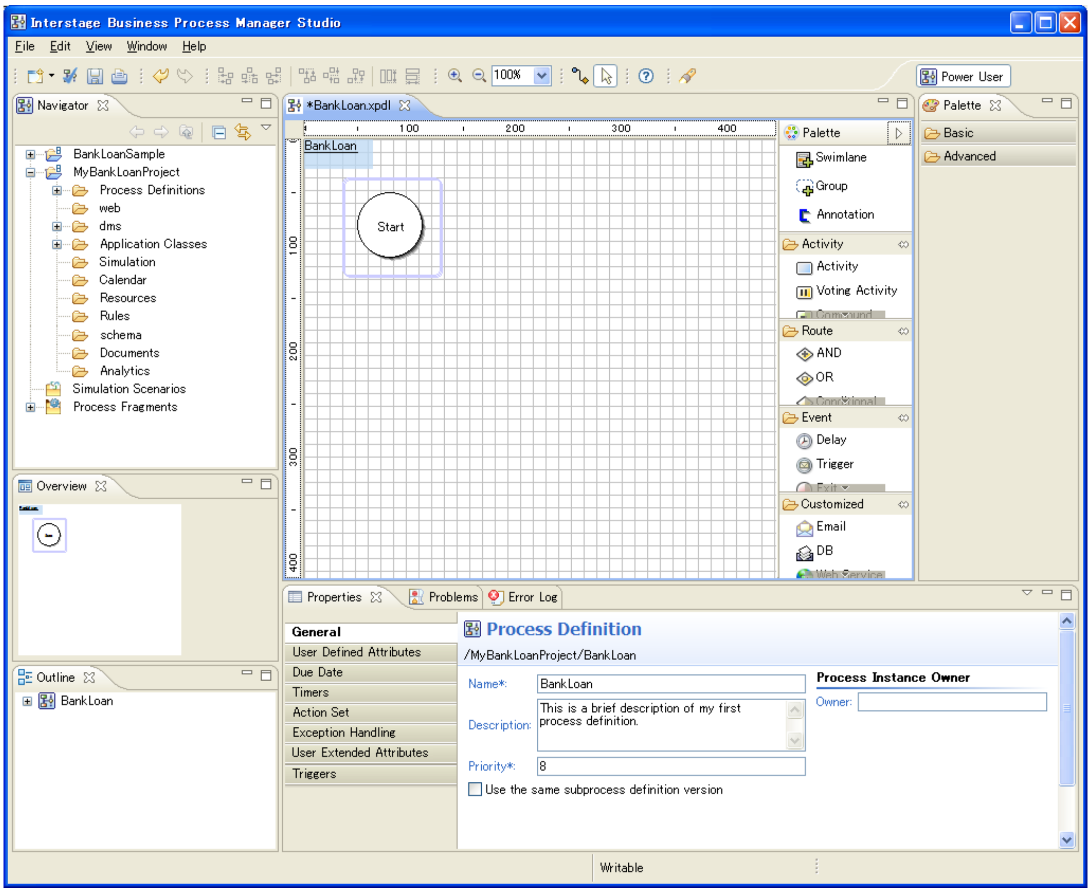
Name: BankLoan

Description: This is a brief description of my first process definition.

Figure 7: Creating a Process Definition

d) Click **Finish**.

Your process definition is now listed in the Navigator view. A Process Definition editor is opened and a Start Node is automatically added. You can now change the name that identifies the process definition and provide a new description.



Process Definition

/MyBankLoanProject/BankLoan

Name*: BankLoan

Description: This is a brief description of my first process definition.

Priority*: 8

☐ Use the same subprocess definition version

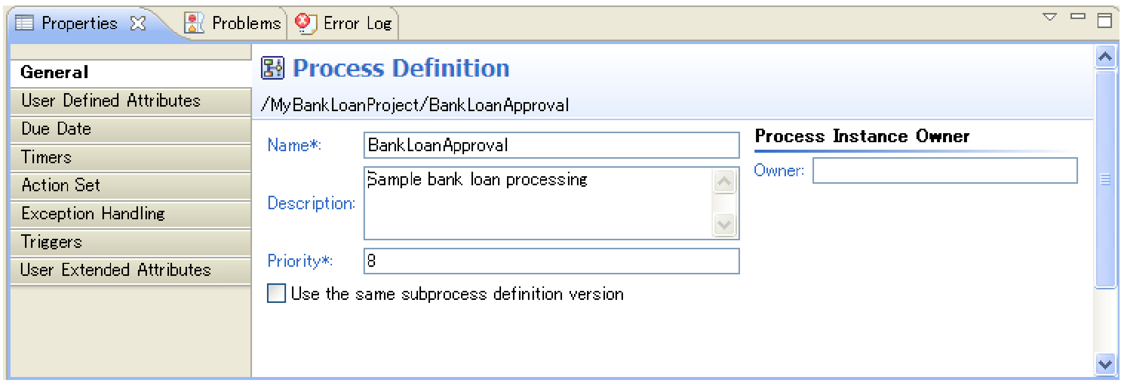
Process Instance Owner

Owner:

Figure 8: Getting Started

4. To change the process definition's name and provide a description:
   a) In the Navigator view, double-click the process definition.

b) In the **General** tab of the Properties view, type `BankLoanApproval` as the name and `Sample bank loan processing` as a description.



/MyBankLoanProject/BankLoanApproval

Name*: BankLoanApproval

Description: Sample bank loan processing

Priority*: 8

☐ Use the same subprocess definition version

Owner:

Figure 9: Defining Name and Description

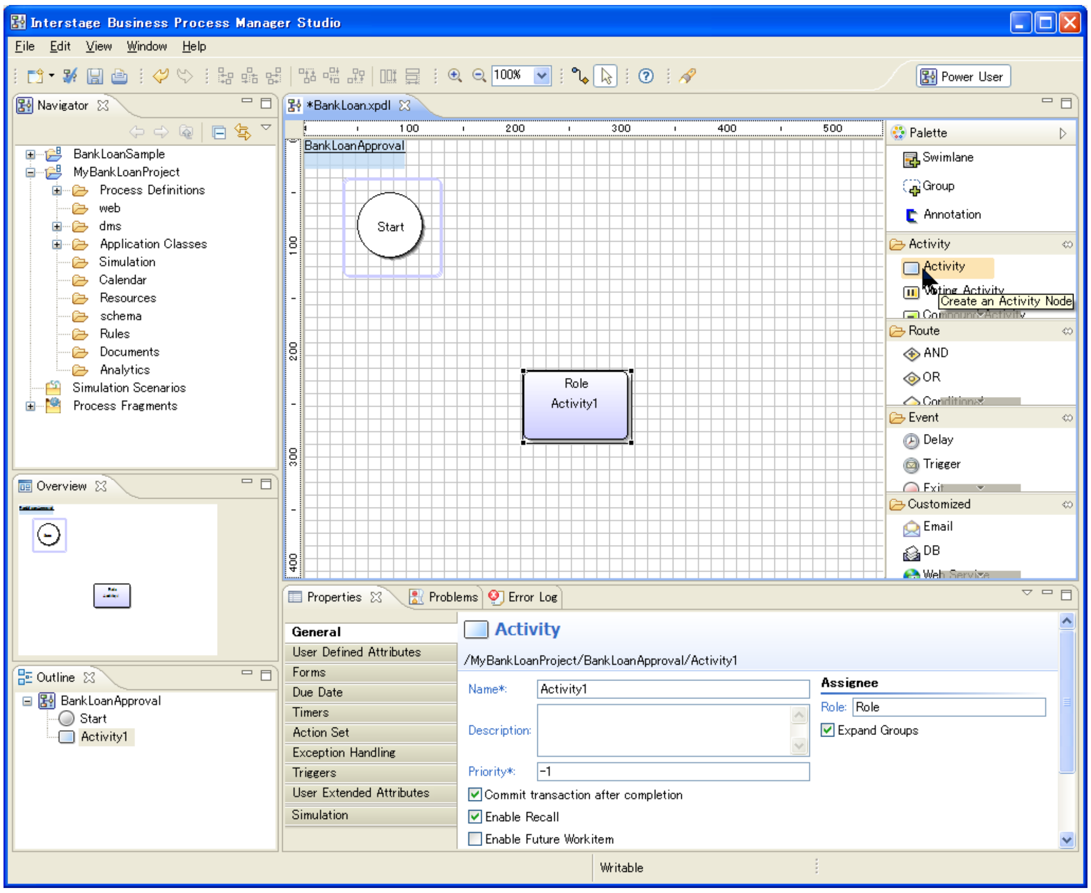
Next, you add a first Activity Node.

5. To add an Activity Node:

a) Open the palette by clicking the Show Palette button on the palette.

b) Click the **Activity** button.

c) In the Process Definition editor, point to the area where you want to place the Activity Node. Click to add it.



Activity

/MyBankLoanProject/BankLoanApproval/Activity1

Name*: Activity1

Description:

Priority*: -1

[✓] Commit transaction after completion

[✓] Enable Recall

[ ] Enable Future Workitem

Assignee

Role: Role

[✓] Expand Groups

Figure 10: Adding an Activity Node

You can now specify the name of the activity and the users that will be performing it.

6. Click the node's name, type `Submit Loan Request` as the new name, and click **Enter** to confirm the name.

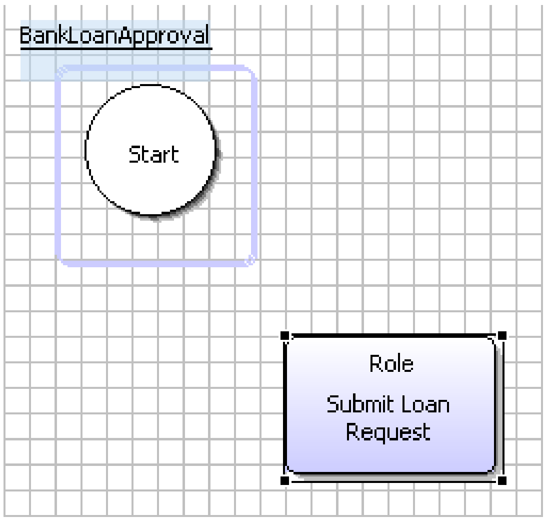
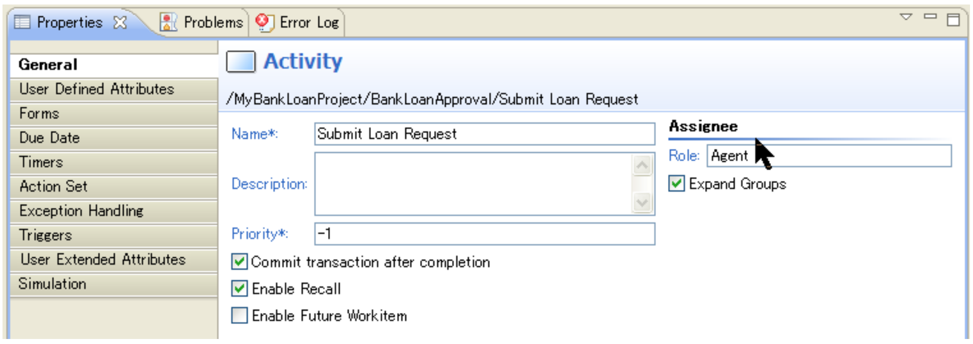


Figure 11: Defining the Activity's Name

7. In the **General** tab of the Properties view, type `Agent` in the **Role** field in the **Assignee** area.



/MyBankLoanProject/BankLoanApproval/Submit Loan Request

Name*: Submit Loan Request

Description:

Priority*: -1

☑ Commit transaction after completion

☑ Enable Recall

☐ Enable Future Workitem

Assignee

Role: Agent

☑ Expand Groups

Figure 12: Defining the Role

The Activity Node should now look like this:

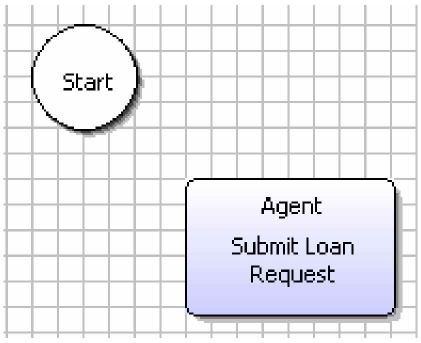
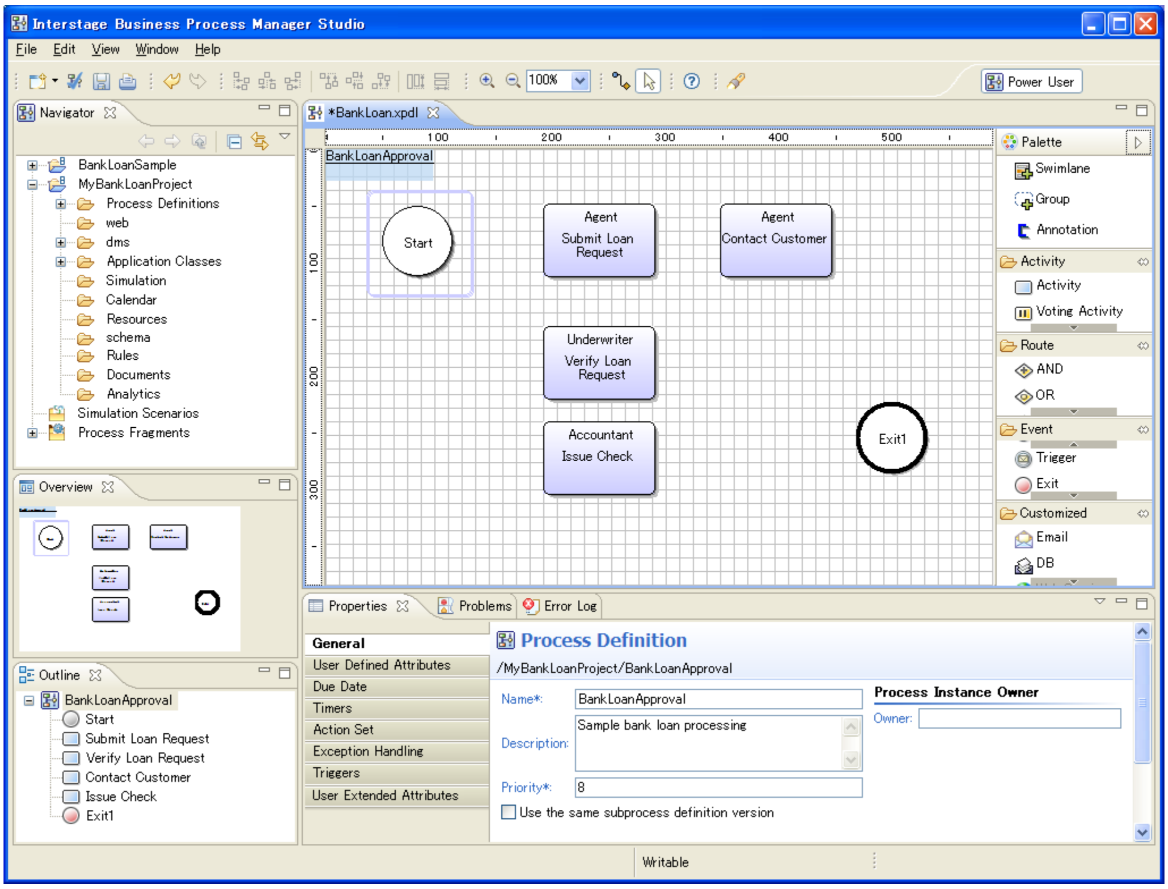


Figure 13: Your First Activity Node

8. Add three more Activity Nodes for the following activities:
   - Activity name: `Verify Loan Request`, Role: `Underwriter`
   - Activity name: `Issue Cheque`, Role: `Accountant`
   - Activity name: `Contact Customer`, Role: `Agent`
9. Add an Exit Node.

This is how your process definition should look now:



Process Definition

/MyBankLoanProject/BankLoanApproval

Name*: BankLoanApproval

Description: Sample bank loan processing

Priority*: 8

[ ] Use the same subprocess definition version

Process Instance Owner

Owner:

Figure 14: Added all Nodes

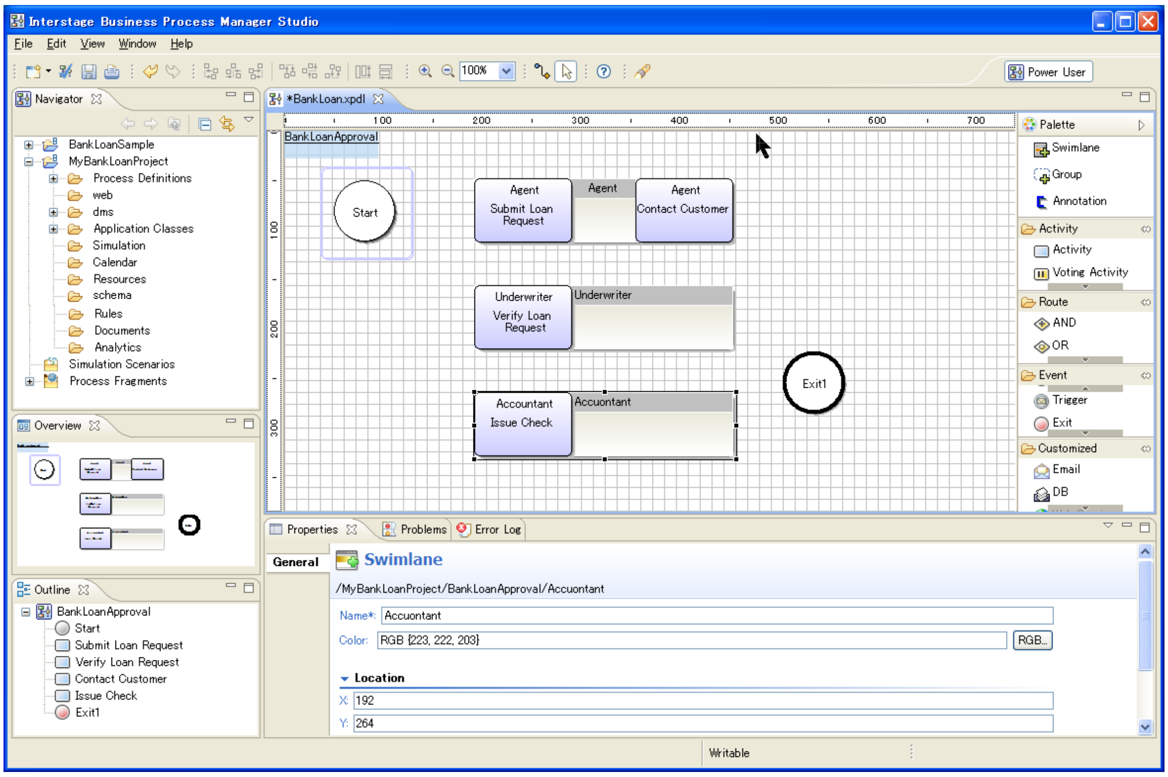
As a next step, you add swimlanes to visually group activities performed by the same Role.

10. To add a swimlane:

    a) Click the swimlane button in the palette.

    b) Drag to draw the swimlane.

c) In the Properties view, type the Role performing the activities as the name.



/MyBankLoanProject/BankLoanApproval/Accountant

Name*: Accountant

Color: RGB {223, 222, 203} RGB...

Location

X: 192

Y: 264

Figure 15: Adding Swimlanes

11. As you added all nodes and swimlanes, you no longer need the palette. Collapse the palette by clicking the Hide Palette button on the palette.

    Next, you connect all the nodes with arrows. To do this, use the Arrow button in the toolbar.

12. To add an arrow:
    a) Click the Arrow button in the toolbar.
    b) Drag the cursor from the source node to the target node.
    c) Point to the place where you want the arrow to end. Release the cursor.

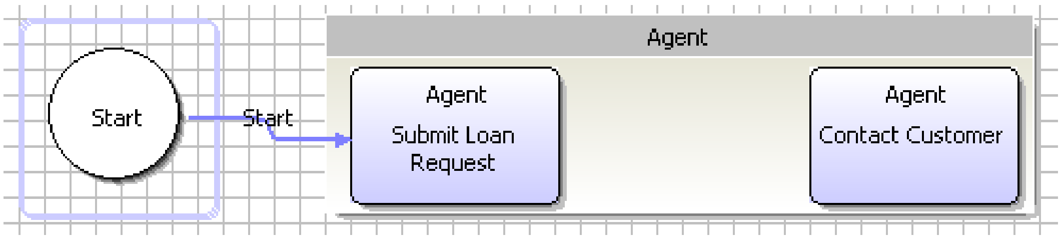


Figure 16: Adding an Arrow

The arrow cursor is still active, so you can draw further arrows.

13. Disable the arrow cursor by clicking the Select button in the toolbar. Then connect the nodes as shown in the following figure. When you are done, press the <Esc> key to disable the Arrow cursor.

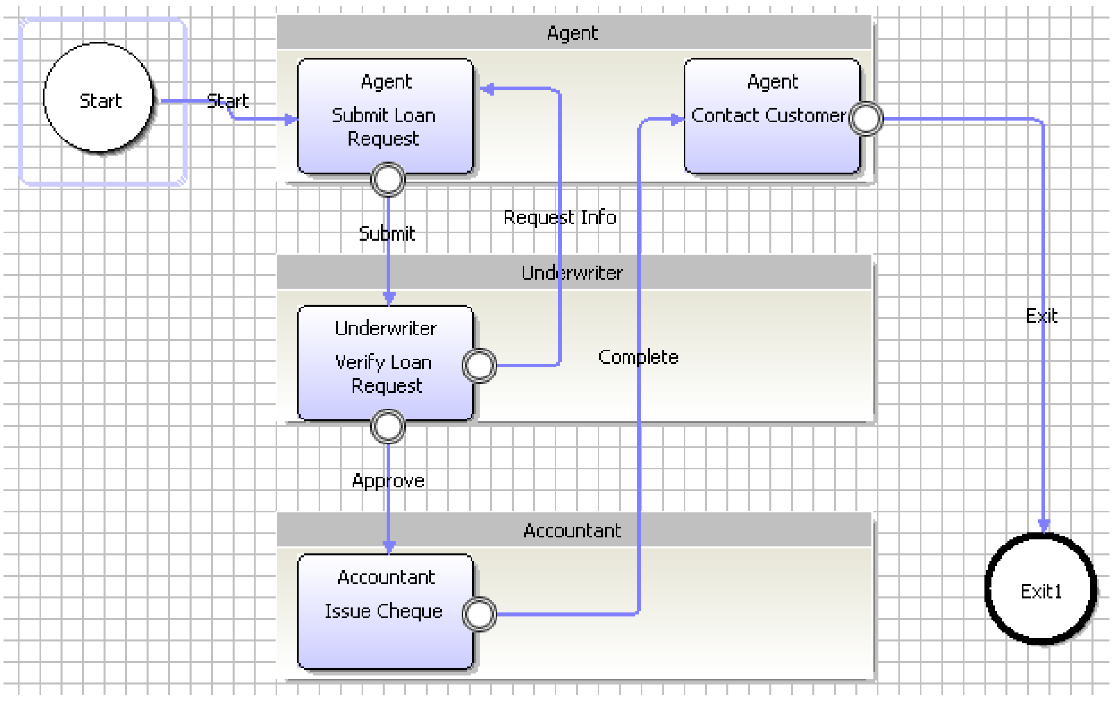
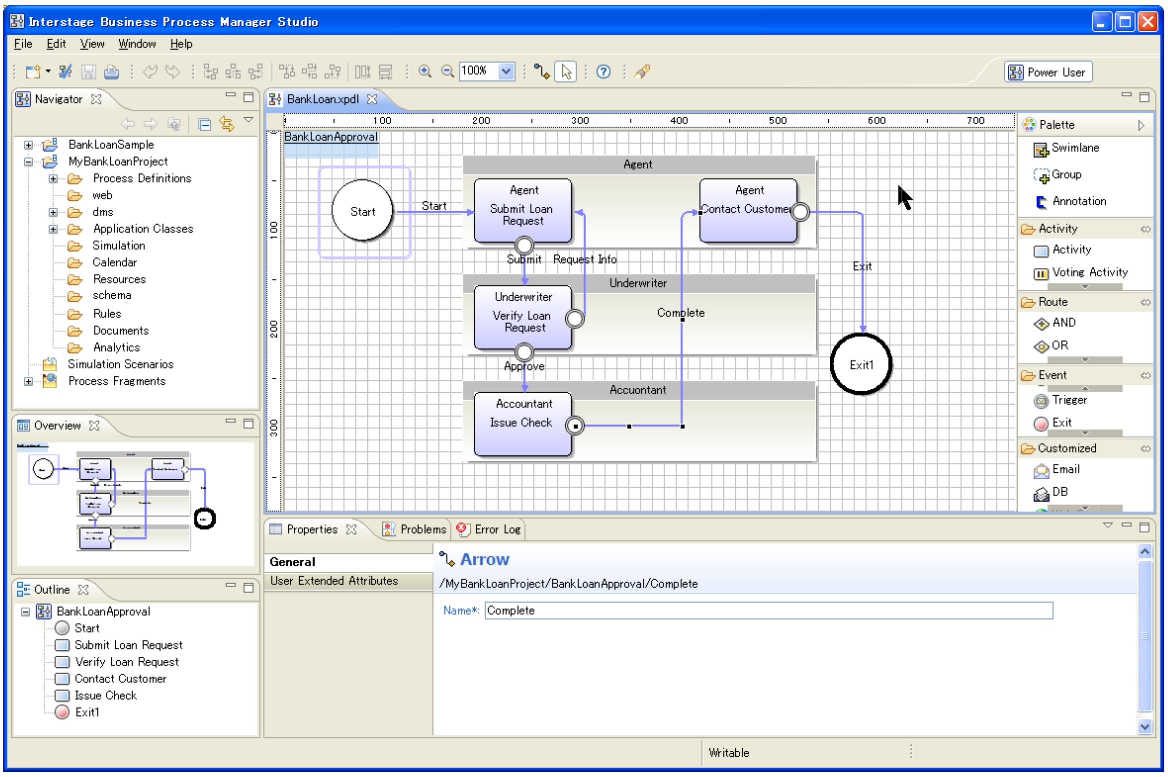


Figure 17: Connecting Nodes

14. Click the arrow's name and type a name for the action associated with the arrow. Perform this step for all the arrows in your process definition.

    If you want to rearrange elements, select them and drag them to a new location. When you move nodes, the arrows connected to them move automatically. You can easily align nodes and swimlanes by selecting them and using the Align buttons in the toolbar.

The sample process definition is now completely defined. A process definition is complete when it has a Start Node, at least one Exit Node, and all nodes are connected with arrows.



Arrow

/MyBankLoanProject/BankLoanApproval/Complete

Name*: Complete

Figure 18: Completing your Process Definition

15. Select **File** > **Save** to save the process definition.

2.3 Using Business User Perspective

You can create new Application projects and Process Definitions in Business User Perspective. You can also edit the existing Application projects and Process Definitions.

Refer *Application View* on page 75 and *Process Outline Editor* on page 76 for more information about various controls in Business User Perspective.

Refer *Creating Workflow Application Projects* on page 85 to know how you can create a new Application project.

You need to create Process Definitions in **Process Outline** view of Business User Perspective.

To create Process Definitions in an Application project in Business User Perspective:

1. Enter a name for the Process Definition in the **Name** field. This is a mandatory field.
2. Create parallel tasks per the requirement:
   a) Select the task (say, A) which you want to define as parallel task of another task.
   b) Click the **Create Parallel Task** button.

      A flag of a particular color is displayed in the **Parallel Task** field of task A.
   c) Create another task (say, B) using the **Add Task** button.

d) Right-click in the **Parallel Task** field of task B and select the flag of the same color as that of the previous task to make this task as a parallel task of the previous task.

Note: To create parallel tasks, the color of the group flag that you select for the parallel tasks, needs to be the same and the tasks need to be at the same level and in a continuous sequence.

Note: You can select a parallel task and click the **Delete Parallel Task** button to remove its parallel task property.

3. Optional: Enter a description of the Process Definition in the **Description** field.
4. Click **Add Task** to add tasks to the **Tasks Table**.

   The task is added in the **Tasks Table**.

Note: Task Name is an editable field. By default, **task0** is added when you click the **Add Task** button. The default task names are incremented by 1 (for example, task1, task2 and so on).

5. Use the various buttons above the **Task Table** to move or delete your tasks.
   - Select a task and click the **Level Down** button to make the selected task the sub-task of the previous task. This button is disabled for the very first task. You can also create sub-tasks to the existing sub-tasks.
   - Select a sub-task and click the **Level Up** button to make the selected sub-task the task.
   - Select a task or a sub-task and click **Move Up** or **Move Down** to move the selected task or sub-task accordingly.
   - Select a task or a sub-task and click **Remove Task** to delete the selected task or sub-task.
6. Enter the assignee name for the task in **Assignee** column.
7. Enter the date when the task is due to be completed (absolute or relative) in the **Due Date** field.
8. Optional: Enter additional information about the task in **Description** field.
9. Enter the duration (in days) for the completion of the task for an assignee in **Duration** field.
10. Save the Process Definition.

2.4 Exiting Interstage BPM Studio

Each time Interstage BPM Studio is exited, your working environment is saved including all settings and windows. The next time Interstage BPM Studio is started, it will appear exactly as it was when it was last exited, except: any process definition located on a server will not be automatically reopened.

To exit Interstage BPM Studio:

1. Select **File** > **Exit**.
2. If there are unsaved changes, a dialog appears where you can select the process definitions, simulation scenarios to save. Select the files and click **OK**.
3. If a process definition or scenario 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 and exit anyway by clicking **YES**.
   - You can cancel saving. You can then fix the errors and try exiting again.

You can also exit Interstage BPM Studio by clicking the window close button (x). A message is displayed asking you to confirm the exiting (**Always exit without prompt** check box). The message box has an option to turn off the prompting. To turn it back on, select **Window** > **Preferences** and then **Interstage BPM Studio** > **Startup and shutdown**. Select **Confirm exit when closing last window**.

3 Understanding and Customizing the User Interface

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

3.1 User Perspectives

Depending on the user requirements, Interstage BPM Studio provides two perspectives for users. These perspectives are:

- Power User Perspective
- Business User Perspective

3.2 Workbench Window of Power User Perspective

By default, Interstage BPM Studio opens in Power User Perspective for the power users of the Interstage BPM Studio. Power users have deep technical knowledge of Interstage BPM Studio and its features. Power users are required to use all the features of the Interstage BPM Studio in order to create detailed Process Definitions and Projects. Using Power User Perspective, users can:

- Create complete Application Projects or Server Projects
- Create complete Process Definitions involving several Nodes, Arrows, Forms, Swimlanes, Activities and so on

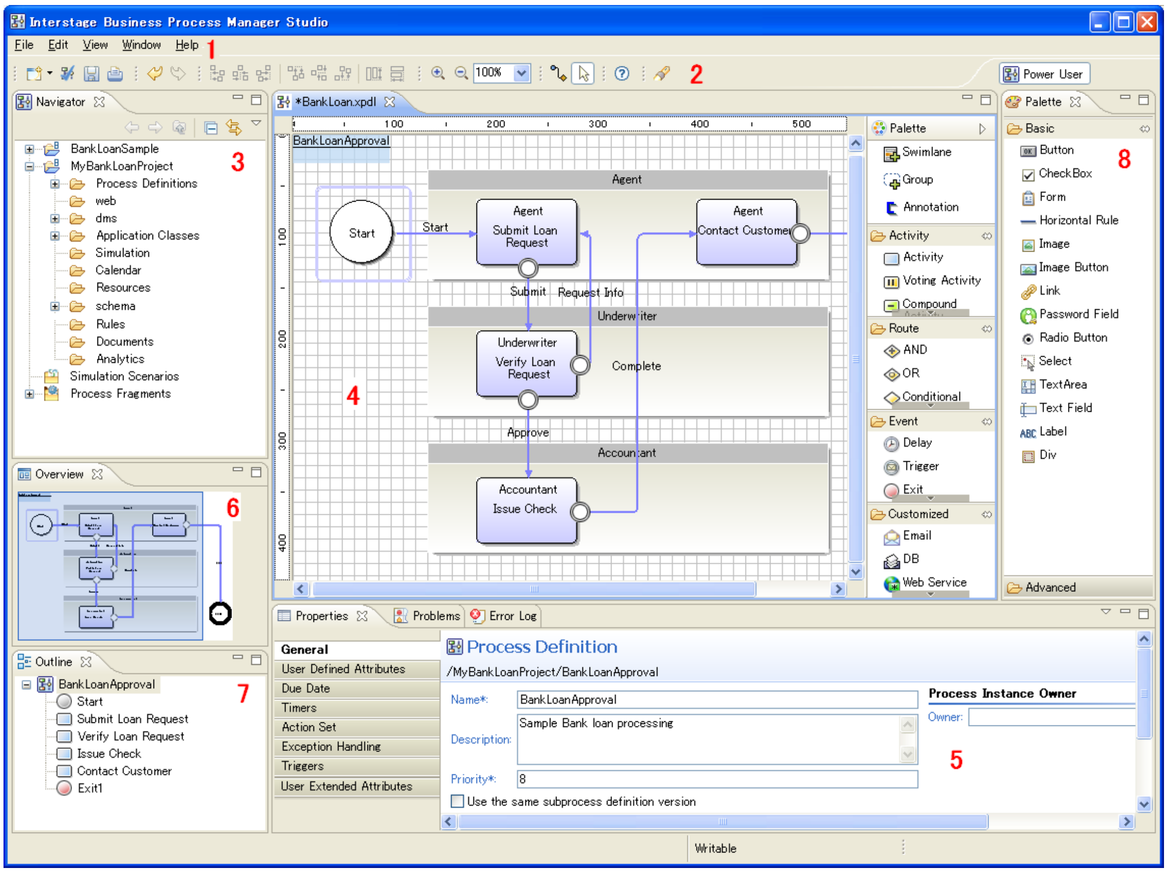
To start Power User Perspective:

1. Click **Window** and select **Open Perspective > Power User**.

Note:

- When you click Power User, you can switch to any perspective using the **Business User** and **Power User** buttons displayed in the right upper corner of the Interstage BPM Studio.
- It is impossible to open any perspective again when you close both the perspectives by right-clicking the perspective buttons and selecting **Close**. In that case, to open the desired perspective, you need to close the current session of Interstage BPM Studio and open a new one.

The user interface of Power User Perspective comprises a menu bar, a toolbar, various views, and various editors. These components are also referred to as Workbench window. The following figure shows the Workbench window:



/MyBankLoanProject/BankLoanApproval

Name*: BankLoanApproval

Description: Sample Bank loan processing

Priority*: 8

Use the same subprocess definition version

Process Instance Owner

Owner:

Figure 19: Workbench Window of Power User Perspective

Key:

- 1 Menu bar
- 2 Toolbar
- 3 Navigator view
- 4 Process Definition editor, Scenario editor, Ajax Page editor, Resource editor, Decision Table Editor, XML Schema Editor
- 5 Properties view, Problem view, and Error Log view (also used for displaying search results)
- 6 Overview view

- 7 Outline view
- 8 Palette View

The following sections explain these components.

3.2.1 Menu Bar

The menu bar contains menus and menu options with which you access the Interstage BPM 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, or Simulation Scenarios project
- Process definition: located on the local machine (local process definition) or located on a server
- Simulation scenario
- Folder of a Workflow Application project

Menu Option	Description
New > Project	Creates a new project.
New > Process Definition	Creates a new process definition.
New > Scenario	Creates a new simulation scenario.
New > QuickForm	Creates a new QuickForm. Available only for Workflow Application projects.
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 an `fileListenerConf.xml` (fixed name) file for Workflow Application projects.
New > Process Scheduler	Creates `ProcessSceduler.xml` file for Workflow Application projects.
New > Java Actions > Data Source	Creates a `DataSourceDefinition.xml` file where you can define data sources for 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.

Menu Option	**Description**
New > Folder	Creates a new subfolder beneath the selected folder. Available only for Workflow Application projects.
New > File	Creates a new file beneath the selected file. Available only for Workflow Application projects.
New > Process Fragment	Creates Process Fragments to define Process Fragment Project
New > XML Schema	Creates a new XML schema.
Close	Closes active editor.
Close All	Closes all open editors.
Save	Saves the modified data in the active editor.
Save As	Saves the active editors with modified data to another location and/or to another file name.
Save All	Saves all the modifications made to all the open editors.
Refresh	Refreshes the Navigator view.
Print	Prints the process definition displayed in the Process Definition editor.
Switch Workspace	Switches to a different workspace. This options restarts Interstage BPM 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.

Menu Option	Description
Validate	Validates project. Process definitions, scenarios, decision tables and XML schema files are validated. If it detects some errors or warnings, they will be shown in the **Problems** view.
Properties	Displays the properties of selected properties/files.
Exit	Closes Interstage BPM Studio.

Edit Menu

The **Edit** menu opens only if a process definition, a simulation scenario, a QuickForm, or an XML schema is open. Refer *Setting Your Preferences* on page 81 for information about XML schema.

Menu Option	Description
Undo	Undoes the last change to a process definition, a scenario or a QuickForm.
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. In the Ajax Page Editor, when you execute this command in the Source view, the selected range is cut and saved to the clipboard as text data. When you execute this command in the Design view, selected parts are cut and saved to the clipboard as text data.
Copy	Copies selected process definition nodes, swimlanes or groups, or text strings defined in a scenario. In the Ajax Page Editor, when you execute this command in the Source view, the selected range is copied and saved to the clipboard as text data. When you execute this command in the Design view, selected parts are copied and saved to the clipboard as text data.
Paste	Pastes cut or copied process definition nodes, swimlanes or groups, or text strings defined in a scenario. In the Ajax Page Editor, when you execute this command in the Source view, the clipboard data is pasted to the cursor position. When you execute this command in the Design view, the clipboard data is pasted to the Design view and this part is 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, selected range or element is deleted. In the **Navigator** view, removes a project, process definition, scenario, QuickForm resource files, and other files and folders in the Workflow Application project structure. Process definitions of a server project and the Simulation Scenario and Analytics folder cannot be deleted.

Menu Option	Description
Select All	Selects all elements of the process definition or text strings defined in the scenario currently displayed. In the Ajax Page Editor, you can use this command in the Source view. When you execute this command, all the source codes are selected. You can use this command in the Process Definition Editor and the Scenario Editor.
Find/Replace	Opens the [Find/Replace] dialog, a specified character is found or replaced.
Find Next	Selected text String or selected element is found next.
Find Previous	Selected text String or selected element is found previous.
Incremental Find Next	Finds next by incremental mode.
Incremental Find Previous	Finds previous by incremental mode.
Align (Submenu)	Contains menu options for aligning nodes, swimlanes, and groups. You can use this command 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. You can use this command in the Process Definition Editor and the Scenario Editor.
Swimlane Style (Submenu)	Contains menu options for switching between the alignment of the swimlane title (at the top or to the left).
Expand Selection To	You can use this in the Source view of the Ajax Page Editor and XML Schema Editor.
Smart Insert Mode	You cannot use this command.
Content Assist	You can use this in the Source view of the Ajax Page Editor and XML Schema Editor.
Show Tooltip Description	You can use this in the Source view of the Ajax Page Editor and XML Schema Editor.
Word Completion	You can use this in the Source view of the Ajax Page Editor and XML Schema Editor.
Quick Fix	You can use this in the Source view of the Ajax Page Editor and XML Schema Editor.
Refresh	You can use this in the Ajax Page Editor. Refresh the Design view.

Source Menu

The Source menu opens only if a Source view of the Ajax Page Editor/XML Schema Editor is active.

Menu Option	Description
Toggle Comment	Change the caret line and the selected line to comment.
Add Block Comment	Change the caret tag to comment.
Remove Block Comment	Delete a block comment.
Shift Left	Shift to left.
Shift Right	Shift to right.
Cleanup Document	Display the [Cleanup] dialog to correct the tag and the attribute in a document along the rule.
Format	Format the editor contents.
Format Active Elements	Format the active element contents.
Occurrences in File	Search the appearance part 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.
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.

Window Menu

Menu Option	Description
New Editor	Opens various files that are currently displayed in a new editor.
Open Perspective	Allows you to switch between the normal BPM Studio and Interstage Analytics perspectives. The Analytics perspective is only visible if you have installed Interstage Analytics before.

Menu Option	Description
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 editors. Also use this submenu for maximizing active views or 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 Interstage BPM Studio. The preferences include appearance of graphical display settings, editor, navigator, startup and shutdown, and server connection settings

Help Menu

Menu Option	Description
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.
Dynamic Help	Opens a Help view where you can browse for help topics.
About Interstage Business Process Manager Studio	Opens a dialog where you can access version information and technical information about Interstage BPM Studio components.

3.2.2 Toolbar

The toolbar contains buttons for frequently used functions.

Button	Description
📁 ▾	Opens a wizard that allows you to create a new project and project components.
✓	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 modified data in active editors.
🖨	Prints the process definition displayed in the Process Definition editor, or the resource file displayed in the Text editor.
↶	Undoes the last change to active editor.
↷	Redoes the change that has been previously undone.

Button	Description
⇤ ⇹ ⇥	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 and the XML Schema Editor's Design view.
🔍	Decreases the magnification of the Process Definition editor and the XML Schema Editor's Design view.
100% ▾	Sets the magnification of the Process Definition editor and the XML Schema Editor's Design view 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 online Studio User's Guide.
🔦	Allows you to search for folders and files within your workspace, in selected resources, or in enclosing projects.
⬒	In Ajax Page Editor, the Design view and the Source view are displayed at the top and bottom.
◨	In Ajax Page Editor, the Design view and the Source view are displayed at the left and right.
🖼	Only the Design view is displayed.
{≡	Only the Source view is displayed.
⟳	Refresh the Design view of the Ajax Page Editor.
⇧	This indicates whether while editing the Source view, the Design view reflects the changes made in the Source view when it is automatically set. It is displayed when reflecting it automatically with the button pushed.
📷	Outputs an image of XML Schema's Design view to a file.

Note: If you use Interstage BPM Studio as an Eclipse plugin, some additional toolbar buttons are displayed. These buttons allow you to switch between the different perspectives, for example the Java and BPM Studio perspective.

3.2.3 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, XML Schema, 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.

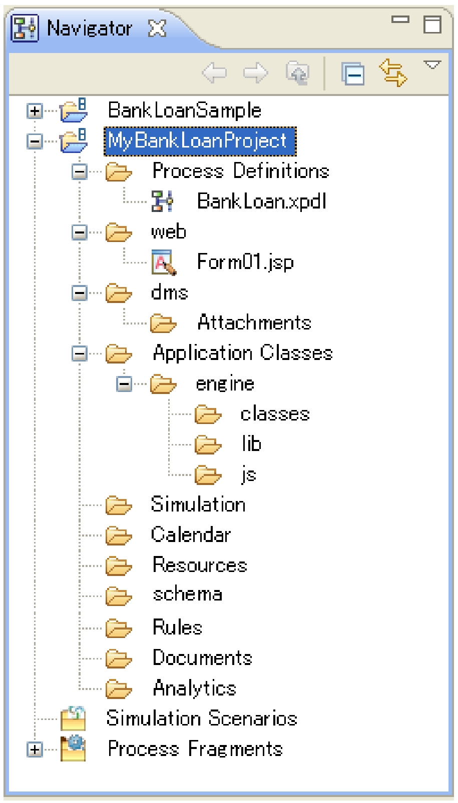


Figure 20: Navigator View

Navigator View Toolbar

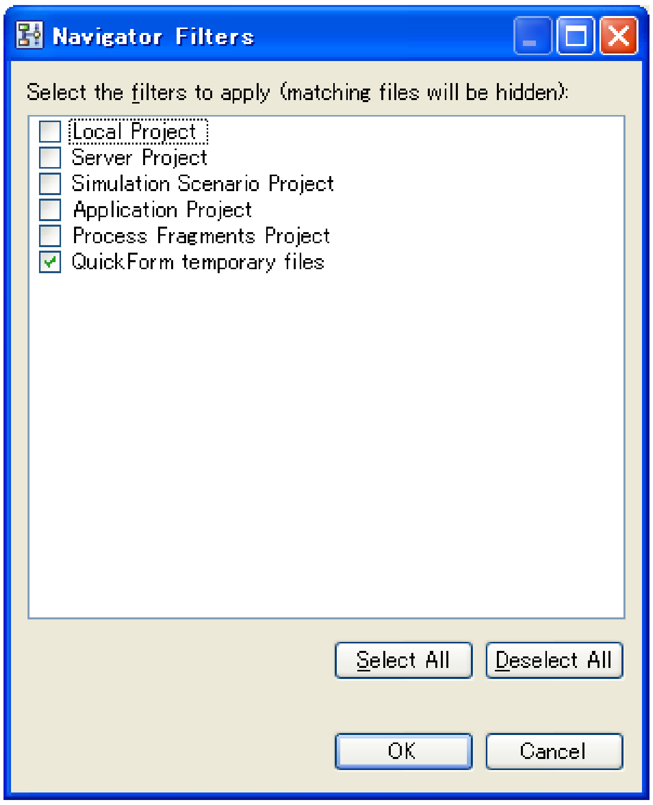
The toolbar of the Navigator view contains buttons for frequently used functions.

Button	Description
⇦	Go back one level in the hierarchy of projects, folders, etc.
⇨	Go forward one level in the hierarchy of projects, folders, etc.

Button	Description
⇧	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:



Navigator Filters

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

Figure 21: Filtering Projects in the Navigator View

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: The **Local Project** check box refers to projects that were created with a release of the Interstage BPM Studio prior to 8.2.

3.2.4 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.

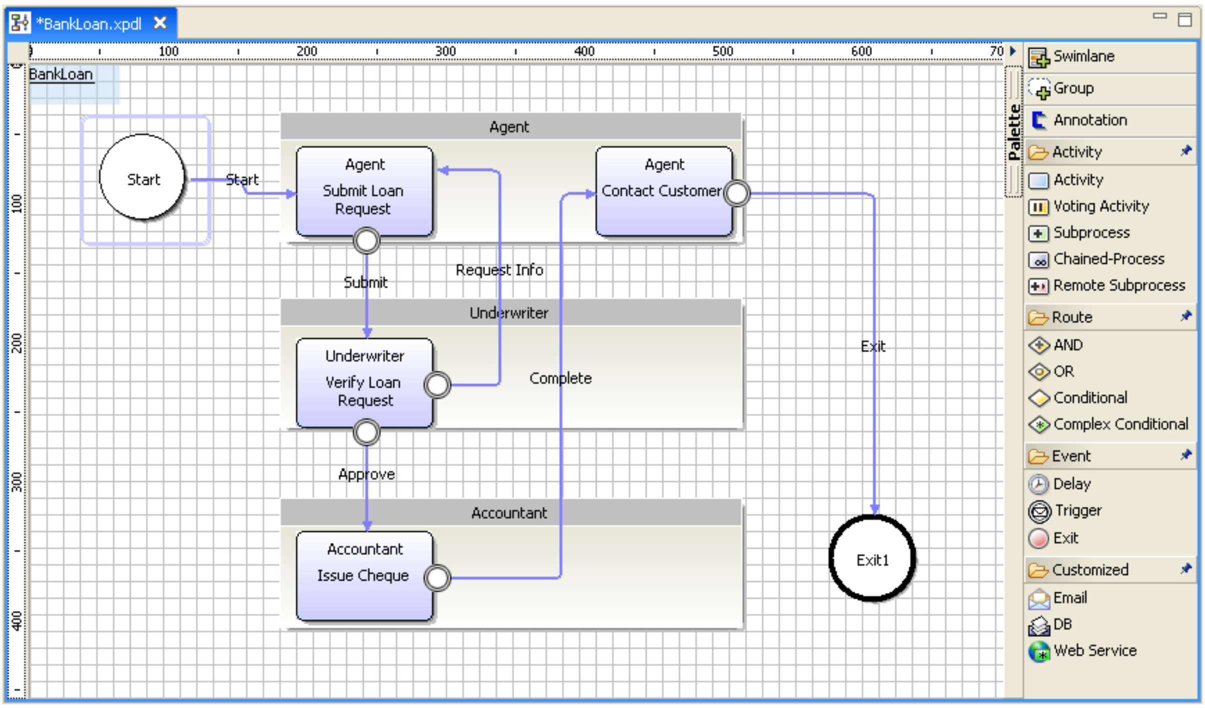


Figure 22: Process Definition Editor

The Process Definition has a palette with buttons for adding nodes, swimlanes or groups, and annotations.

- 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 *Setting Your Preferences* on page 81 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.

3.2.5 Ajax Page Editor and Palette View

There are following two types of QuickForms:

- **QuickForm (to display or update the UDA Value)**

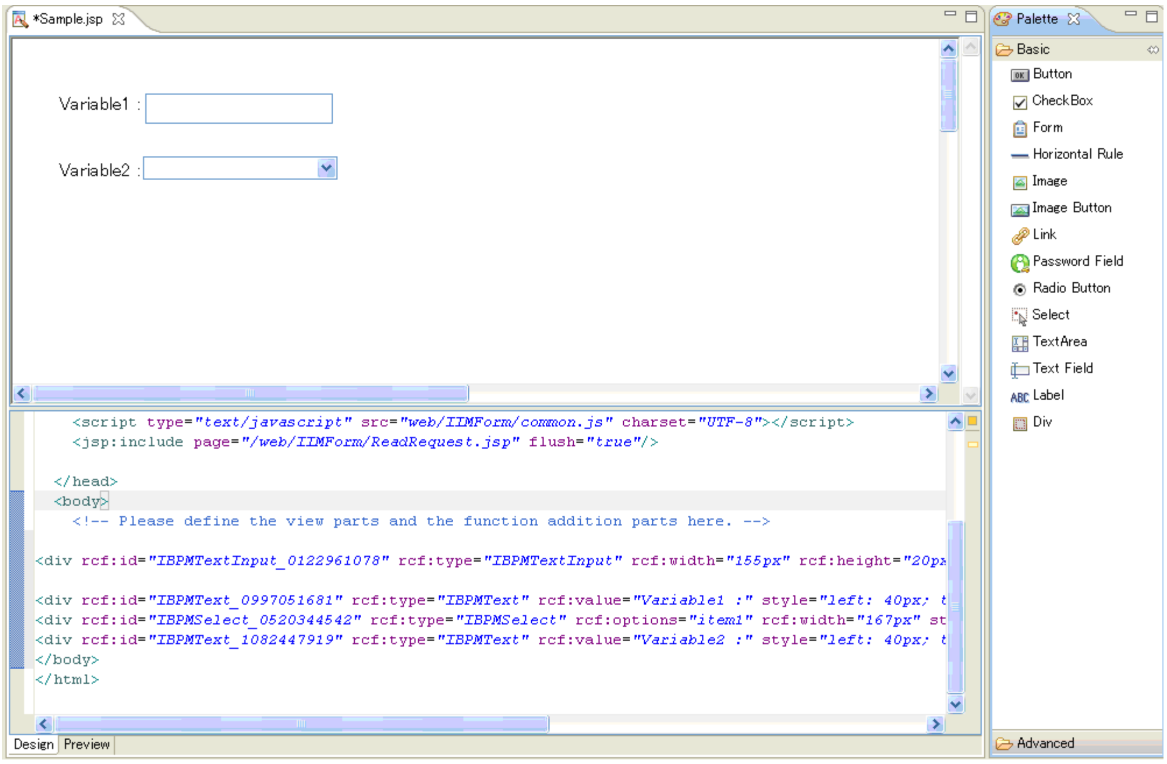
  This is the Form to display or update the UDA value. You can create this form graphically by using Ajax Page Editor. In this Editor, you can freely lay out Item (TextInput, Select, and Radio Button, etc.). Each Item is related to UDA. This file extension is **.jsp**.

- **QuickForm (for compatibility)**

  This QuickForm is a structured, field-based HTML file. You can create these QuickForms using the Interstage BPM Studio as follows: Based on a few settings, Interstage BPM Studio generates an HTML file that contains fields for all UDAs that are to appear on the form. The QuickForm is generated with attributes that make it work with Java Server Pages (JSPs).

  Once you have generated the form, you can customize it according to your own particular needs as long as you do not edit the HTML components needed by Interstage BPM for data and process information display. You can modify the layout using any HTML editor or by modifying the HTML code directly. This file extension is **.qf**.

Ajax Page Editor and Palette View are used to edit a QuickForm (file extension is **.jsp**). When you open a QuickForm, the Form Editor is automatically opened. You can open several QuickForms in parallel. Each QuickForm is displayed in a tab. The tab shows the file name of the QuickForm; an asterisk (*) to the left of the file name or name indicates that there are unsaved changes.



```
    <script type="text/javascript" src="web/IIMForm/common.js" charset="UTF-8"></script>
    <jsp:include page="/web/IIMForm/ReadRequest.jsp" flush="true"/>

  </head>
  <body>
    <!-- Please define the view parts and the function addition parts here. -->

<div rcf:id="IBPMTextInput_0122961078" rcf:type="IBPMTextInput" rcf:width="155px" rcf:height="20px

<div rcf:id="IBPMText_0997051681" rcf:type="IBPMText" rcf:value="Variable1 :" style="left: 40px; t
<div rcf:id="IBPMSelect_0520344542" rcf:type="IBPMSelect" rcf:options="item1" rcf:width="167px" st
<div rcf:id="IBPMText_1082447919" rcf:type="IBPMText" rcf:value="Variable2 :" style="left: 40px; t
</body>
</html>
```

Figure 23: Form Editor

3.2.6 XML Schema Editor

XML Schema Editor is used to edit an XML Schema file. XML Schema Editor has the Design view and the Source view. The Design view visually displays the structure of the XML schema. The Source view displays the source code of XML.

When you open an XML Schema file, the XML Schema Editor is automatically opened. You can open several XML Schema files at a time. Each XML Schema file is displayed in a tab. The tab shows the file name of the XML Schema file; an asterisk (*) to the left of the file name indicates that there are unsaved changes.

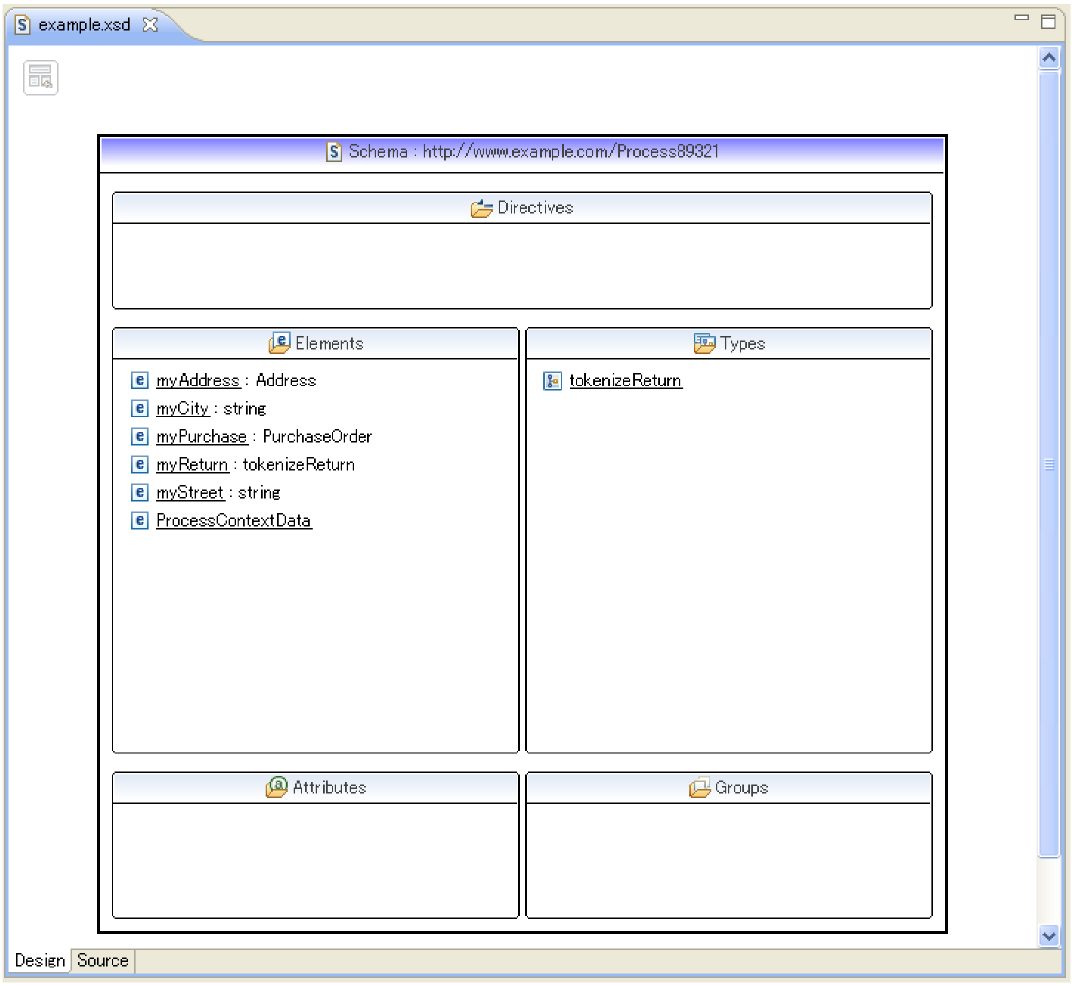


Figure 24: XML Schema Editor

Note:	This XML Schema editor (from WTP 3.0.1) described above is included in Interstage BPM Studio for user's convenience. It is not officially supported by Fujitsu at this time.

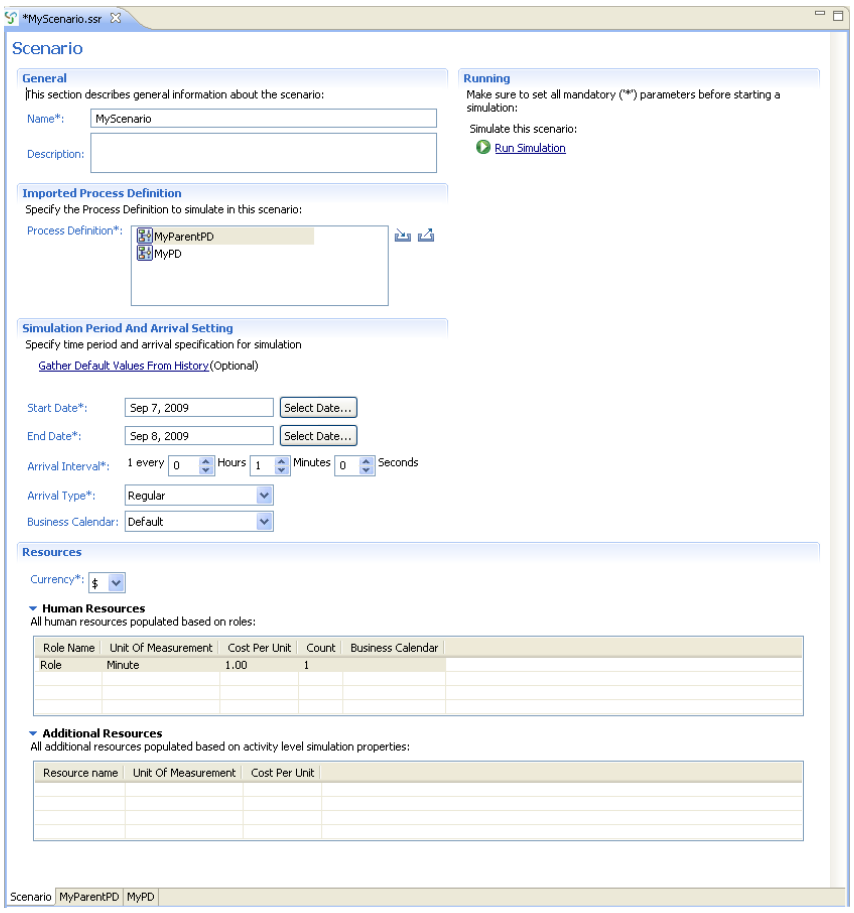
3.2.7 Scenario Editor

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.


*MyScenario.ssr

Scenario

General
This section describes general information about the scenario:

Name*: MyScenario

Description:

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

Simulate this scenario:

Run Simulation

Imported Process Definition
Specify the Process Definition to simulate in this scenario:

Process Definition*: MyParentPD, MyPD

Simulation Period And Arrival Setting
Specify time period and arrival specification for simulation

Gather Default Values From History (Optional)

Start Date*: Sep 7, 2009 Select Date...

End Date*: Sep 8, 2009 Select Date...

Arrival Interval*: 1 every 0 Hours 1 Minutes 0 Seconds

Arrival Type*: Regular

Business Calendar: Default

Resources

Currency*: $

Human Resources
All human resources populated based on roles:

Role Name	Unit Of Measurement	Cost Per Unit	Count	Business Calendar
Role	Minute	1.00	1	

Additional Resources
All additional resources populated based on activity level simulation properties:

Resource name	Unit Of Measurement	Cost Per Unit

Scenario | MyParentPD | MyPD

Figure 25: Scenario Editor

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 the **Run Simulation** command on a selected simulation scenario, the Simulation Controller is displayed just below the Scenario editor.

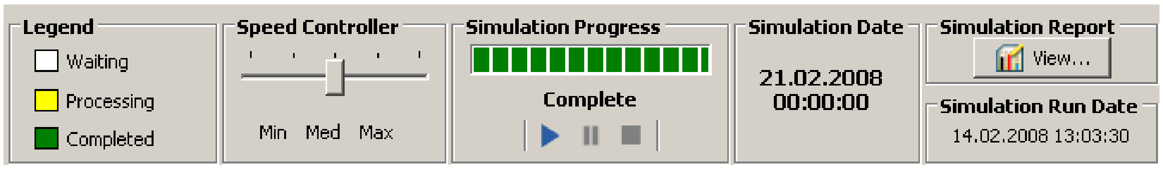


Figure 26: Displaying the Simulation Controller

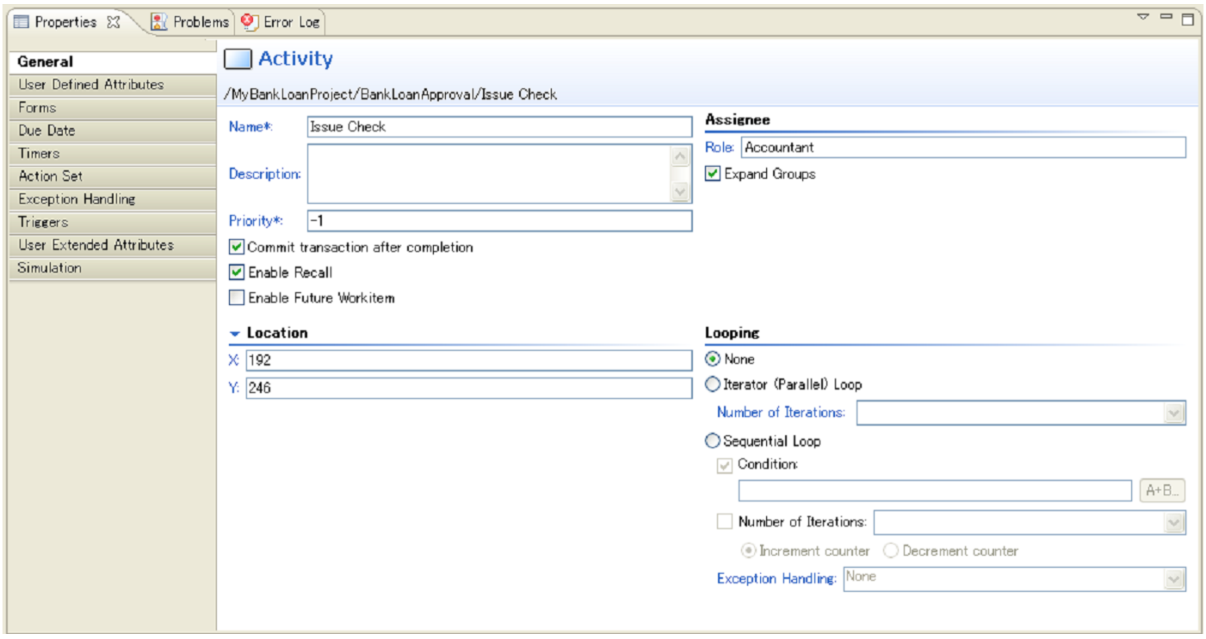
3.2.8 Decision Table Editor

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. Desision Table editor is used to define Decision Table.

3.2.9 Properties View

The Properties view displays either the information of projects and files selected in the Navigator view, or properties of elements selected in each editor. The information on the Properties view varies depending on the currently selected object. You can edit the information if applicable.

For example, 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 UDAs, 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.



Activity

/MyBankLoanProject/BankLoanApproval/Issue Check

Name*: Issue Check

Description:

Priority*: -1

[✓] Commit transaction after completion

[✓] Enable Recall

[ ] Enable Future Workitem

Location

X: 192

Y: 246

Assignee

Role: Accountant

[✓] Expand Groups

Looping

(●) None

( ) Iterator (Parallel) Loop

Number of Iterations:

( ) Sequential Loop

[✓] Condition:

A+B...

[ ] Number of Iterations:

(●) Increment counter ( ) Decrement counter

Exception Handling: None

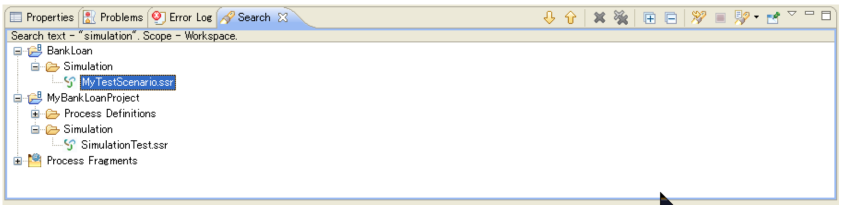
Figure 27: Properties View for an Activity Node

Note: Type of information displayed in Properties view for a process definition, a simulation scenario, or a decision table, changes depending on the display mode (**Name** or **File name**) of the Navigator's preference setting.

3.2.10 Search Results View

The Search Results 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 Results View after searching for the word "simulation":



Search text - "simulation". Scope - Workspace.

```
BankLoan
  Simulation
    MyTestScenario.ssr
MyBankLoanProject
  Process Definitions
  Simulation
    SimulationTest.ssr
Process Fragments
```

Figure 28: Search Results View

3.2.11 Problems View

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

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 Navigator view, you can confirm every validation of project by selecting the Workflow Application project, and right-clicking and selecting **Validate** from the Pop-up menu (right-click menu).

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.

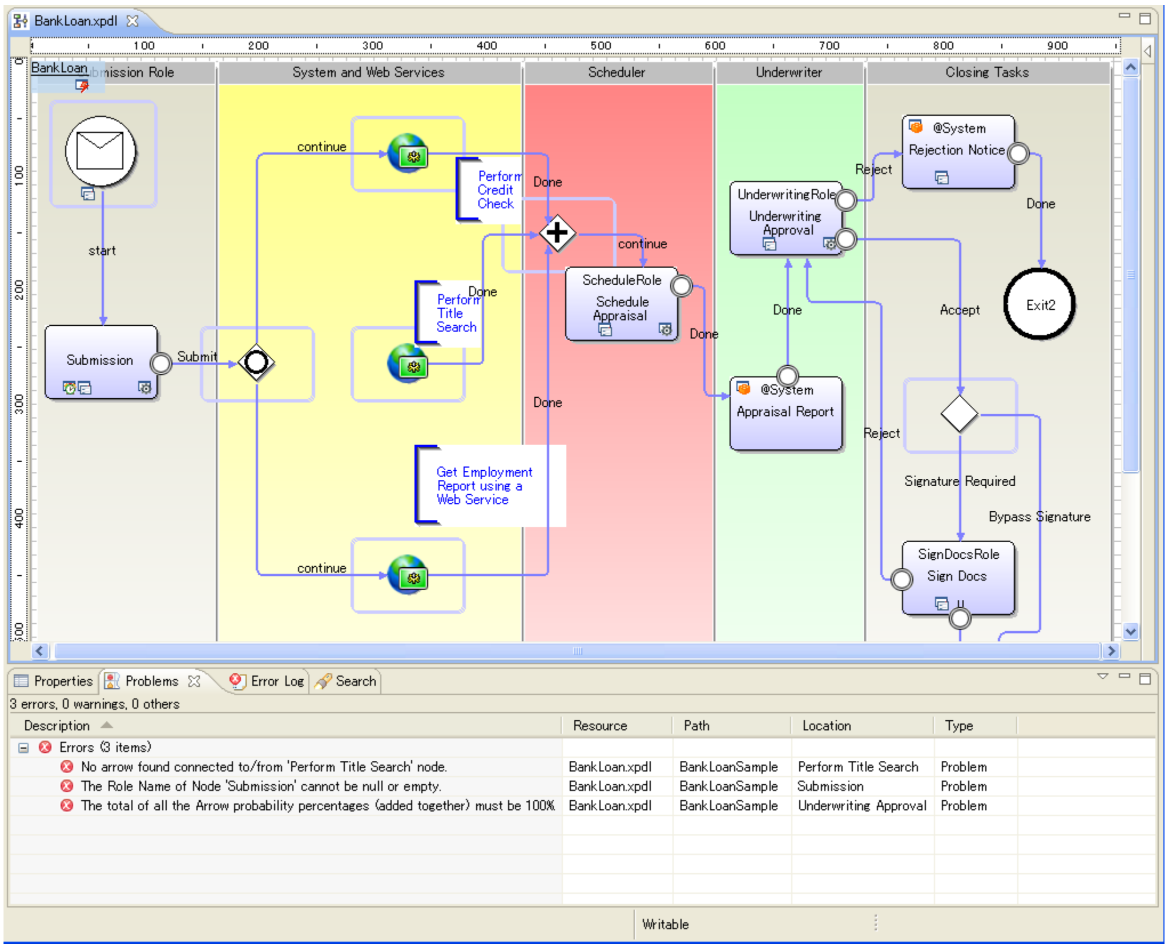
It displays the following information for each error:

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

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. Refer to *Validating Decision Tables for Errors and Warnings* on page 259 for more information about validating Decision Tables for errors and warnings.

Refer *Validating XML Schema File* on page 235 for details about validating XML Schema files.

Note: In Navigator view, when a process definition, scenario, Decision Table, or XML Schema file are deleted, the errors associated with them are also deleted from Problems view.



3 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (3 items)				
No arrow found connected to/from 'Perform Title Search' node.	BankLoan.xpdl	BankLoanSample	Perform Title Search	Problem
The Role Name of Node 'Submission' cannot be null or empty.	BankLoan.xpdl	BankLoanSample	Submission	Problem
The total of all the Arrow probability percentages (added together) must be 100%	BankLoan.xpdl	BankLoanSample	Underwriting Approval	Problem

Figure 29: Problems View for a Process Definition

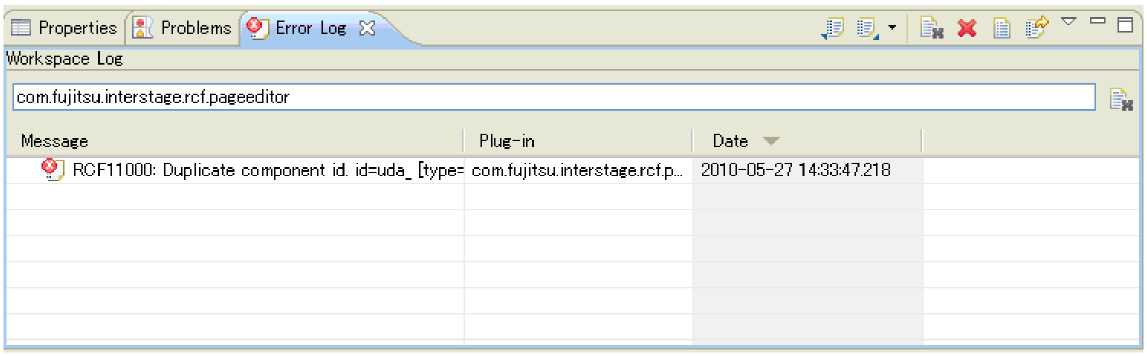
3.2.12 Error Log View

The Error Log view displays the errors or warnings for a QuickForm.

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

Field	Description
Message	Describes the nature of the error and warning.
Plug-in	Displays plug-in name where the error and warning occur.
Date	Displays date when the error and warning occurred.

Note: Please set the following values to the filter: `"com.fujitsu.interstage.rcf.pageeditor"`



com.fujitsu.interstage.rcf.pageeditor

Message	Plug-in	Date
RCF11000: Duplicate component id. id=uda_ [type=	com.fujitsu.interstage.rcf.p...	2010-05-27 14:33:47.218

Figure 30: Error Log View

3.2.13 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.

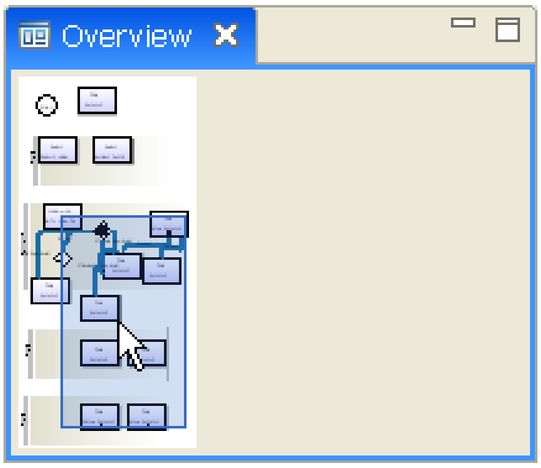


Figure 31: Overview View

3.2.14 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*Nodes* on page 20 for an explanation of the icons. Note that any annotations to Actions or Nodes are not visible in Outline view.

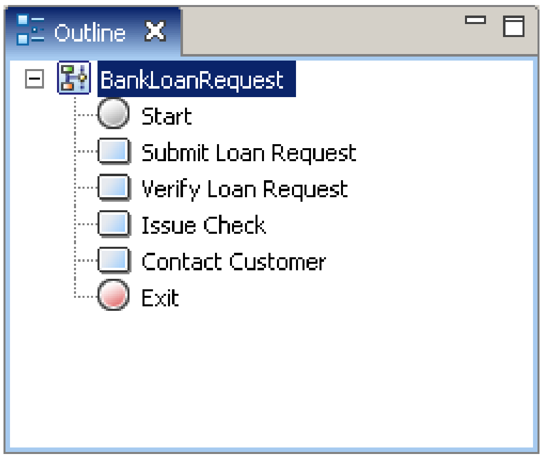


Figure 32: Outline View

If a simulation scenario is active, the Outline view only displays the scenario name. The Outline view also displays the list of widgets on the Ajax Page Editor, when the Ajax Page Editor is activated.

When the XML Schema Editor is activated, the Outline view also displays the list of widgets on the XML Schema Editor.

3.2.15 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 or in an editor and right click.

3.2.16 Displaying Properties

Properties help you specify process definitions, nodes and arrows (for more information on properties and their symbols, refer to section *Property Symbols* on page 24).

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, Interstage BPM 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:

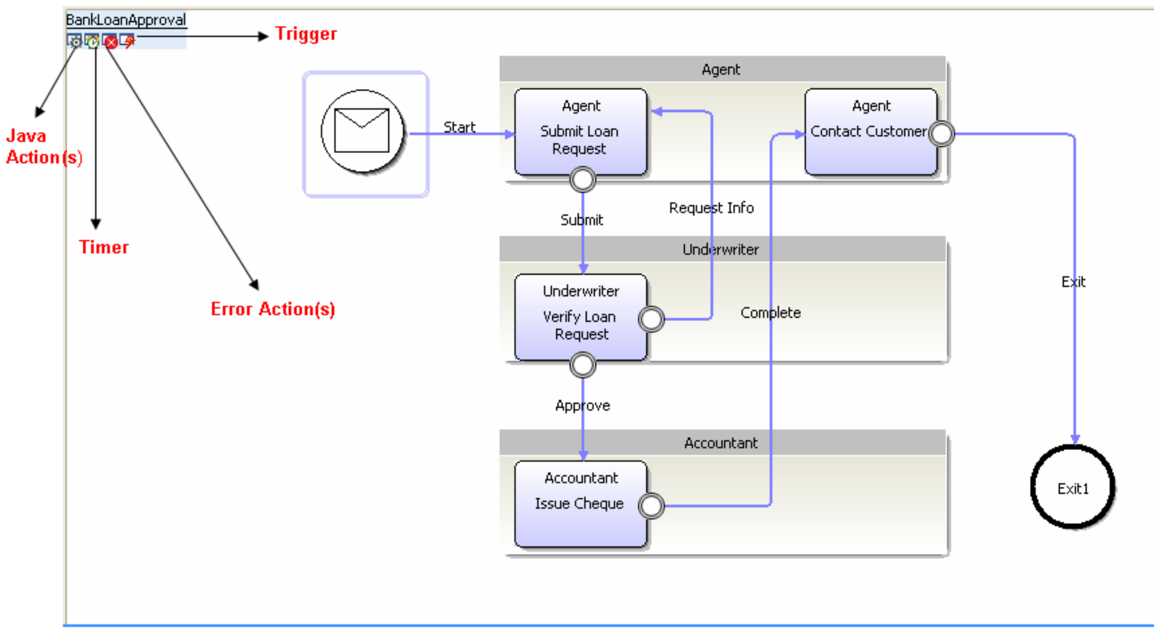


Figure 33: Displaying property symbols for process definitions

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

No.	Symbol	Meaning
1	⚙	A Java Action has been defined. If you see two Java Action symbols, one Owner or Init Java Action and one Commit Java Action have been defined.
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.

- **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 available properties have been set for the Activity Node (triggers, timers, Error Actions, and Forms):

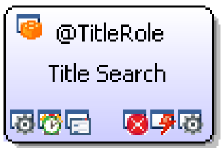


Figure 34: Displaying property symbols for an Activity Node

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

No.	Symbol	Meaning
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.

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).

3.2.17 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**.

3.2.18 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 section *Setting Your Preferences* on page 81 for information on how to set your grid preference.

3.2.19 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**.

3.2.20 Changing Display Mode

Interstage BPM 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:

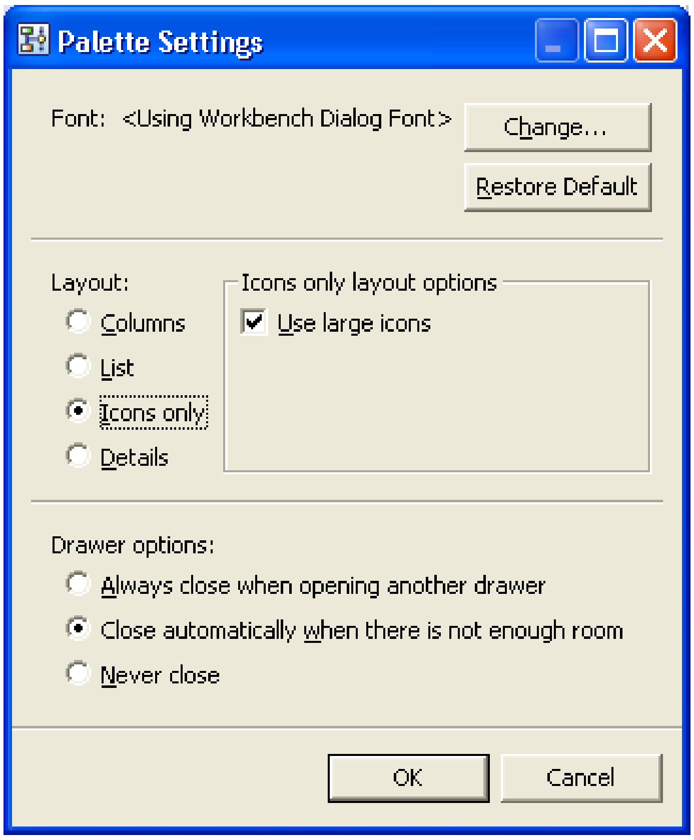
1. Click the Process Definition editor to make it active.
2. Select **View** > **Display Mode**. Select the display mode you want to use.

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

3.2.21 Changing Palette Settings at 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.

To change palette settings, right click the palette and select **Settings** from the pop-up menu. The **Palette Settings** dialog is displayed:



Font: <Using Workbench Dialog Font>

Layout:
- Columns
- List
- [✓ Icons only]
- Details

Icons only layout options
- [✓ Use large icons]

Drawer options:
- Always close when opening another drawer
- [✓ Close automatically when there is not enough room]
- Never close

Figure 35: Palette Settings

1. To change the **Font** settings for the icon texts:
   a) Click **Change**.
   b) Change font properties like font family or font size.
   c) Close the dialog by clicking **OK**.
2. To change the palette **Layout**, select the desired layout, for example **Icons only**. For every layout, you can choose between large and small icons be checking or unchecking the **Use large icons** check box.
3. The buttons in the palette are grouped in drawers. To change **Drawer options**, choose between always closing a drawer when opening another one, automatically closing drawers when there is not enough room, or never closing drawers.
4. Save your settings by clicking **OK**.

5. Depending on your settings, drawers may be closed automatically. To permanently display a drawer, click its Pin Open button.

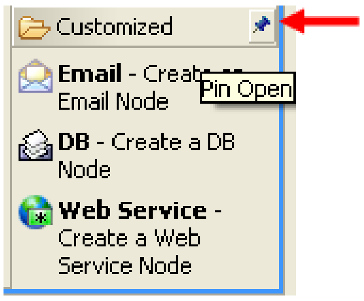
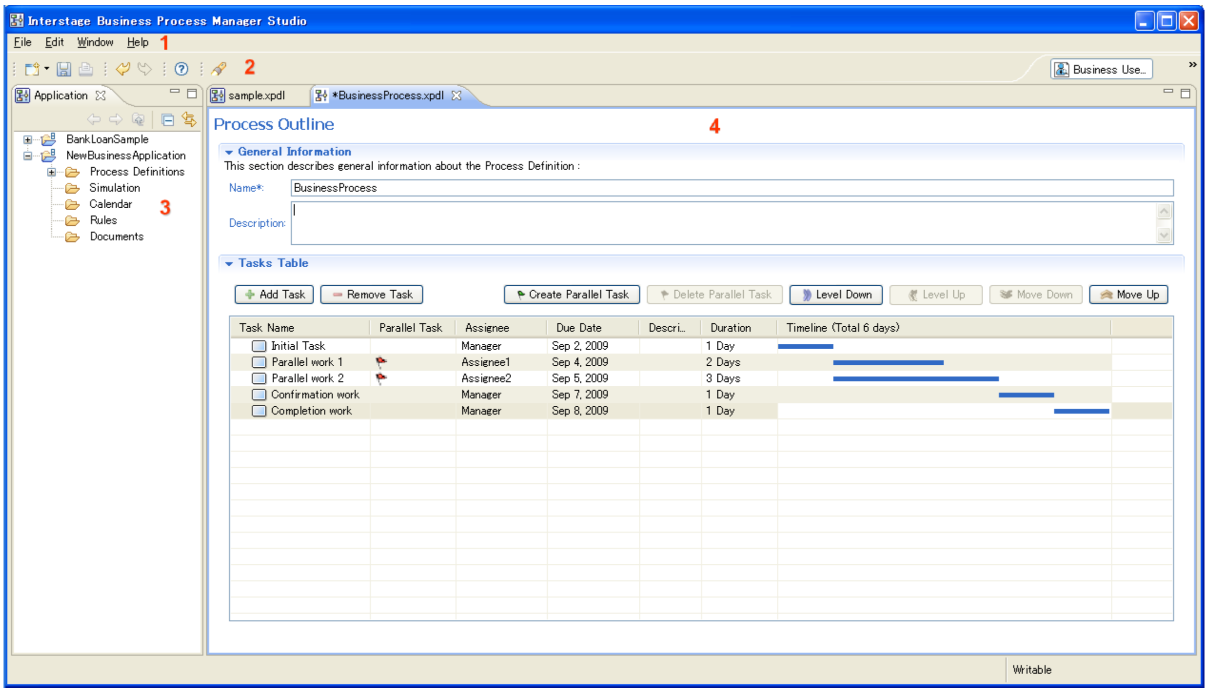

Customized
Email - Create an Email Node
Pin Open
DB - Create a DB Node
Web Service - Create a Web Service Node

Figure 36: Pin Open Button

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

3.3 Workbench Window of Business User Perspective

Business User Perspective of Interstage BPM Studio provides a simple way of creating process to the business users who do not possess deep technical knowledge of Interstage BPM Studio. The user interface of Business User Perspective comprises a menu bar, a toolbar, Application View and Process Outline Editor. These components are also referred to as Workbench Window. The following figure shows the Business User Workbench Window:



Process Outline

General Information

This section describes general information about the Process Definition :

Name*: BusinessProcess

Description:

Tasks Table

Task Name	Parallel Task	Assignee	Due Date	Descri...	Duration	Timeline (Total 6 days)
Initial Task		Manager	Sep 2, 2009		1 Day	
Parallel work 1		Assignee1	Sep 4, 2009		2 Days	
Parallel work 2		Assignee2	Sep 5, 2009		3 Days	
Confirmation work		Manager	Sep 7, 2009		1 Day	
Completion work		Manager	Sep 8, 2009		1 Day	

Figure 37: Workbench of Business User Perspective

Key:

- 1 Menu bar: It is a subset of the menu of Power User Perspective.
- 2 Toolbar: It is a subset of the toolbar of Power User Perspective.
- 3 Application view
- 4 Process Outline editor

Note: When you save an Application project in Business User Perspective, you can also view and edit it in Power User Perspective.

To start Business User Perspective:

1. Click **Window** and select **Open Perspective > Business User**.

Note:
- When you click **Business Perspective**, you can switch to any perspective using the **Business User** and **Power User** buttons displayed in the right upper corner of the Interstage BPM Studio.
- It is impossible to open any perspective again when you close both the perspectives by right-clicking the perspective buttons and selecting **Close**. In that case, to open the desired perspective, you need to close the current session of Interstage BPM Studio and open a new one.

3.3.1 Application View

The **Application** view displays the following:

- **Application project**: Application projects are displayed in the **Application** view under which Process Definitions, Simulation files, Calendar files, Rules files and Document files are displayed. Refer *Modeling Your First Process* on page 34 and *Managing Workflow Application Projects* on page 85 to know more about creating Applications. You can perform the following operations on an Application project:
  1. Create, delete, and rename an Application project
  2. Import, and export an Application project

Note: When you import or export an Application Project, all the resources (**web** folder and **Form** folder and so on) in that particular Applicaiton Project are imported or exported respectively.

- **Process Definitions**: Process Definitions under an Application project are displayed. Refer *Managing Process Definitions* on page 122 to know more about Process Definitions and operations to be performed on them. You can perform the following operations on Process Definitions:
  1. Create, delete, rename, and open Process Definitions
  2. Import, and export Process Definitions to or from Application projects
- **Simulation files**: Simulation files under an Application project are displayed. Refer *Simulating Processes* on page 376 to know more about Simulation. You can perfrom the following operations on Simulation files:
  1. Create, delete, rename, and open Simulation files
  2. Import and export Simulation files to or from Application projects

- **Calendar files**: Calendar files under an Application project are displayed. Refer *Creating Your Own Business Calendars* on page 178 to know more about Calendars. You can perform the following operations on Calendar files:
  1. Create, delete, rename, and open Calendar files
  2. Import and export Calendar files to or from Application projects
- **Rules files**: Rules files under an Application project are displayed. Refer *Workflow Application Projects* on page 28 and *Using Rules Actions* on page 314 to know more about Rules. You can perform the following operations on Rules files:
  1. Create, delete, and rename Rules Set
  2. Create, delete, rename, and open Rules files
  3. Import and export Rules files to or from Application projects
- **Document files**: Document files under an Application project are displayed. You can perform the following operations on Documents files:
  1. Create, delete, rename, and open Documents files
  2. Import and export Documents files to or from Application projects

The following figure shows the **Application** view:

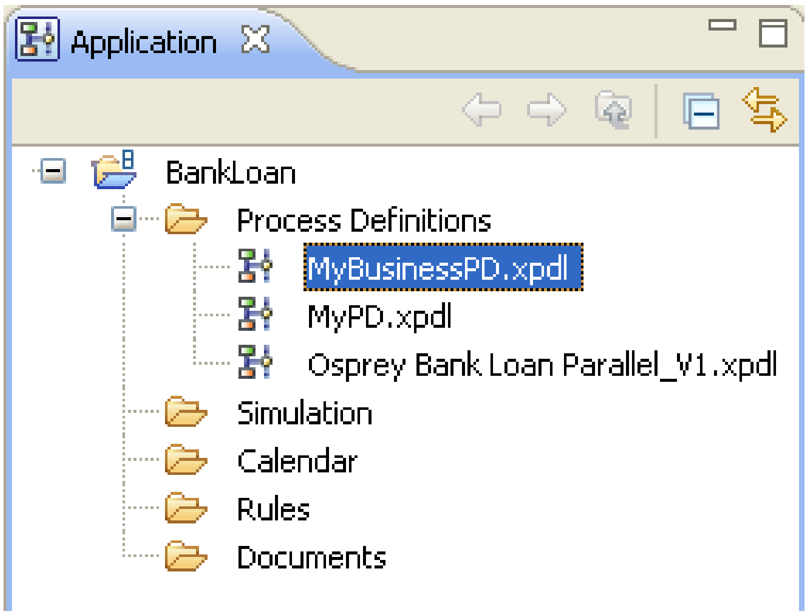


Figure 38: Application View

Note: Business User Perspective does not support Server Projects.

3.3.2 Process Outline Editor

The **Process Outline** editor contains the list of tasks, which are Activities defined in the Process Definition. The **Process Outline** editor displays the following:

General Information

- **Name**: Name of the Process Definition

- **Description**: Description of the Process Definition

Tasks Table

The **Tasks Table** has the following columns:

Column Name	Description
Task Name	Name of the task or subtask
Parallel Task	Flag to identify a parallel task
Assignee	Assignee of the task
Due Date	Date when the task is due to be completed
Description	Additional information about the task
Duration	Duration for the completion of the task for an assignee
Timeline	Gantt Chart based on the duration is displayed in this column. This is a view-only column.

The **Tasks Table** has the following function buttons:

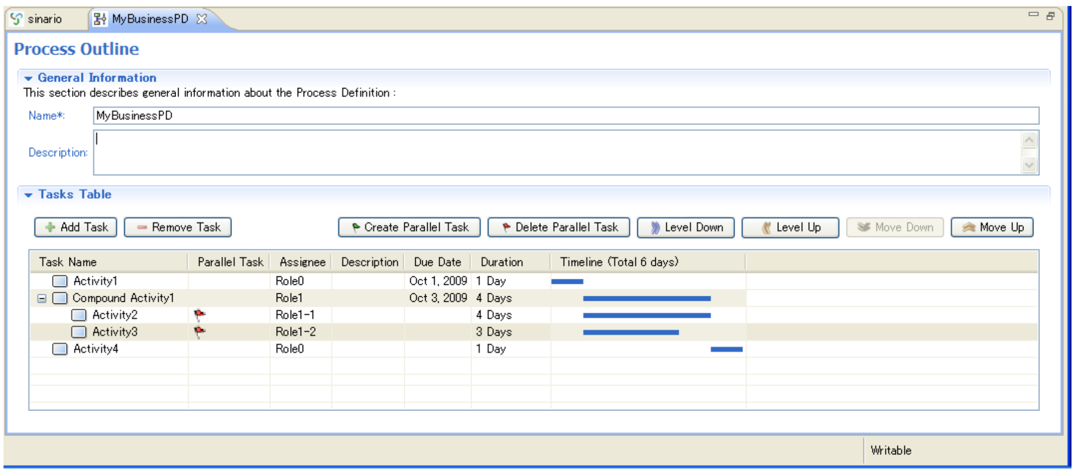
Function Name	Description
Add Task	To add tasks to the **Tasks Table**
Remove Task	To remove the selected task from the **Tasks Table**
Create Parallel Task	To create parallel task in the **Tasks Table**
Delete Parallel Task	To delete selected parallel tasks from the **Tasks Table**
Level Down	To make the selected task a sub-task
Level Up	To make the selected sub-task a task
Move Down	To move the selected task down
Move Up	To move the selected task up

Note: When you click in the **Due Date** field, a calendar icon is displayed. You can select the due date using the calendar. The date you select using the calendar is absolute date.

To set a relative due date for a task, you need to again click in the **Due Date** field (after the calendar icon is displayed) without selecting a date in the calendar (you do not need to click the calendar icon). Then you can enter a relative date which is entered as a number of days after a certain task is completed in the editable **Due Date** field.

Note: You can define a parallel flag at up to five places.

The following figure shows the **Process Outline** editor:



This section describes general information about the Process Definition :

Name*: MyBusinessPD

Description:

Task Name	Parallel Task	Assignee	Description	Due Date	Duration	Timeline (Total 6 days)
Activity1		Role0		Oct 1, 2009	1 Day	
Compound Activity1		Role1		Oct 3, 2009	4 Days	
Activity2		Role1-1			4 Days	
Activity3		Role1-2			3 Days	
Activity4		Role0			1 Day	

Figure 39: Process Outline Editor

If you right-click the task row in **Task Name** column, the following options are displayed:

- **Add Task**: Adds a task
- **Add Sub-task**: Adds a sub-task to the selected task
- **Remove Task**: Removes the selected task
- **Level Up**: Makes a selected sub-task a task
- **Level Down**: Makes a selected task a sub-task
- **Move Up**: Moves up the selected task
- **Move Down**: Moves down the selected task
- **Columns**: Selects or hides columns per your requirements

If you right-click the task row in **Parallel Task** column, the options are displayed that can be used to group parallel tasks. For example, tasks A and B can be grouped as parallel tasks using **Task Group 1** option.

Note: To create parallel tasks, the color of the group flag that you select for the parallel tasks, needs to be the same and the tasks need to be at the same level and in a continuous sequence.

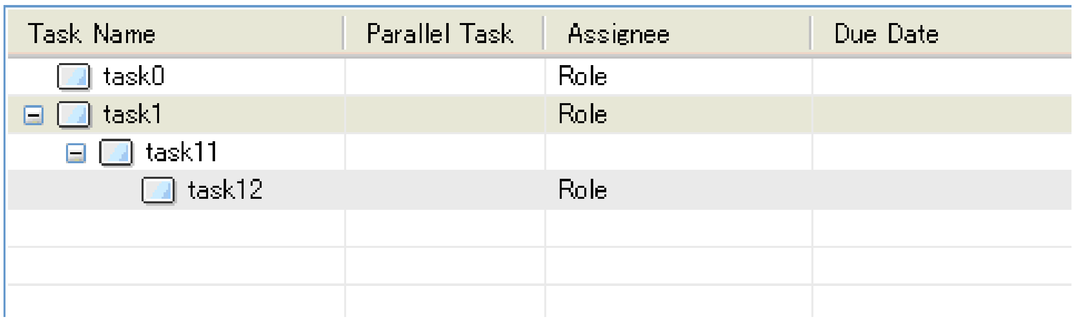
Relation between Tasks and Nodes

The type of node can be related with tasks in the following ways:

- Through task level
- Through color of parallel task flag

Through Task Level

Consider tasks that have been defined as in the figure below:



Task Name	Parallel Task	Assignee	Due Date
task0		Role	
⊟ task1		Role	
⊟ task11			
task12		Role	

Figure 40: Task Level

This process definition is displayed in the process editor as follows:

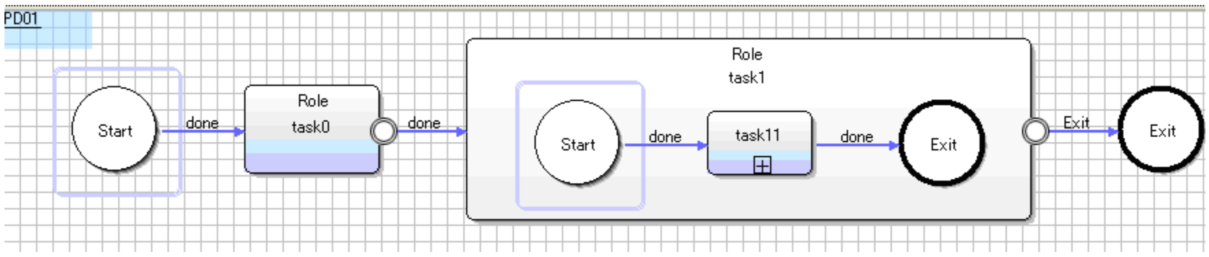


Figure 41: Related Process Definition for Task Level

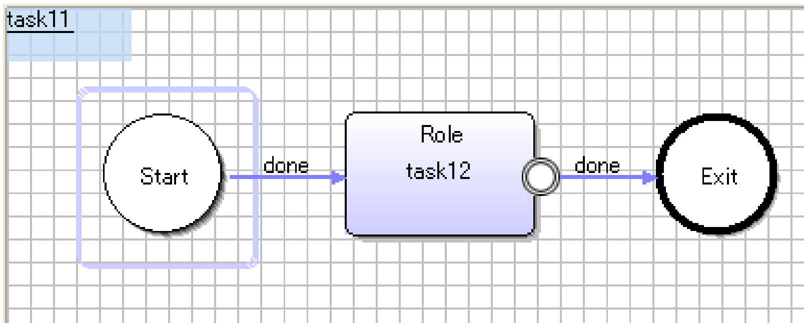


Figure 42: Child Task

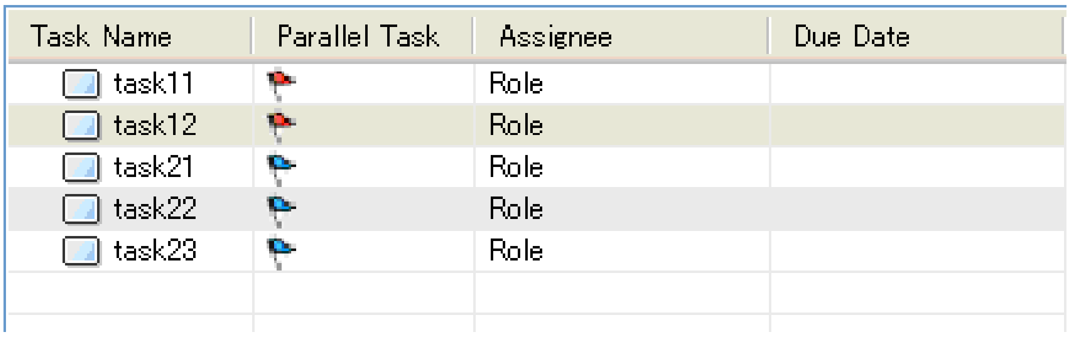
- **1st Level Tasks** (In this case, `task0` and `task1`):
  - If a task does not have a child task, the node is defined as an Activity.
  - If a task has a child task, the node is defined as a Compound Activity.
- **2nd Level Tasks** (In this case, `task11`):
  - This task is included in a Compound Activity
  - If this task does not have a child task, the node is defined as an Activity.
  - If this task has a child task, the node is defined as a Subprocess Node.

- A Subprocess definition of the same name as this Subprocess Node is created.

- **3rd Level Tasks** (In this case, `task12`):
  - This task is included in a Subprocess Definition.
  - If this task does not have a child task, the node is defined as an Activity.
  - If this task has a child task, the node is defined as a Subprocess Node.
  - A Subprocess definition of the same name as this Subprocess Node is created.

Through Color of Parallel Task Flag

Consider tasks defined as follows:



Task Name	Parallel Task	Assignee	Due Date
task11	🟥	Role	
task12	🟥	Role	
task21	🟦	Role	
task22	🟦	Role	
task23	🟦	Role	

Figure 43: Parallel Tasks

This process definition is displayed in the process definition editor as follows:

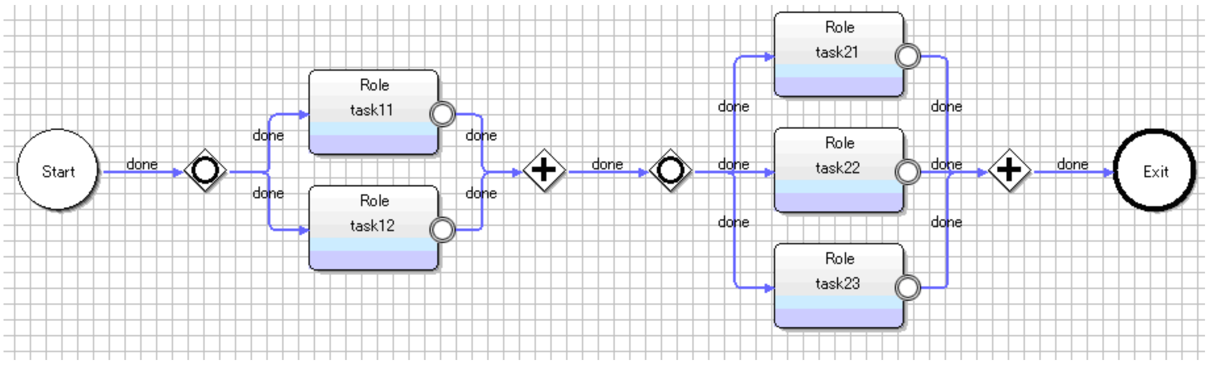


Figure 44: Process Definition for Parallel Tasks

Tasks with the same color of Parallel Task flag are defined to be processed parallely.

Notes on Business User Perspective

Note: When you define a new process definition, and you save this process definition with the subprocess node, the subprocess definition file is created at the same time with the following file name:

`<ProcessDefinitionName>_<SubprocessNodeName>.xpdl`

Note: When you open a process definition, change the task name that corresponds to the subprocess node and you save this process definition, it is saved in the same subprocess definition file. The file of the subprocess definition of a new name is not created.

Note: Ensure you do not open the subprocess definition file; instead, open the process definition.

Note: Ensure you do not change the subprocess definition file name when you are editing a process definition.

Note: If you save a process definition with parallel tasks, close the editor, and re-open the process definition, the color of the Parallel Task flag may be different.

3.4 Setting Your Preferences

You can set your preferences for the Process Definition editor and for the display of process definition elements. 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. You can change some settings for currently open Process Definition editors using the **View** menu and the pop-up menu of the Palette.

You can set your preferences for the Navigator view, too: You can choose whether the names or file names of local process definitions or scenarios are displayed. In addition, you can define which editor is to be used for opening a process definition, simulation file, or resource file.

You can also set preferences for messages that appear when starting and exiting Interstage BPM Studio.

To set your preferences:

1. Select **Window** > **Preferences**.
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) Make sure 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) Make sure 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) Make sure 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) Make sure 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) Make sure that the **Editor** page is open.

   b) Choose whether to turn feedback lines on or off (**Snap to Geometry**).

   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 your preferences for the editor that you want to use for opening a file in the Navigator view:

    a) Expand **Editors** and click **File Associations**.

    The **File Associations** page displays all available file types.

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

12. To set your preferences for messages that appear when starting and exiting Interstage BPM Studio:

    a) Click **Startup and Shutdown**.

    b) Change the settings as required.

13. To set your preferences for the server connections that are used by Interstage BPM Studio:

    a) Click **Server connection settings**.

    b) In the **Server connections** field, select a server, and click **Apply**. If you want to edit an existing server or add a new server to the list:

    c) Click **New** or **Edit**.

d) Type in the configuration parameters for the new or updated server connection.

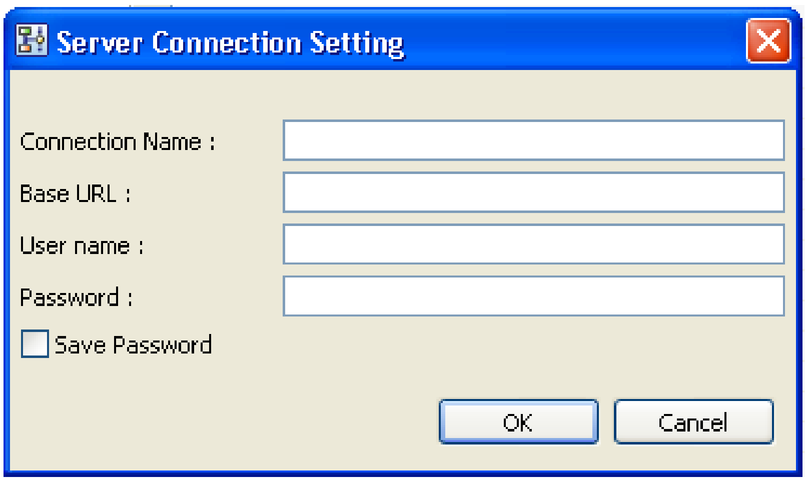


Figure 45: Server Connection Setting

Type in the following parameters:

- **Connection Name**: Name of the specified server connection
- **Base URL**: The Base URL is the URL used to create a connection to the server. The sample URL is:

```
http://<hostname>:<port>/<content>/_wfxml/<tenant>/
```

In the following URL, `console` is the context root for server connection, `ibpmserver` is the `hostname` and `49950` is the `port`.

```
http://ibpmserver:49950/console/_wfxml/default/
```

Note: Base URL is the URL that connects to the tenant if you are working in SaaS mode.

Note: In SaaS mode, the `<tenant>` in the URL connects you to the specific tenant on the Interstage BPM Server. Refer *Interstage BPM Studio Features* on page 17 to know more about SaaS mode.

Note: If you want to connect to Interstage BPM Server in non-SaaS mode, enter **default** in `<tenant>` in the URL. This will, through the default tenant, directly connect you to the server specified in `<hostname>` in the URL.

- **User name**: User name to authenticate with Interstage BPM
- **Password**: Password associated with the user name

e) 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 Interstage BPM Server.

f) Click **OK**.

14. To set your preferences for the **Process Outnile Editor**:

a) Click **Process Outline** editor.

b) In the Hide/Show Colums, set the displayed column and the column's order when you open Process Outline Editor.

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

4 Managing Projects

Projects are containers that help you organize your process definitions, resource files, forms, simulation scenarios, attachments, XML Schema 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.

Interstage BPM differentiates the following types of project:

- **Workflow Application project**: Interstage BPM 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. Later you can deploy such an application on an Interstage BPM Server that can be accessed by clients. 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.

  By default, Interstage BPM Studio shows a project named "BankLoan" in the Navigator view after installation. This is a Workflow Application project consisting of a process definition, several JSP pages, xml configuration files, etc. Take a look at the project structure to get a first impression on where to store which application artefacts.

  If you are upgrading from a previous version of Interstage BPM Studio and are using your previous workspace, Interstage BPM Studio shows a project named "Local" in the Navigator view. You can use existing process definitions and copy them to new Workflow Application projects or transfer them individually to the Interstage BPM Server.

- **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 an Interstage BPM server. By default, Interstage BPM 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 *Simulating Processes* on page 376 for details.
- **Server project**: Project displaying the process definitions stored on an Interstage BPM server. Before you can work with these process definitions, you need to log in to the Interstage BPM server as an Interstage BPM user. You can transfer any local process definition to the server or export one from the server to your local machine.

  In SaaS mode, you can connect to a tenant on an Interstage BPM Server. Each tenant has its own applications. Refer *Interstage BPM Studio Features* on page 17 and *Online Editing: Server Projects* on page 31 to know more about SaaS and non-SaaS modes.

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

4.1 Managing Workflow Application Projects

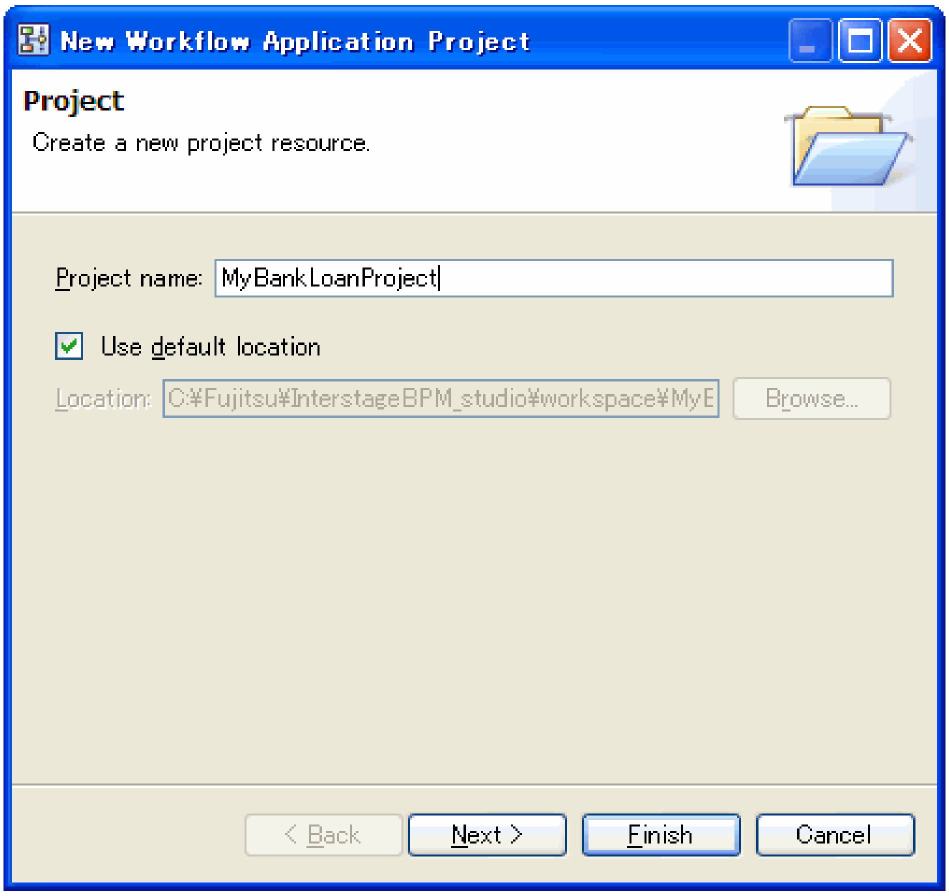
This section explains the functions available for managing Workflow Application projects.

4.1.1 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.

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


New Workflow Application Project
Project
Create a new project resource.
< Back
Next >
Finish
Cancel
Browse...

Project name: MyBankLoanProject

✓ Use default location

Location: C:¥Fujitsu¥InterstageBPM_studio¥workspace¥MyE

Figure 46: Creating a Workflow Application Project

3. 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**.
4. Click **Next**.

   Type in the required information:
   - **Description**: Provide a brief description of your project.
   - **Owner Group**: Specify the group to which the user who manages (begin/stop/update/uninstall) the application, belongs.
   - **Email Customized Class**: Specify the class name in order to customize the emails which notify the start of a workitem. For details, refer *Setting Email Customized Class* on page 114.
5. Click **Finish**.

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

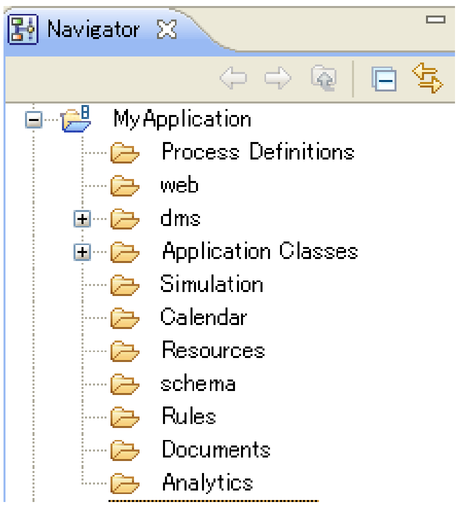


Figure 47: Creating a new Workflow Application project

4.1.2 Components of a Workflow Application Project

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

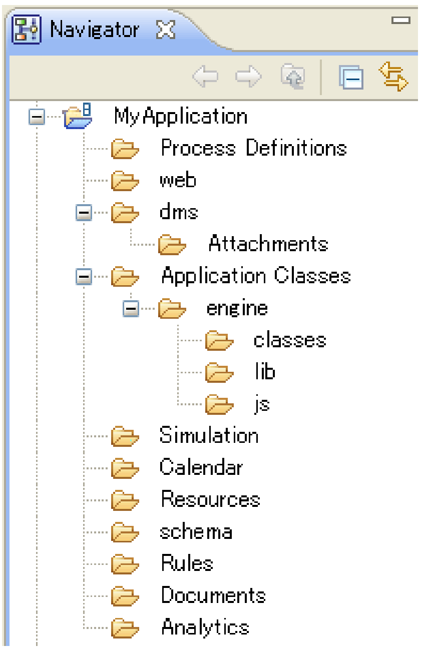


Figure 48: Workflow Application Project Folder Structure

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 material that is to be made available to Interstage BPM clients through the web. The files placed in this folder are exposed to the Internet. 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.
  - In **Attachments** folder, Blaze files too are stored. Refer *Integrating Blaze Rules* on page 315 to know more about Blaze files.
  - New ILOG rules files will be placed in this folder, too. Refer *Integrating ILOG JRules* on page 318 to know more about ILOG rules.

  The **dms** folder will, together with any subfolders, be copied to the corresponding folder on the Interstage BPM 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 *Executing a Simulation Scenario* on page 390.
- **Calendar**: This is where you store the calendar `(.cal)` files of your application. For more information on business calendars, refer to section *Creating Your Own Business Calendars* on page 178.
- **Resources**: This is where you store resource files as listed below. For more information on these resource files, refer to section *Working With Resources* on page 94. The **Resources** folder contains and supports creating:
  - Configuration files for Java Action Agents (refer *Defining Java Agents* on page 362)
  - Configuration files for new data sources used by Interstage BPM Studio (refer *Configuring Interstage BPM for Database Actions* on page 323)
  - FTP Agent files (refer *Defining FTP Agents* on page 363
  - HTTP Agent files (refer *Defining HTTP Agents* on page 367)
  - Custom Config files
  - File Listener files (refer *Defining File Listener* on page 370)
  - Process Scheduler File (refer *Interstage Business Process Manager Developer's Guide* for more information)
- **schema**: Saves XML Schema files for using Custom Data type UDA. For more information about XML Schema files, refer *UDAs of Type CUSTOM* on page 223.
- **Rules**: Rules folder contains **Rules Set** folder and **Rules Set** folder further contains Decision Tables. Refer *Creating a New Rule Set* on page 249 for more information about Rules Set.
- **Documents**: The Process Definition reports are stored in this folder. Refer *Using Process Definition Reports* on page 127 for more information about reports and **Documents** folder.

- **Analytics**: This is the convenient access point for the Analytics application. If Interstage Analytics is installed, this folder allows you to create and edit the Analytics settings.

Note: You have to install the Interstage Analytics plugin before you can use the Analytics function with Interstage BPM Studio.

MyApplication 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 *Project Components* on page 411.

Note: In Interstage BPM Console, if you want to set the Application Skin or the Application Logo of this application, you have to do the following operations in Interstage BPM Studio.

1. Create **styles** folder under the **web** folder of this application project.
2. Import style sheet files (*.css) to the **styles** folder.
3. Create **images** folder under the **web** folder of this application project.
4. Import image files (for using application logos) to this **images** folder.

For more information about application skin or application logo, refer the *Interstage Business Process Manager Console Online Help*.

4.1.3 Using Folders

In any Workflow Application project, you can create additional folders or structure the existing folders to your needs in order 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.

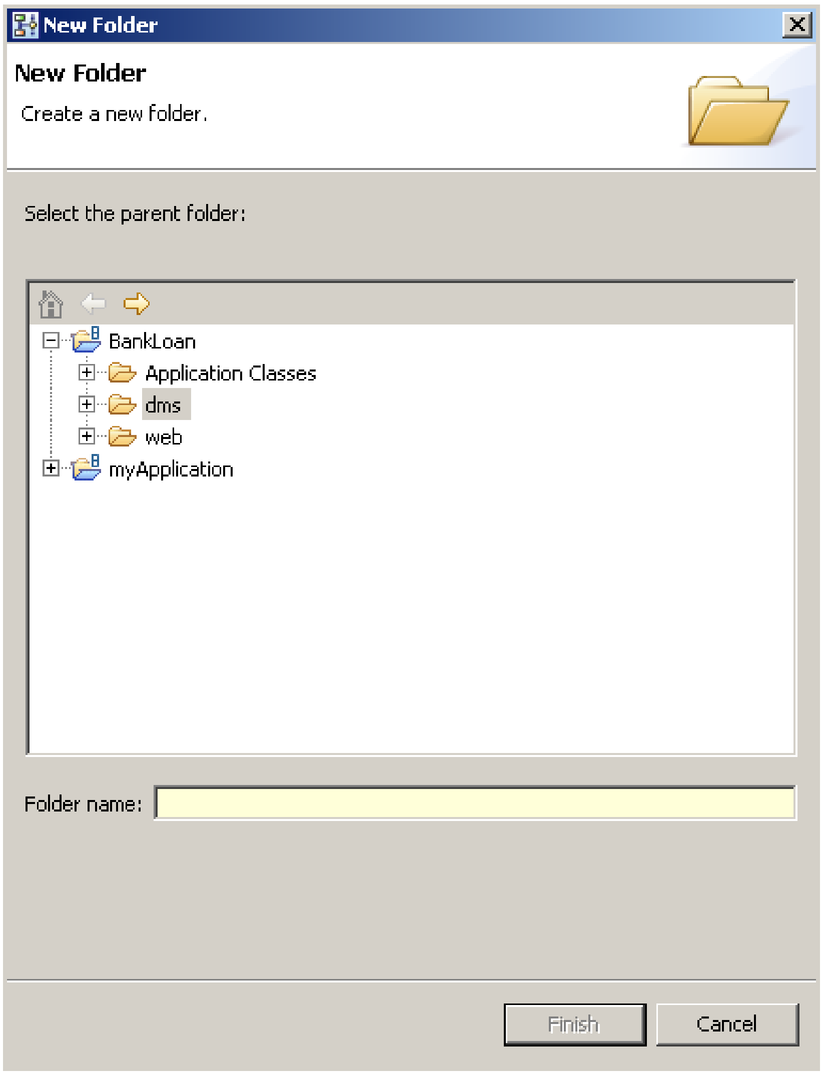


Figure 49: Creating a new folder

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 *Importing Folders and Files* on page 91 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.

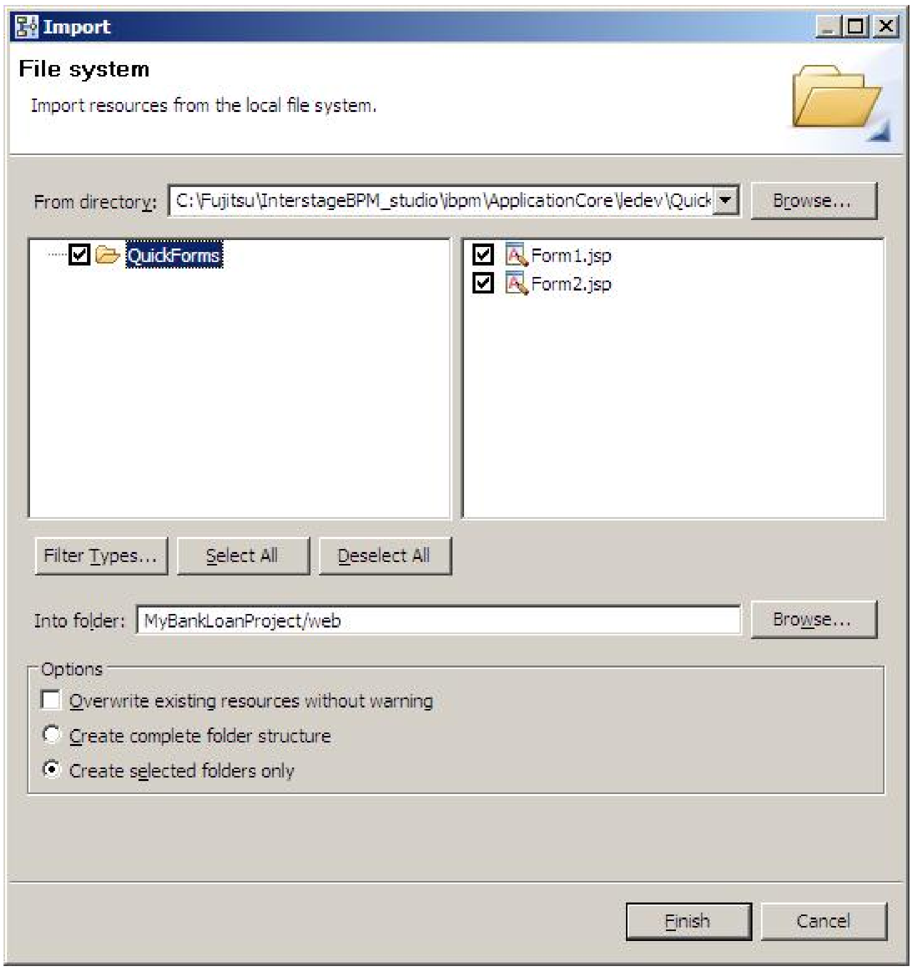
4.1.4 Importing Folders and Files

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.

To import folders or files from a file system:

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:



Import

File system

Import resources from the local file system.

From directory: C:\Fujitsu\InterstageBPM_studio\bpm\ApplicationCore\edev\Quick

[✓] QuickForms

[✓] Form1.jsp

[✓] Form2.jsp

Into folder: MyBankLoanProject/web

Options

[ ] Overwrite existing resources without warning

( ) Create complete folder structure

(●) Create selected folders only

Figure 50: Importing Folders and Files

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 you select the **Create complete folder structure** option, no overwrite warning message is displayed regardless of selection of the **Overwrite existing resources without warning** option.

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 *Importing Workflow Application Projects* on page 97.

4.1.5 Searching for Folders and Files

You have created a Workflow Application project where you can search for specific folders and files.

Interstage BPM 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.

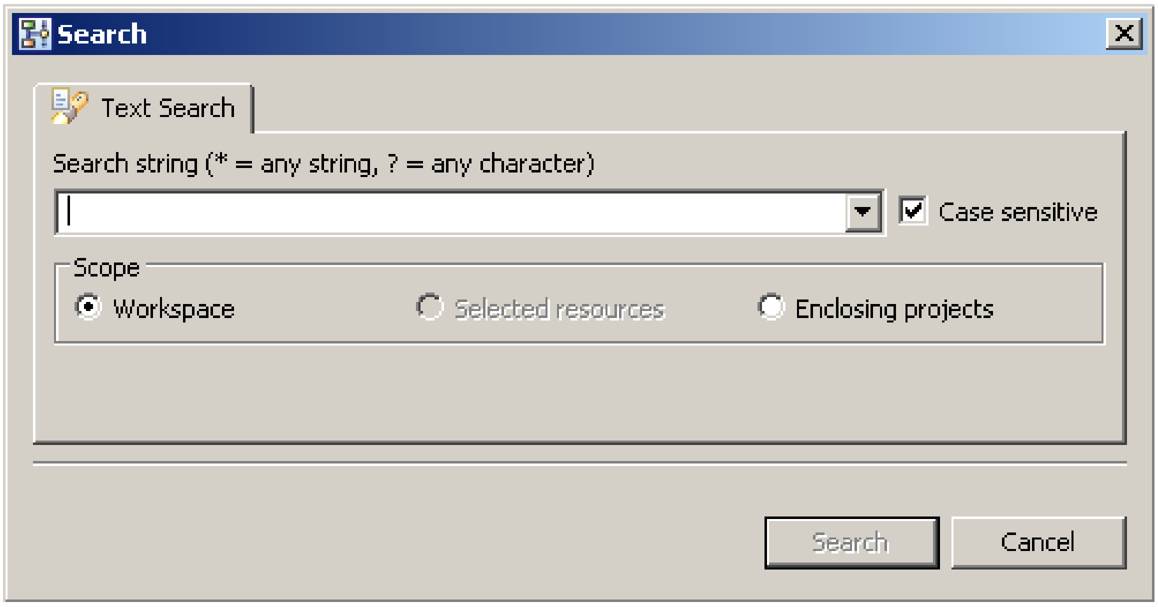


Figure 51: Searching for files

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

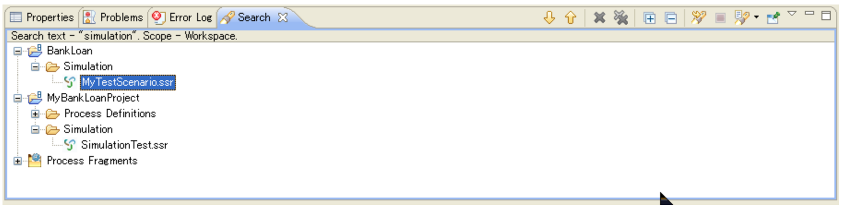
3. If you want Interstage BPM Studio to distinguish between uppercase and lowercase letters, select the **Case sensitive** check box.
4. Select one of the **Scope** check boxes to narrow the scope of search.

   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:



```
Search text - "simulation". Scope - Workspace.
BankLoan
    Simulation
        MyTestScenario.ssr
MyBankLoanProject
    Process Definitions
    Simulation
        SimulationTest.ssr
Process Fragments
```

Figure 52: Displaying Search Results

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.

4.1.6 Working With Resources

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

Interstage BPM Studio allows you to add new resources to Workflow Application projects. The term "Resources" refers to the following:

- FTP Agent files
- HTTP Agent files
- Custom Config files
- Configuration file for Java Agents used by the application (`agentsConfig.xml`)
- Configuration files for new data sources used by Interstage BPM Studio (`DataSourceDefinition.xml`)
- File Listener file (`fileListenerConf.xml`)
- Process Scheduler file (`ProcessScheduler.xml`)

After creating a new project, the project folder structure contains, among other folders, an empty **Resources** folder. This folder is intended for the resource files listed above.

Note: You can add only one Java Agent, one data source configuration file, one File Listener file and one Process Scheduler 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.
   - **Java Actions** > **Data Source**: Use this option to add a new data source.
   - **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
      - `agentsConfig.xml`
      - `DataSourceDefinition.xml`
      - `fileListenerConf.xml`
      - `ProcessScheduler.xml`

   b) Please refer *Interstage Business Process Manager Developer's Guide* to know about the tags of each of these files.

Note: For more information, refer to sections:
- *Defining FTP Agents* on page 363 (FTP Agents)
- *Defining HTTP Agents* on page 367 (HTTP Agents)
- *Defining Java Agents* on page 362 (Java Agents)
- *Configuring Interstage BPM for Database Actions* on page 323 (Data Source)
- *Defining File Listener* on page 370 (File Listener)
- *Defining Process Scheduler* in *Interstage Business Process Manager Developer's Guide*

   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.

4.1.7 Copying Workflow Application Projects

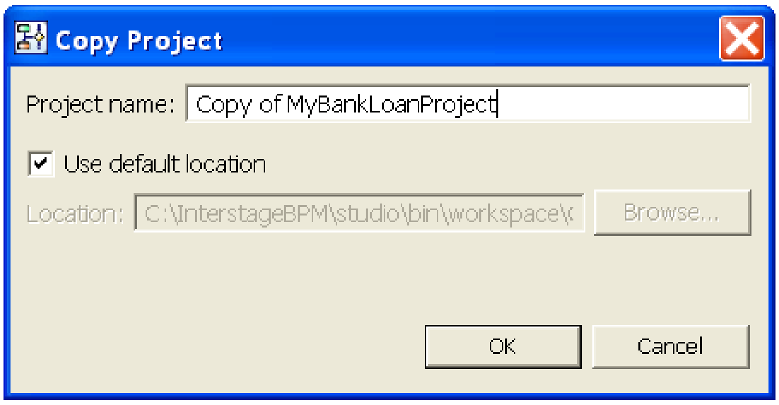
You can copy Workflow Application projects with all of their contents from one location to another.

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.



Project name: Copy of MyBankLoanProject

Location: C:\InterstageBPM\studio\bin\workspace\(

Figure 53: Copying a Project

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.

4.1.8 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.

4.1.9 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 *Managing Server Projects* on page 117.

4.1.10 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 *Managing Server Projects* on page 117.

To open a project:

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

4.1.11 Removing Workflow Application Projects

You can remove projects with all of their contents from Interstage BPM Studio.

Note: You cannot remove the projects named "Simulation Scenarios". 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**.

4.1.12 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.

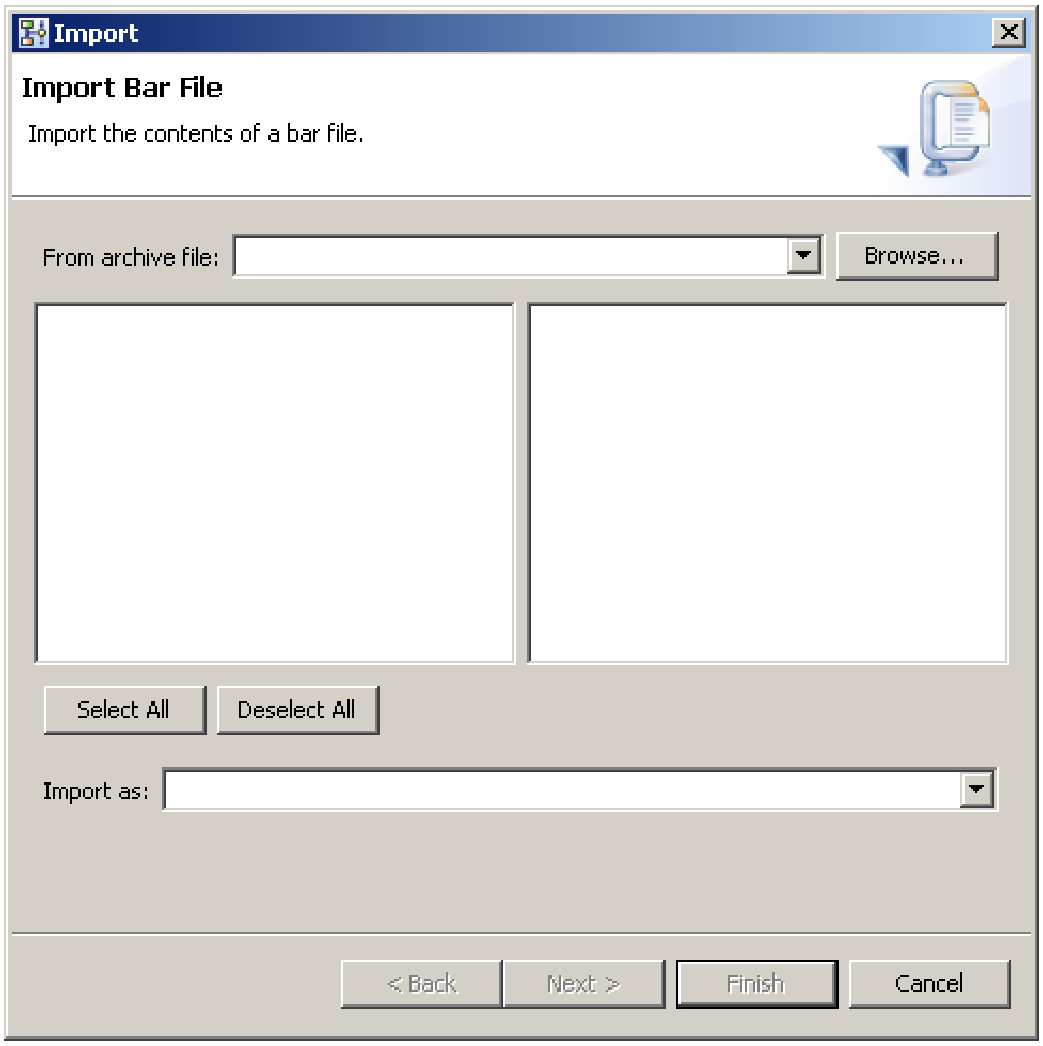
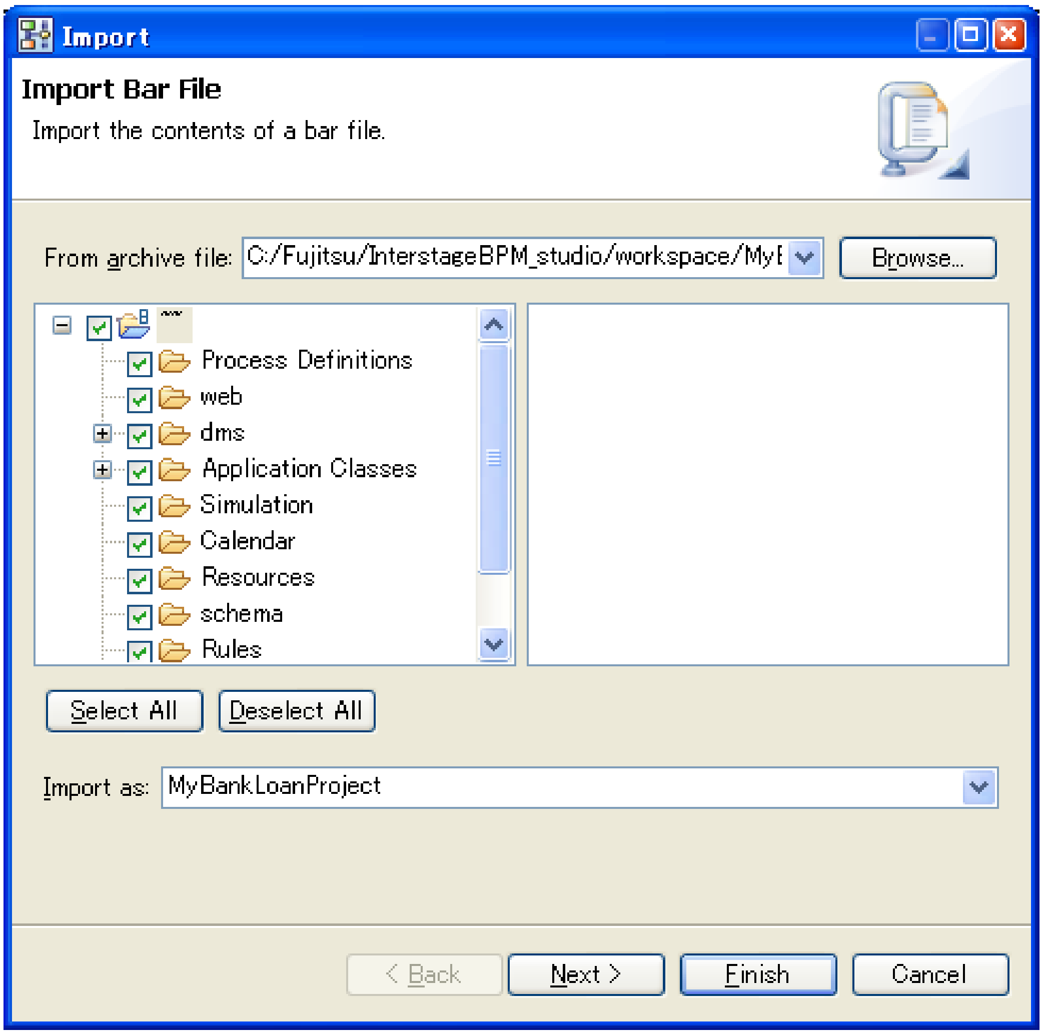


Figure 54: Displaying the Import Bar File dialog

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 file path is displayed, for example `C:\Fujitsu\InterstageBPM_studio\workspace\MyBankLoanProject.bar`.

      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:


Import

Import Bar File

Import the contents of a bar file.

From archive file: C:/Fujitsu/InterstageBPM_studio/workspace/MyI

Process Definitions
web
dms
Application Classes
Simulation
Calendar
Resources
schema
Rules

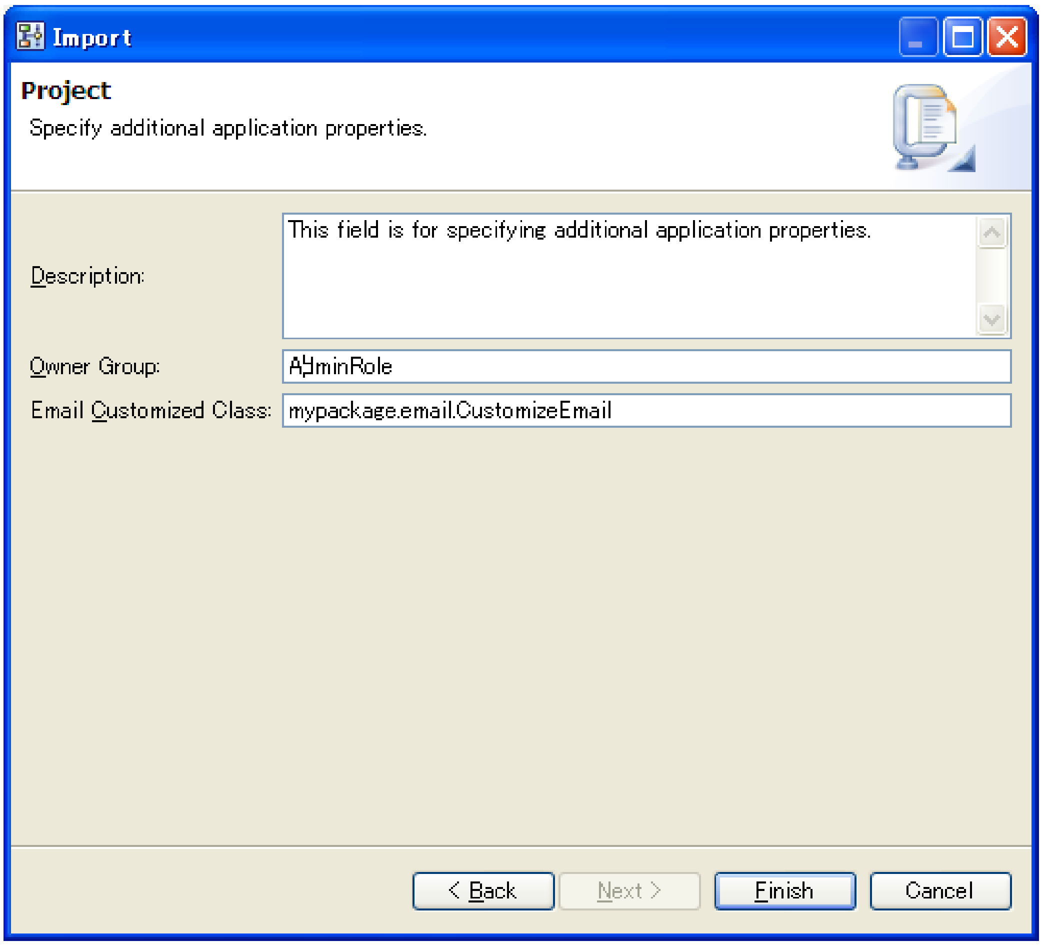
Import as: MyBankLoanProject

Figure 55: Specifying import settings

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.
   c) Provide the class name in the **Email Customized Class** field. You can customize the notification emails. Refer *Setting Email Customized Class* on page 114 for information about customizing email notifications.

   Providing additional project information is optional. You can provide the additional project information in the **Description** field. Also, provide the name of the owner group in the **Owner Group** field. This is the group to which the user who will own the particular project, belongs.

The following figure shows an example of specifying the **Description**, **Owner Group**, and **Email Customized Class**:



Project

Specify additional application properties.

Description:	This field is for specifying additional application properties.
Owner Group:	AdminRole
Email Customized Class:	mypackage.email.CustomizeEmail

Figure 56: Providing additional information

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.

4.1.13 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 later be deployed on an Interstage BPM Server using the deployment mechanisms provided by Interstage BPM.

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.

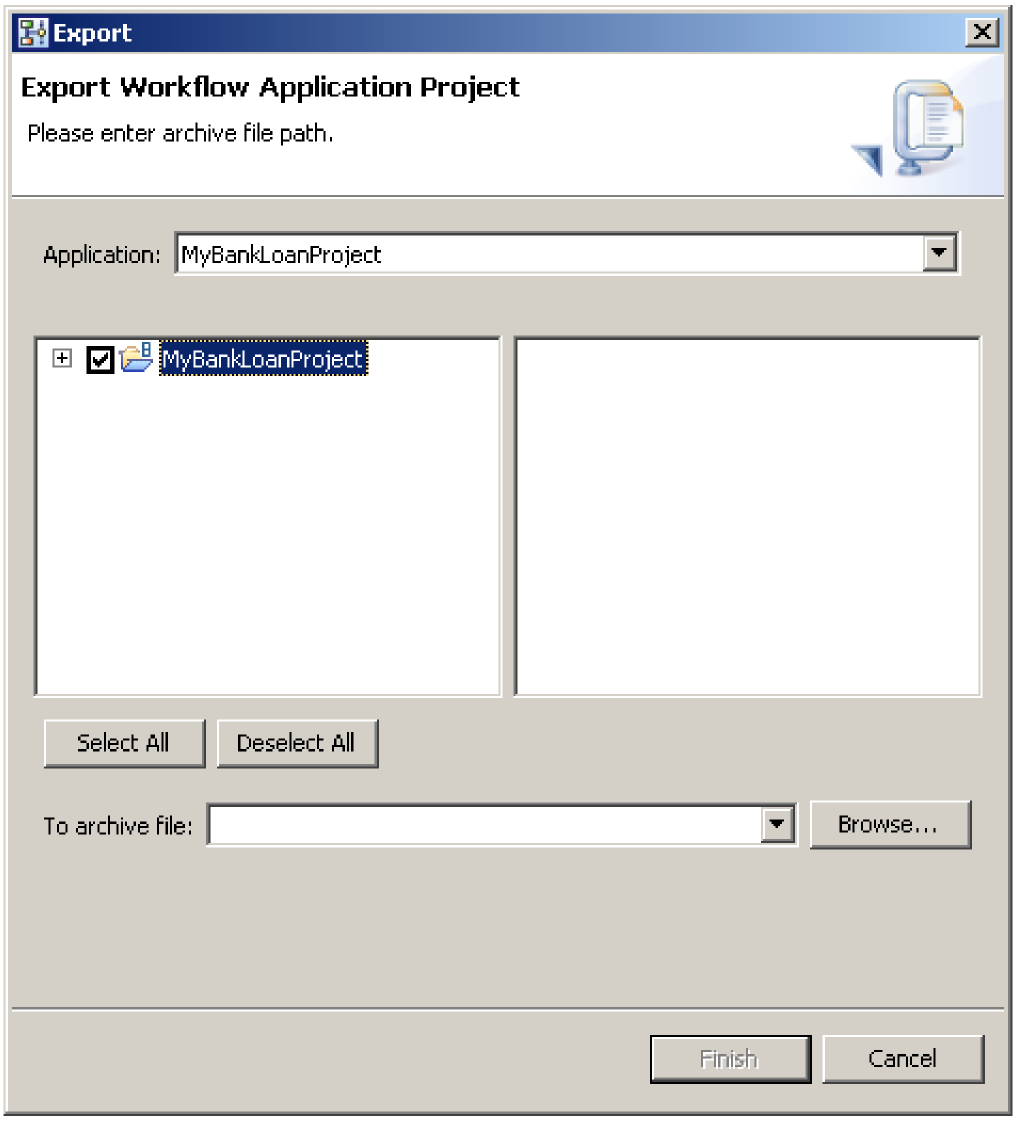

Export
Export Workflow Application Project
Please enter archive file path.
Application: MyBankLoanProject
MyBankLoanProject
Select All
Deselect All
To archive file:
Browse...
Finish
Cancel

Figure 57: Displaying the Export Workflow Application Project dialog

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\InterstageBPM_studio\workspace\Archive`).

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

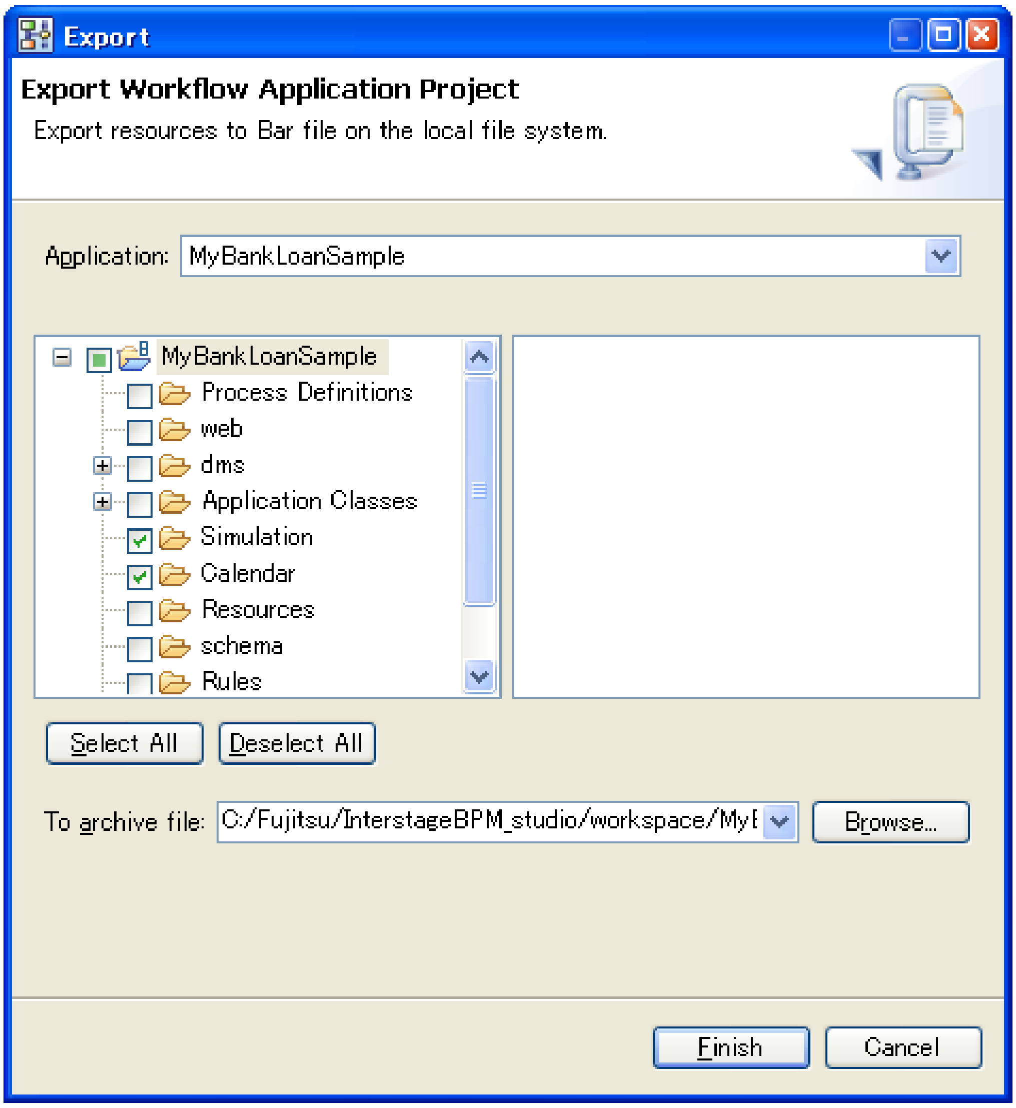

Export
Export Workflow Application Project
Export resources to Bar file on the local file system.
Application: MyBankLoanSample
MyBankLoanSample
Process Definitions
web
dms
Application Classes
Simulation
Calendar
Resources
schema
Rules
Select All
Deselect All
To archive file: C:/Fujitsu/InterstageBPM_studio/workspace/MyE
Browse...
Finish
Cancel

Figure 58: Specifying project components and archive file

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 `MyBankLoanProject.bar` file in the specified directory in the local file system.

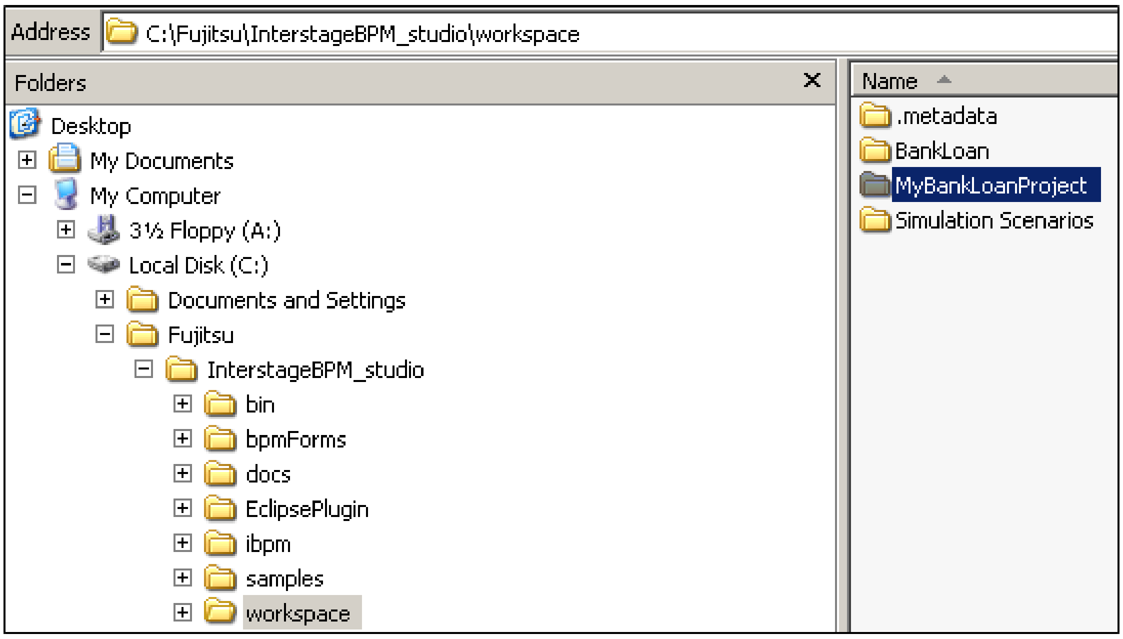


Figure 59: Location of the exported project

4.1.14 Downloading Workflow Application Projects From a Server

You can download Workflow Application projects packaged into `.bar` files from an Interstage BPM 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:

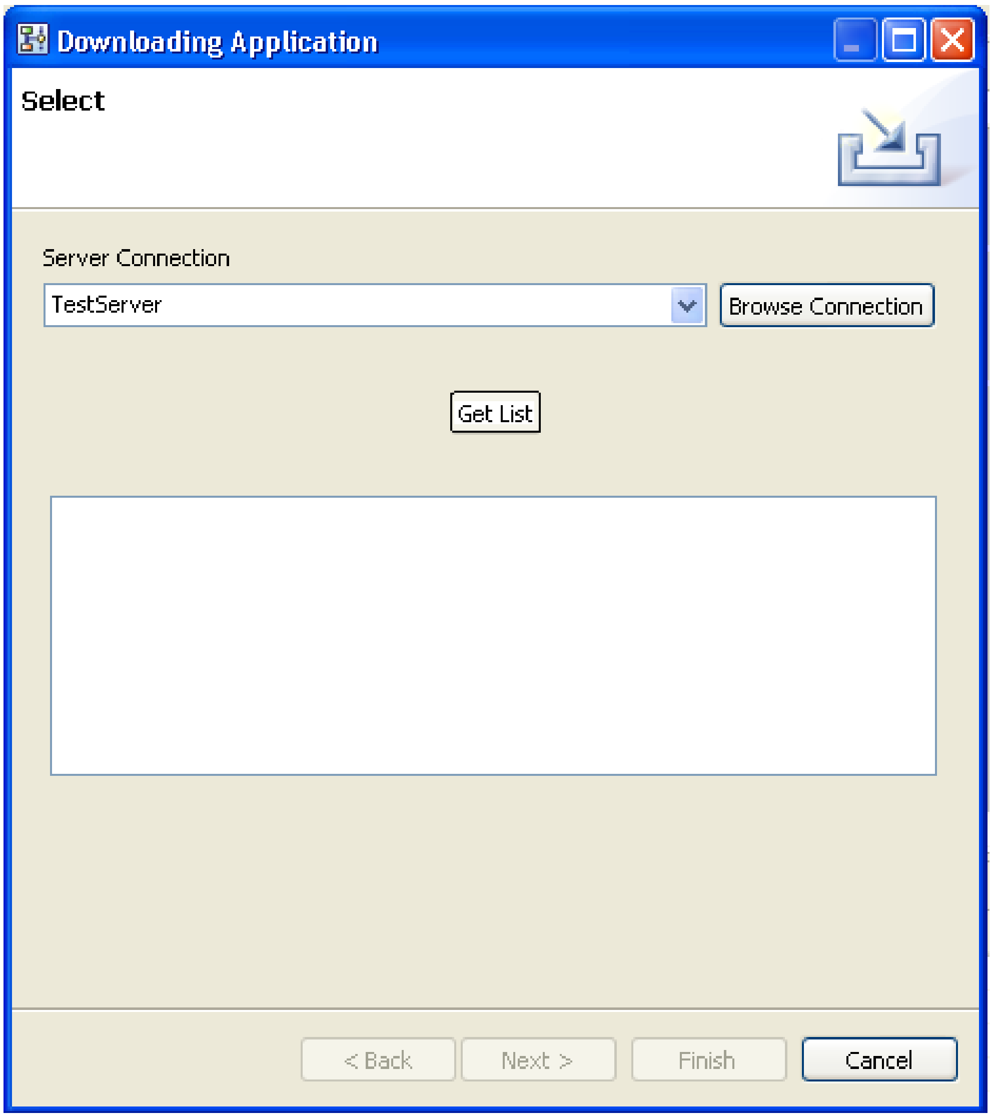


Figure 60: Displaying the Downloading application dialog

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.

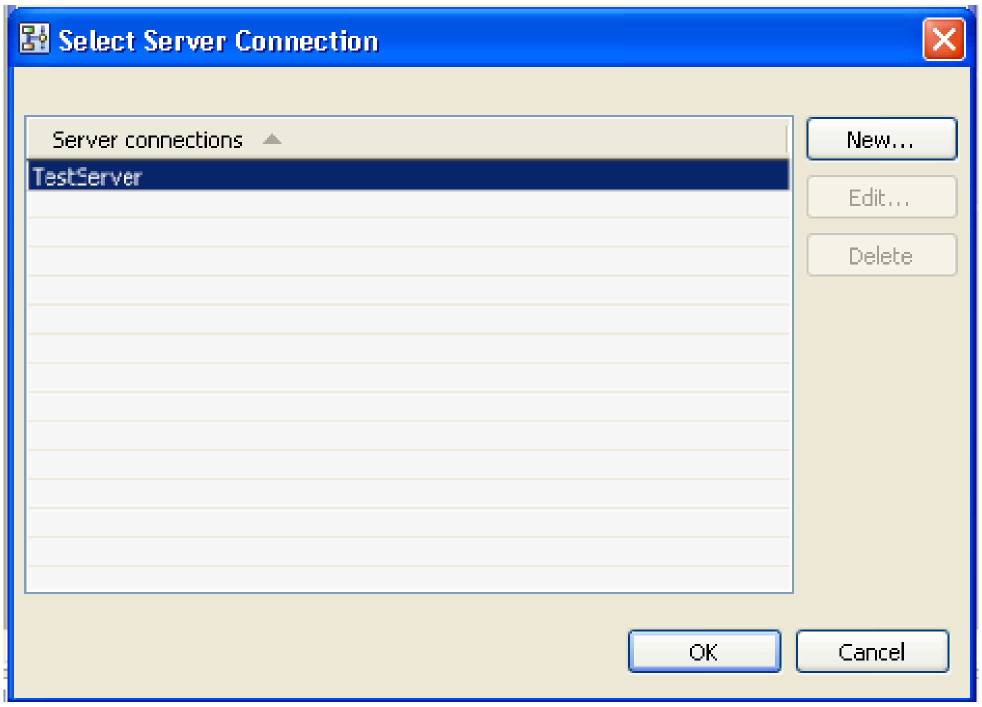


Figure 61: Displaying server connections

b) In the **Select Server Connection** dialog, click **New**.

   The **Server Connection Setting** dialog is displayed.

c) In this dialog:

   1. Specify the parameters for the new server connection. Refer to section *Setting Your Preferences* on page 81 for details on the required server parameters.

Note: In SaaS mode, the `<tenant>` in the URL connects you to the specific tenant on the Interstage BPM Server. Refer *Interstage BPM Studio Features* on page 17 to know more about SaaS mode.

Note: If you want to connect to Interstage BPM Server in non-SaaS mode, enter **default** in `<tenant>` in the URL. This will, through the default tenant, directly connect you to the server specified in `<hostname>` in the URL.

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

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

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

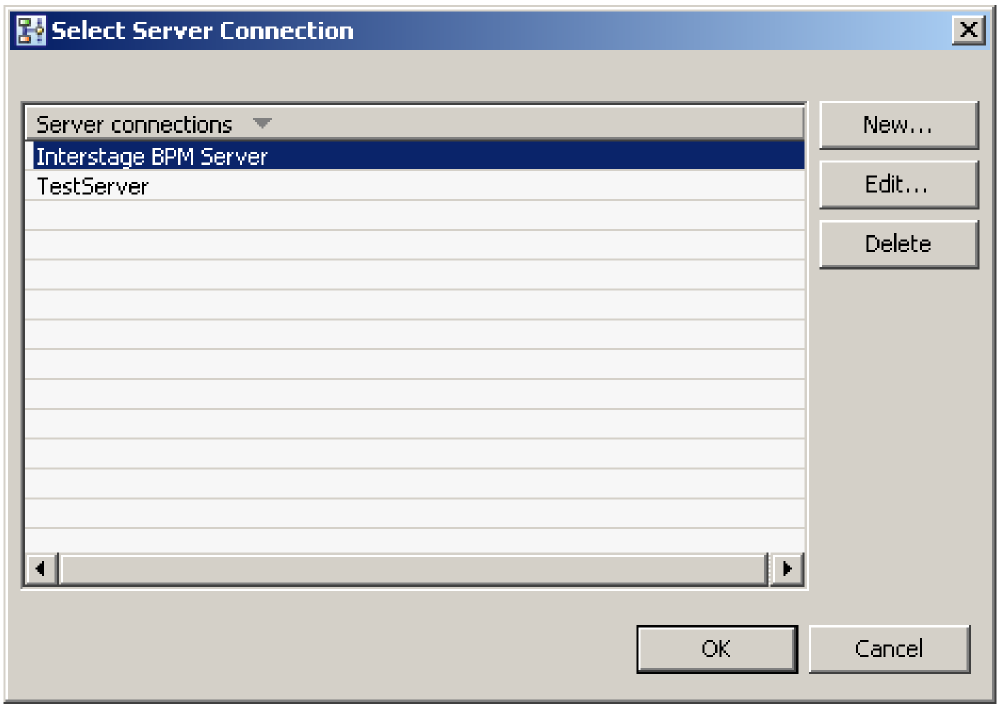


Figure 62: Displaying new server connection

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

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

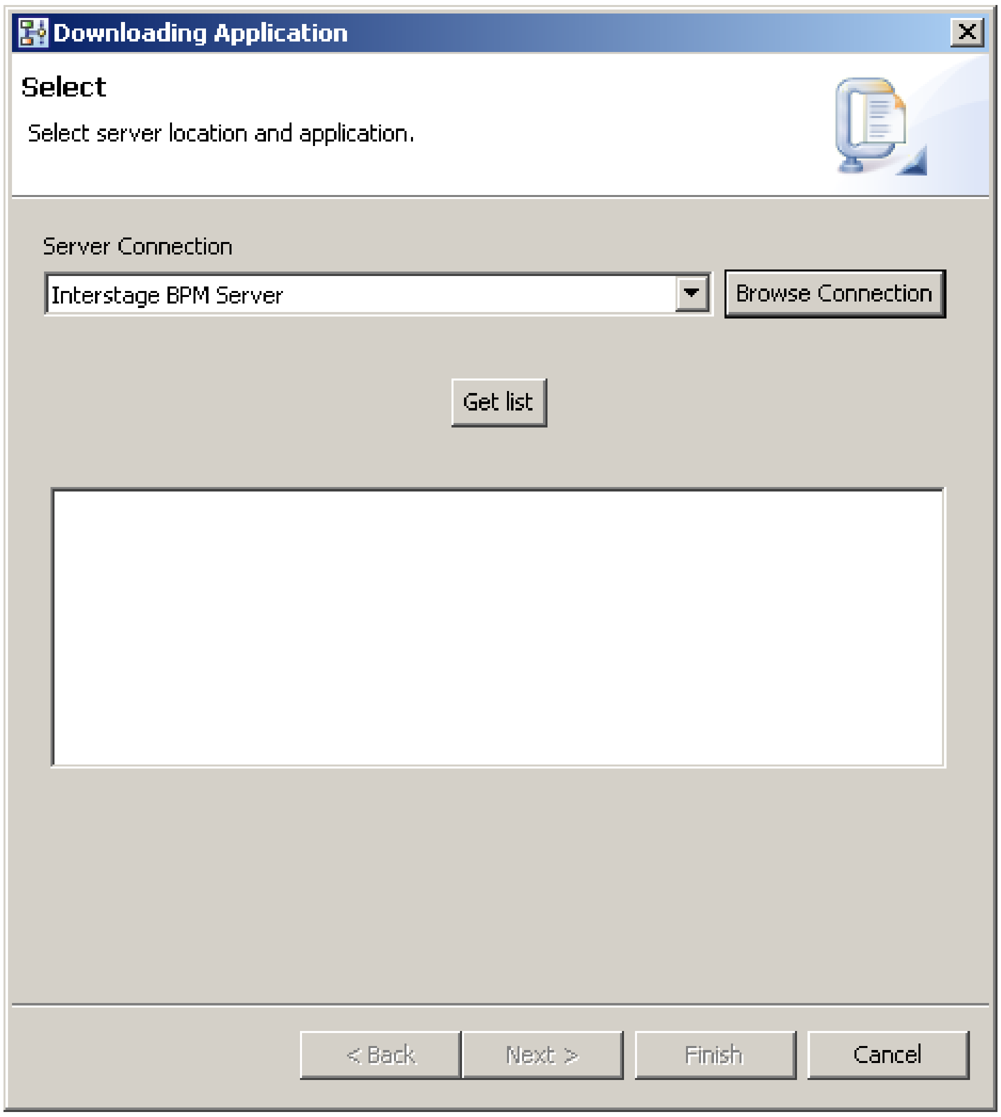


Figure 63: Displaying the new server connection

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.

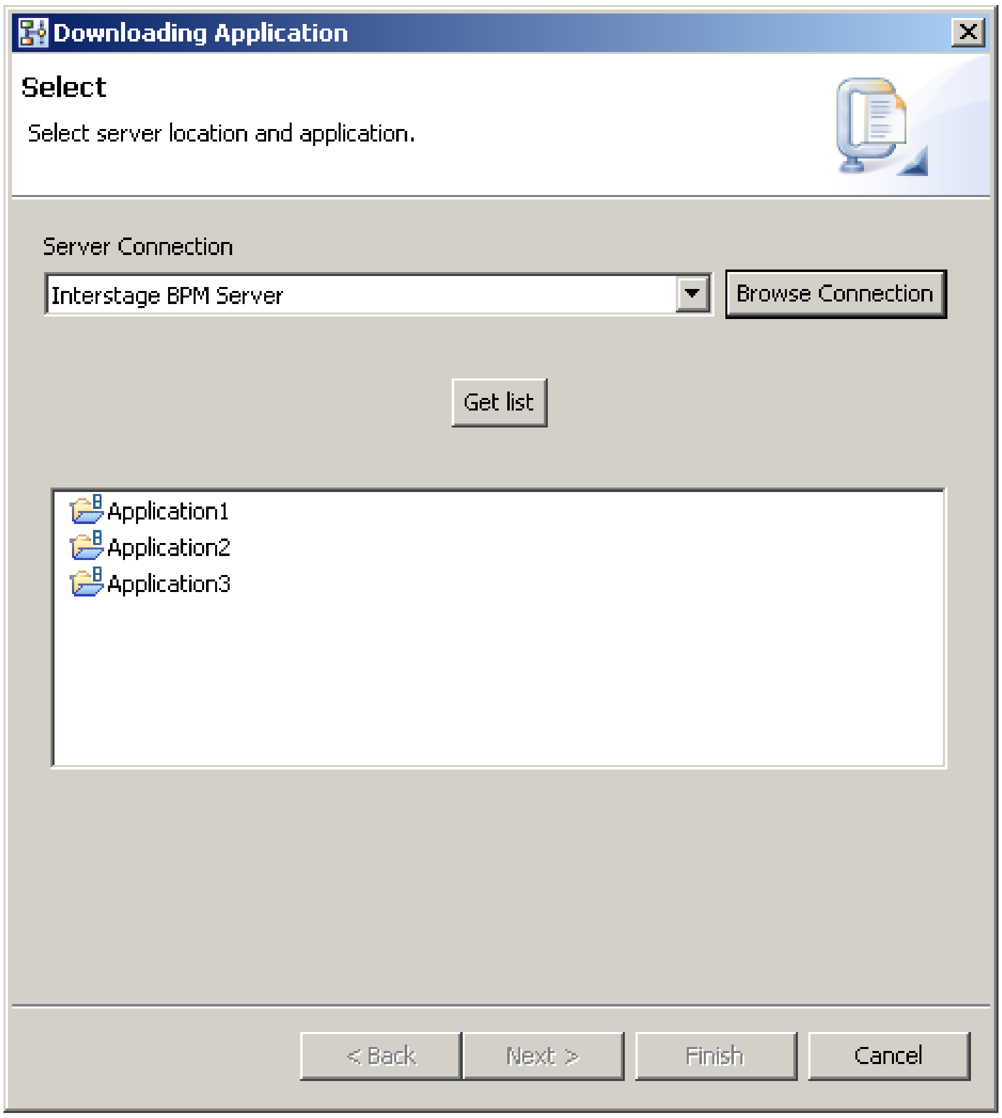


Figure 64: Displaying Workflow Application projects

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. Provide the additional project information in the **Description** field.

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

8. Provide the name of the owner group in the **Owner Group** field. This is the group to which the user who will own the particular project, belongs.
9. Provide the class name in the **Email Customized Class** field. You can customize the notification emails. Refer *Setting Email Customized Class* on page 114 for information about customizing email notifications.
10. 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: If the logged-in user, who is not an Interstage BPM Administrator (whether application owner or not), downloads an application from a server using the Interstage BPM Studio, Draft Process Definitions that are part of the application belonging to other users are not exported.

4.1.15 Uploading Workflow Application Projects to a Server

Pre-requisite:

An application space is created with the same name as that of the application in Interstage BPM Console before the application can be uploaded. This application is in offline state before the upload.

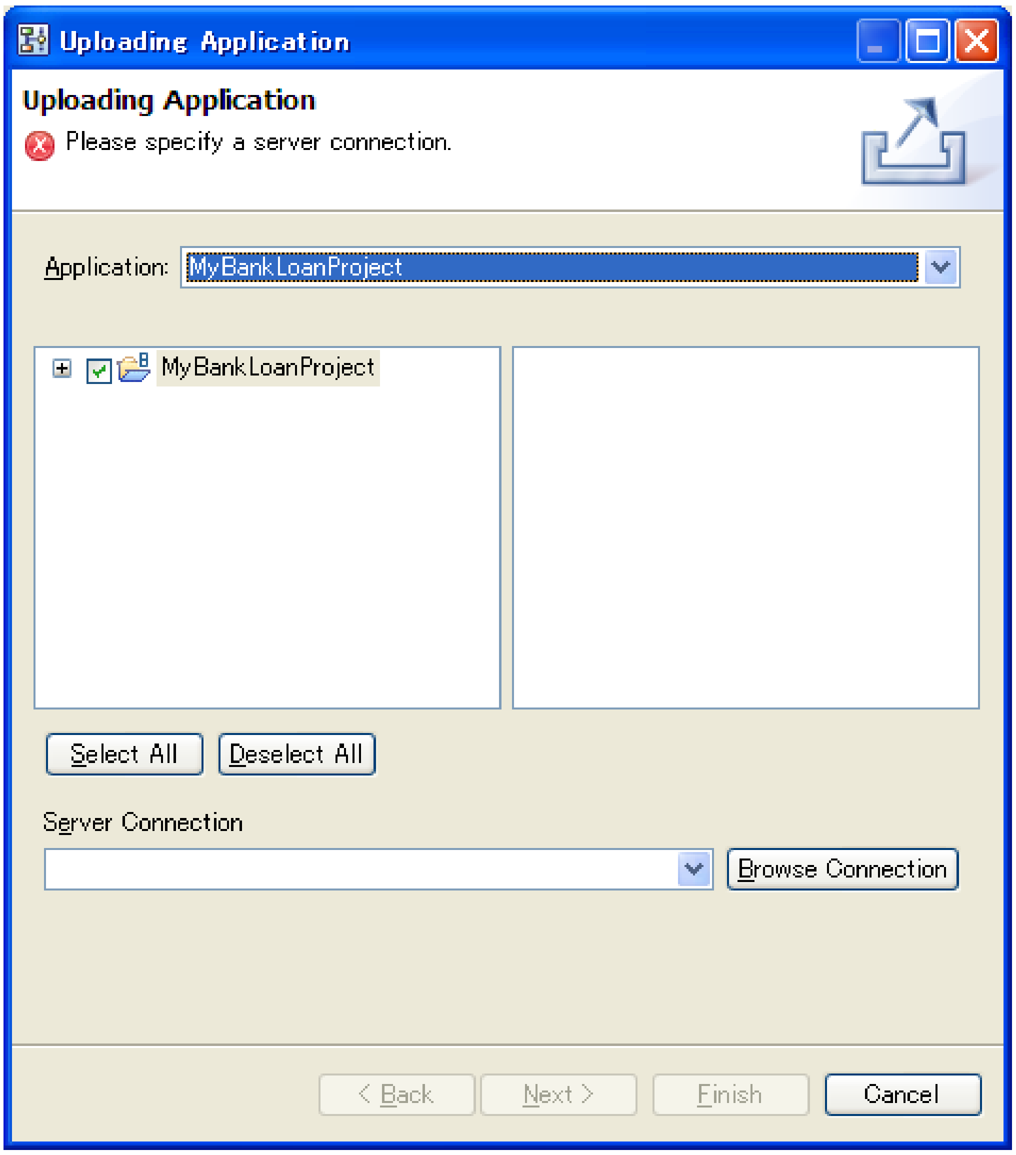
You can upload Workflow Application projects from Interstage BPM Studio to the Interstage BPM Server and package all files that make up your application into a `.bar` file. This `.bar` file can later be deployed on an Interstage BPM Server using the deployment mechanisms provided by the Interstage BPM Console. 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.


Uploading Application

Please specify a server connection.

Application: MyBankLoanProject

MyBankLoanProject

Figure 65: Displaying the Uploading application dialog

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 single components of a project. 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.

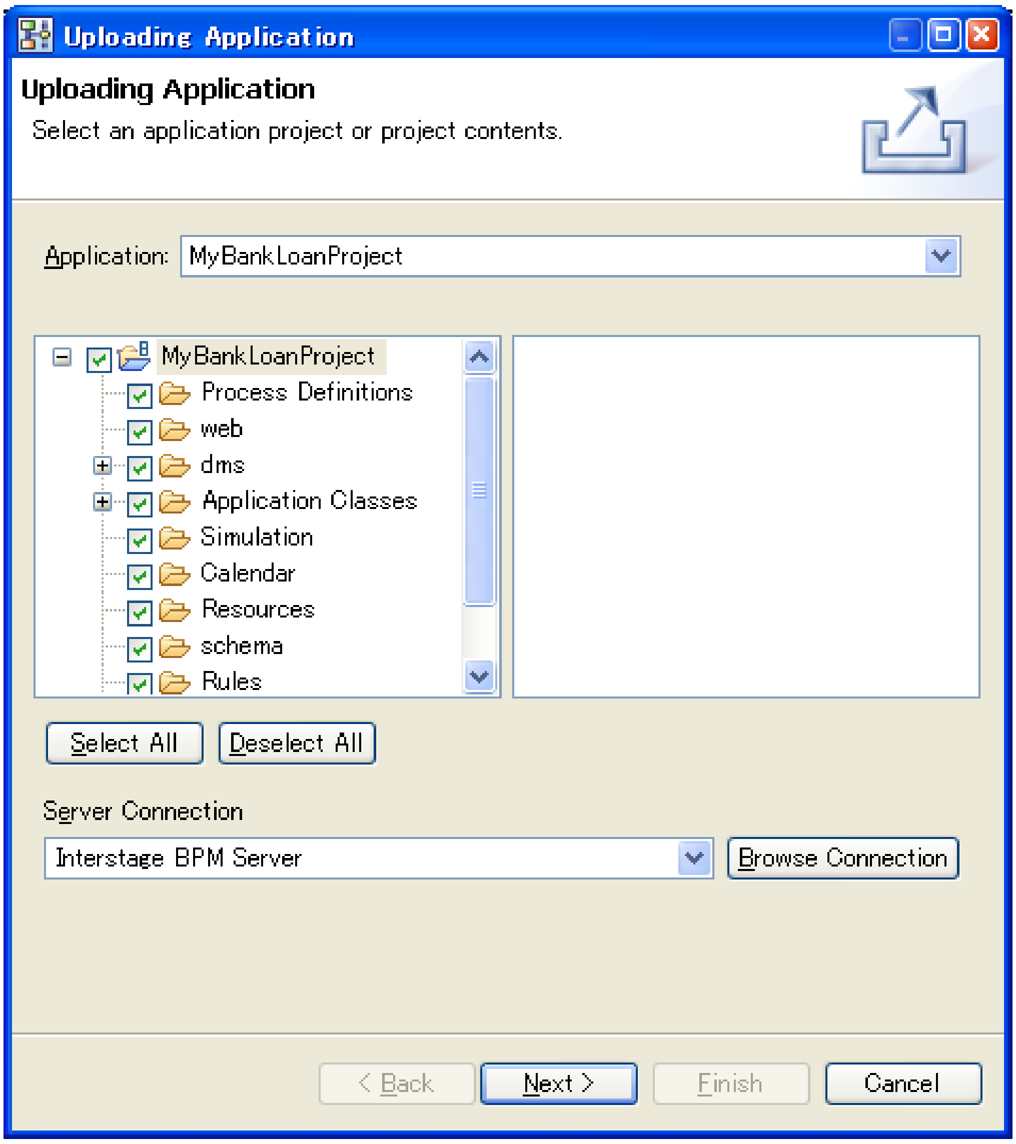
   You can select a server from a drop-down list or by browsing available server connections.

   - 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** tab, 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 *Setting Your Preferences* on page 81 to know more about various parameters.

Note: In SaaS mode, the `<tenant>` in the URL connects you to the specific tenant on the Interstage BPM Server. Refer *Interstage BPM Studio Features* on page 17 for more information about SaaS mode. Also, refer *Creating Server Projects* on page 117 for more information about the parameters.

Note: If you want to connect to Interstage BPM Server in non-SaaS mode, enter **default** in `<tenant>` in the URL. This will, through the default tenant, directly connect you to the server specified in `<hostname>` in the URL.

- In the example, you have specified an Interstage BPM Server. The **Uploading Application** dialog now looks as follows:



Uploading Application

Uploading Application

Select an application project or project contents.

Application: MyBankLoanProject

MyBankLoanProject
- Process Definitions
- web
- dms
- Application Classes
- Simulation
- Calendar
- Resources
- schema
- Rules

Select All | Deselect All

Server Connection

Interstage BPM Server | Browse Connection

< Back | Next > | Finish | Cancel

Figure 66: Selecting project components and 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:

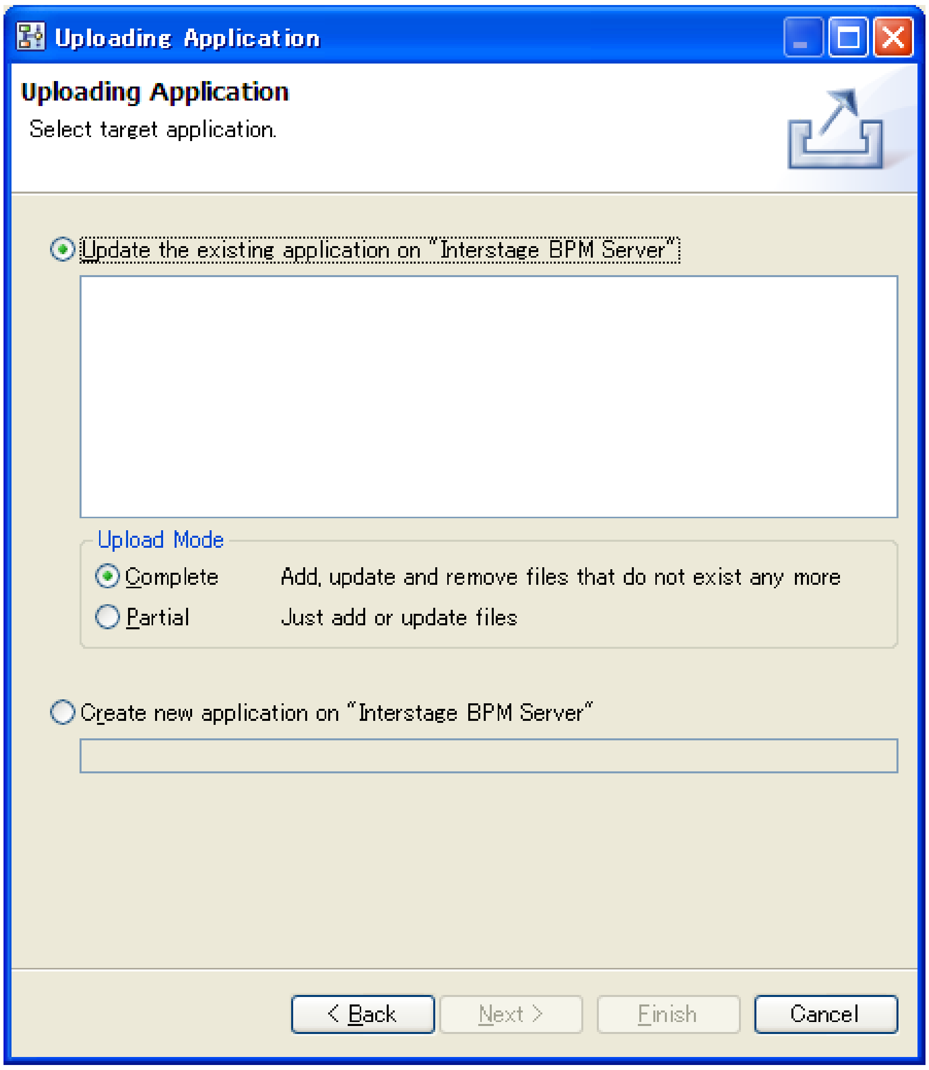

Uploading Application
Uploading Application
Select target application.
Update the existing application on "Interstage BPM Server"
Upload Mode
Complete
Add, update and remove files that do not exist any more
Partial
Just add or update files
Create new application on "Interstage BPM Server"
< Back
Next >
Finish
Cancel

Figure 67: Choosing to update or create an application

- Select the **Update the existing application** checkbox to upload your project to another project that already exists on the server. You can choose to completely or partially update an existing project. Select the **Complete** check box to completely replace the project on the server. Select the **Partial** check box to add or update selected project components.

Note: When you choose **Complete**, the uploaded process definitions are added as the latest version. This means that any existing process definitions are not removed, but remain as previous versions. All other resources are replaced.

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

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

4.1.16 Setting Email Customized Class

Prerequisite: The Email Customized Class file has been created and put in the folder. For details about the class file and the folder, refer *Using Email Notifications* in *Interstage BPM Developers Guide*.

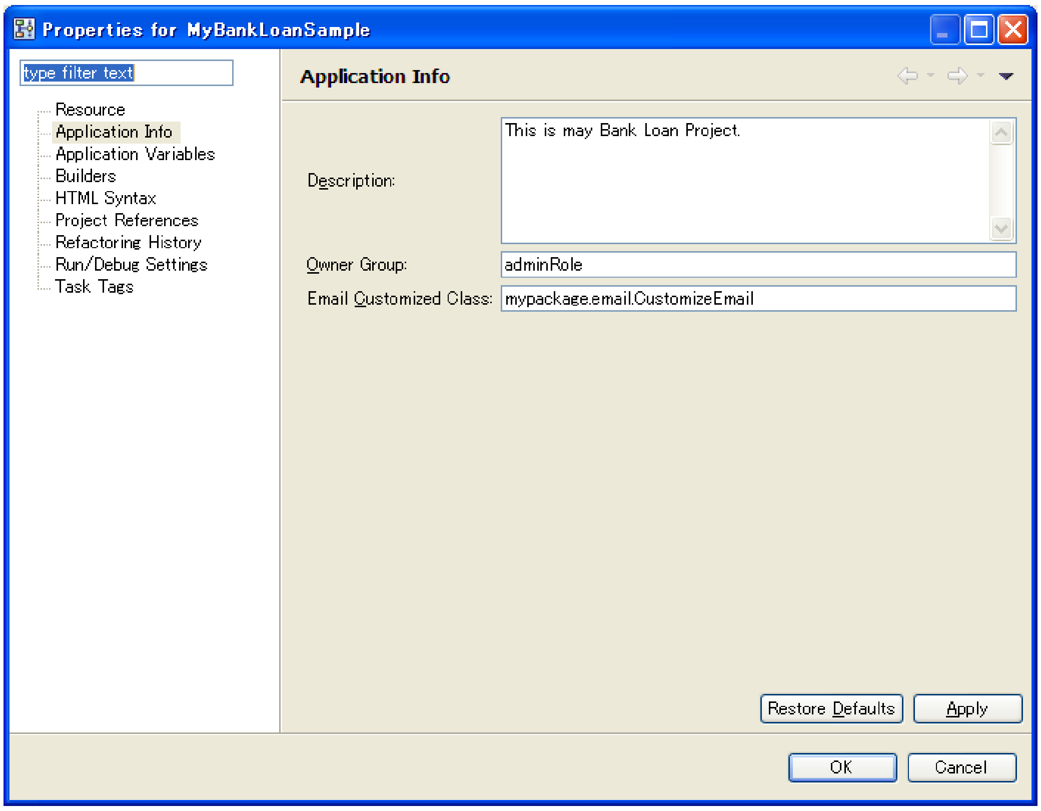
When a workitem is started, it is notified to the user in an email. Interstage BPM Studio provides the method to set the class in order to customize this email. If you do not set the class, system sends a default notification via email.

To set Email Customized Class:

1. In the **Navigator** view, right-click the project which you would like to set the class file for.
2. Click **Properties** in the popup menu.

   Properties dialog for the selected project is displayed.
3. Select the **Application Info** in the left pane.

   The following dialog is displayed:


Properties for MyBankLoanSample

Application Info

Description: This is may Bank Loan Project.

Owner Group: adminRole

Email Customized Class: mypackage.email.CustomizeEmail

Figure 68: Properties dialog

4. Optional: Provide the additional project information in the **Description** field.
5. Enter the name of the owner group in the **Owner Group** field. This is the group to which the user who owns the particular project, belongs.
6. Enter the class name in the **Email Customized Class** field.
7. Click **Apply** and then click **OK**.

Note: Class name is formatted as a fully qualified class name, for example, `mypackage.email.CustomizeEmail`.

4.1.17 Application Variables

Application Variables are the variables that you can define at application project level. By defining the Application Variables, you can share these variables across all the processes within a specific application project. This will save your effort for creating separate variables for each process.

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 Web Service location. This will allow you to change the Web Service 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 *Defining JavaScript Expressions* on page 371 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**.

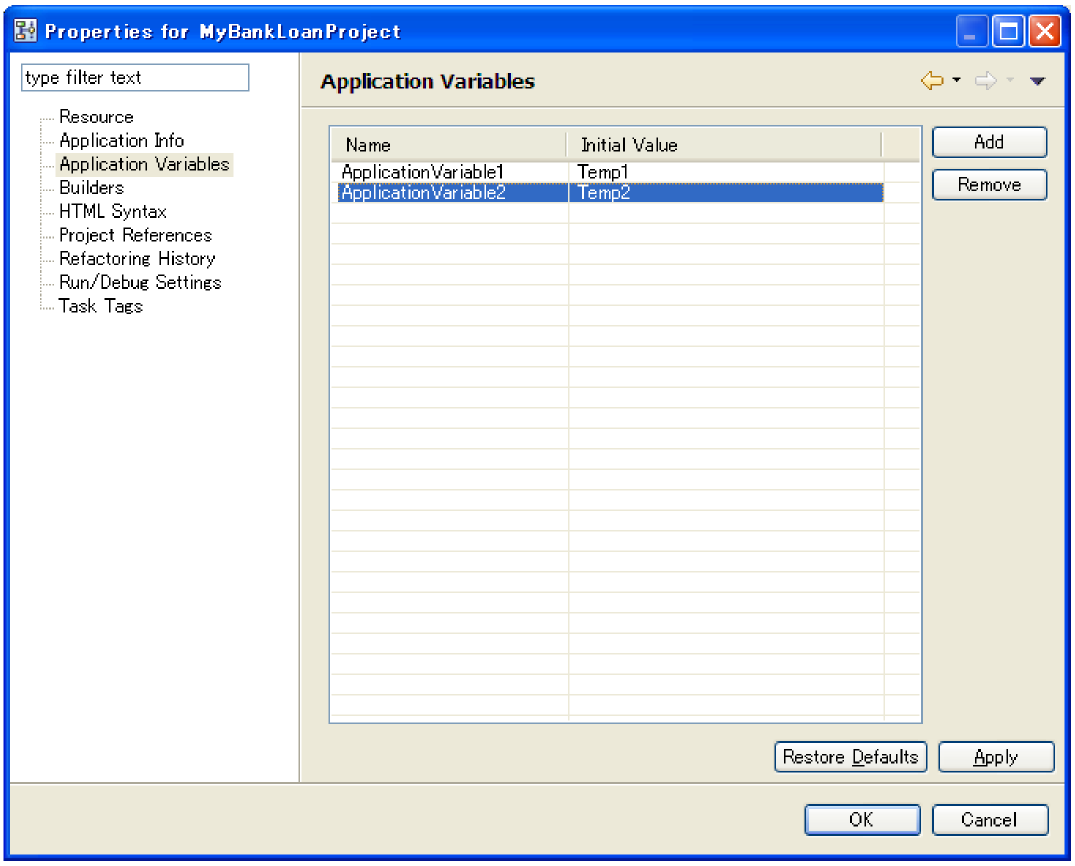
4.1.18 Defining Application Variables

To define Application Variables:

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

   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.



Name	Initial Value
ApplicationVariable1	Temp1
ApplicationVariable2	Temp2

Figure 69: Defining Application Variables

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

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.

Note: 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.

4.2 Managing Server Projects

This section explains the functions available for managing server projects.

Note: Make sure that you do not edit any process definition that is part of a Workflow Application project of a server project.

4.2.1 Creating Server Projects

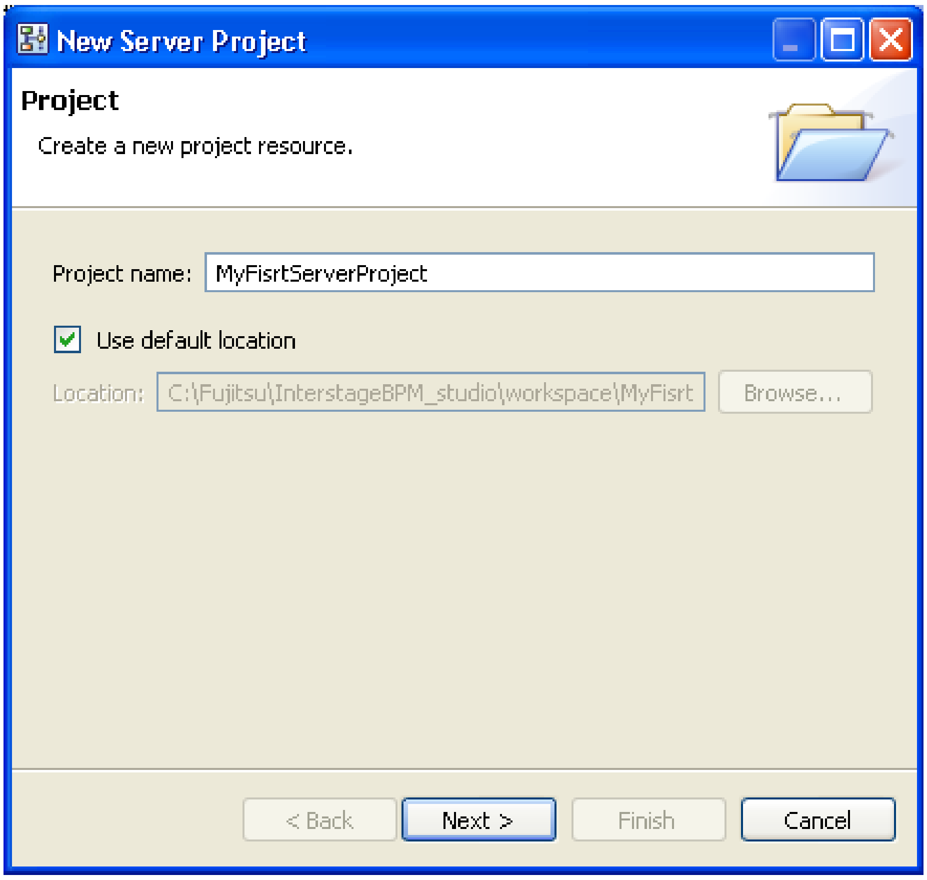
To create a project that accesses an Interstage BPM server:

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.


New Server Project

Project

Create a new project resource.

Project name: MyFisrtServerProject

✓ Use default location

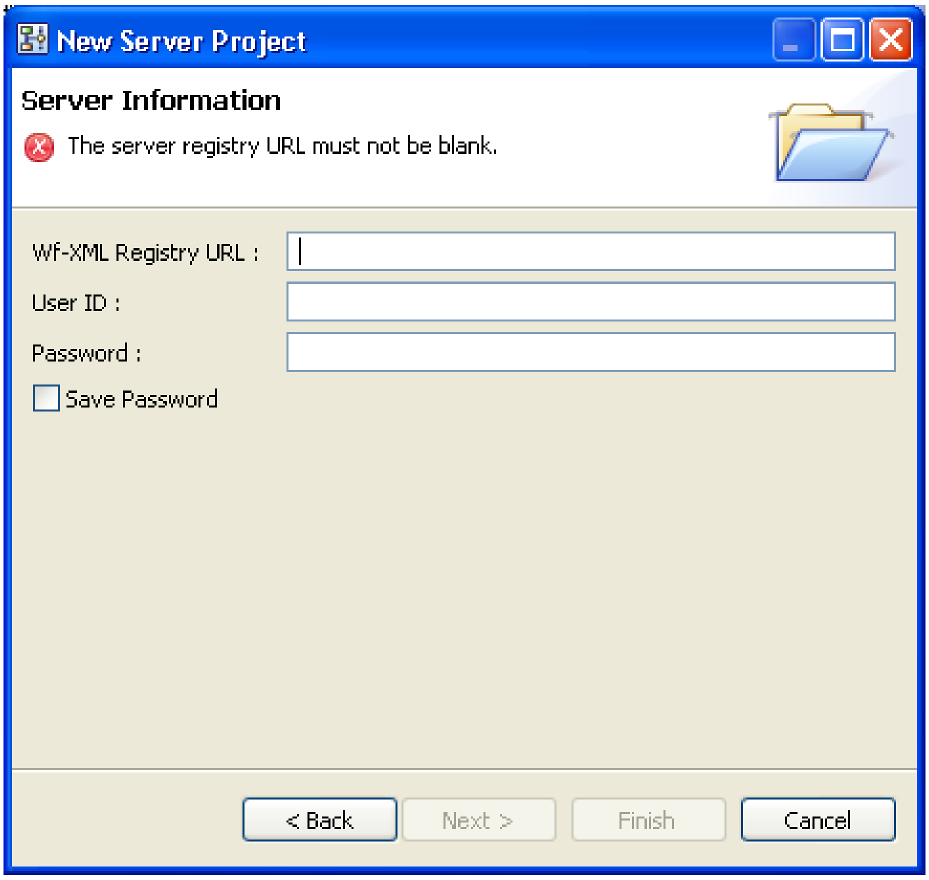
Location: C:\Fujitsu\InterstageBPM_studio\workspace\MyFisrt

Figure 70: Creating a Server Project

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 Interstage BPM server is opened.

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



New Server Project

Server Information

The server registry URL must not be blank.

Wf-XML Registry URL :

User ID :

Password :

☐ Save Password

Figure 71: Providing server information

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, `ibpmserver` is the hostname and `49950` is the port.

```
http://ibpmserver:49950/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 Interstage BPM 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 Interstage BPM Server.

8. Click **Finish**.

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

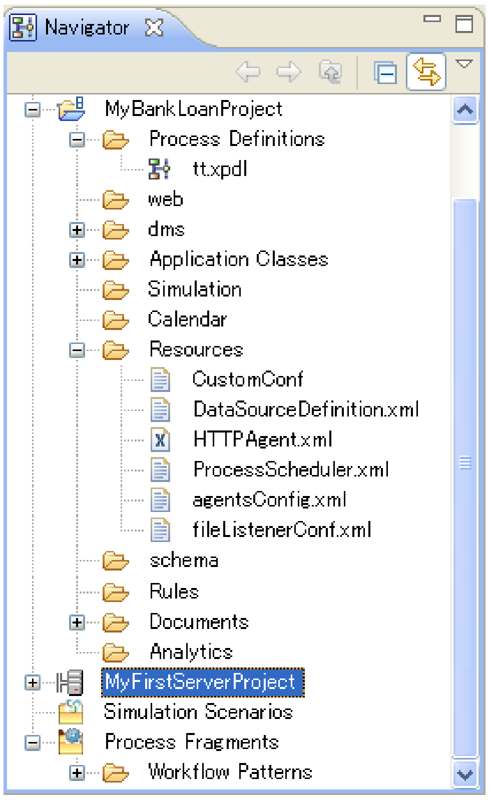


Figure 72: Displaying the new server project

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

Note: You have to install the Interstage Analytics plugin before you can use the Analytics function for the new server project.

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.

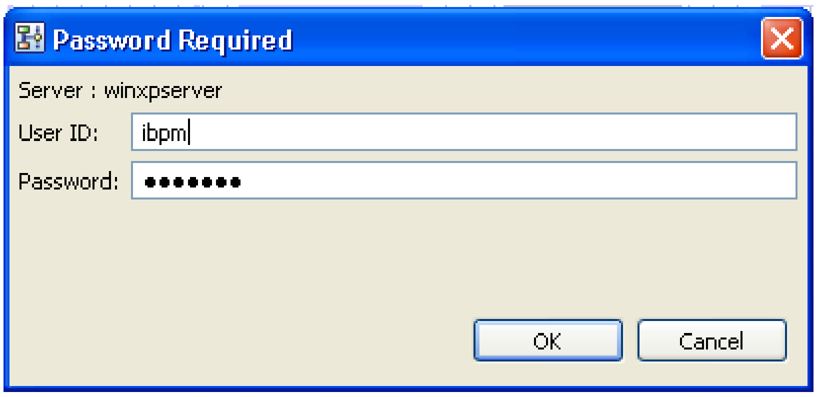


Figure 73: Entering a Server Password

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, make sure that these files are available on the computer on which the Interstage BPM Studio is installed and that any file paths are adjusted. For more information, refer to the following sections:

- *Using Forms* on page 191
- *Creating Your Own Business Calendars* on page 178
- *Integrating ILOG JRules* on page 318
- *Integrating Blaze Rules* on page 315
- *Using Generic Java Actions* on page 345
- *Integrating Decision Tables*

Note: If logging in to the Interstage BPM 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:

- Make sure that the connection information required to access the Interstage BPM server is correct (Host Name, Port 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.

4.2.2 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.

4.2.3 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.

4.2.4 Removing Server Projects

You can remove server projects with all of their contents from Interstage BPM Studio.

Note: When you remove a server project, this means that the registered Interstage BPM 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**.

5 Managing Process Definitions

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

5.1 Creating Process Definitions

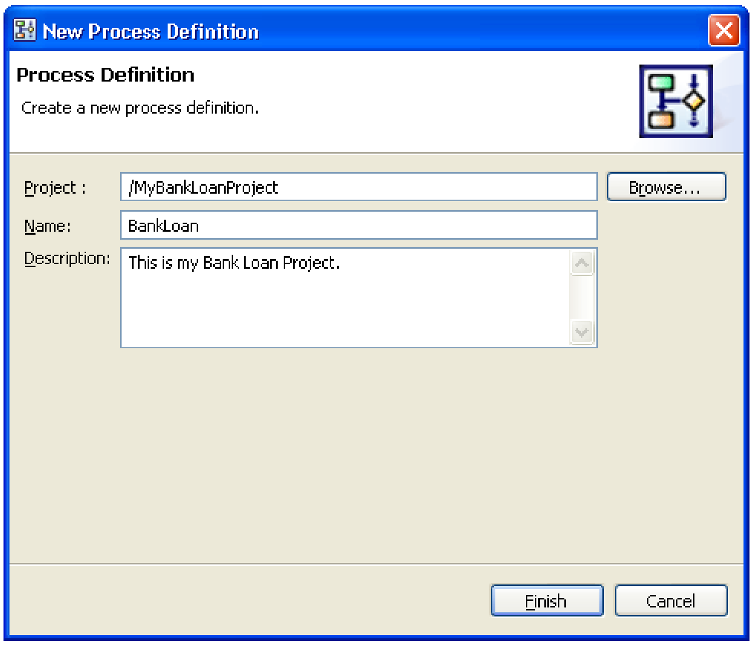
Prerequisite: You have created a project where the process definitions can be stored.

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.



Project : /MyBankLoanProject

Name: BankLoan

Description: This is my Bank Loan Project.

Figure 74: Creating a Process Definition

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'.
- Even if you create a new process definition, it might not be complete due to some errors. These errors are displayed in the **Problems** view.

5.2 Validating Process Definitions

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

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

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 *Problems View* on page 66.

5.3 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 cancel saving by clicking **No**. You can then fix the errors and try saving the process definition again.

5.4 Saving a Process Definition to a Different Name or Project

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

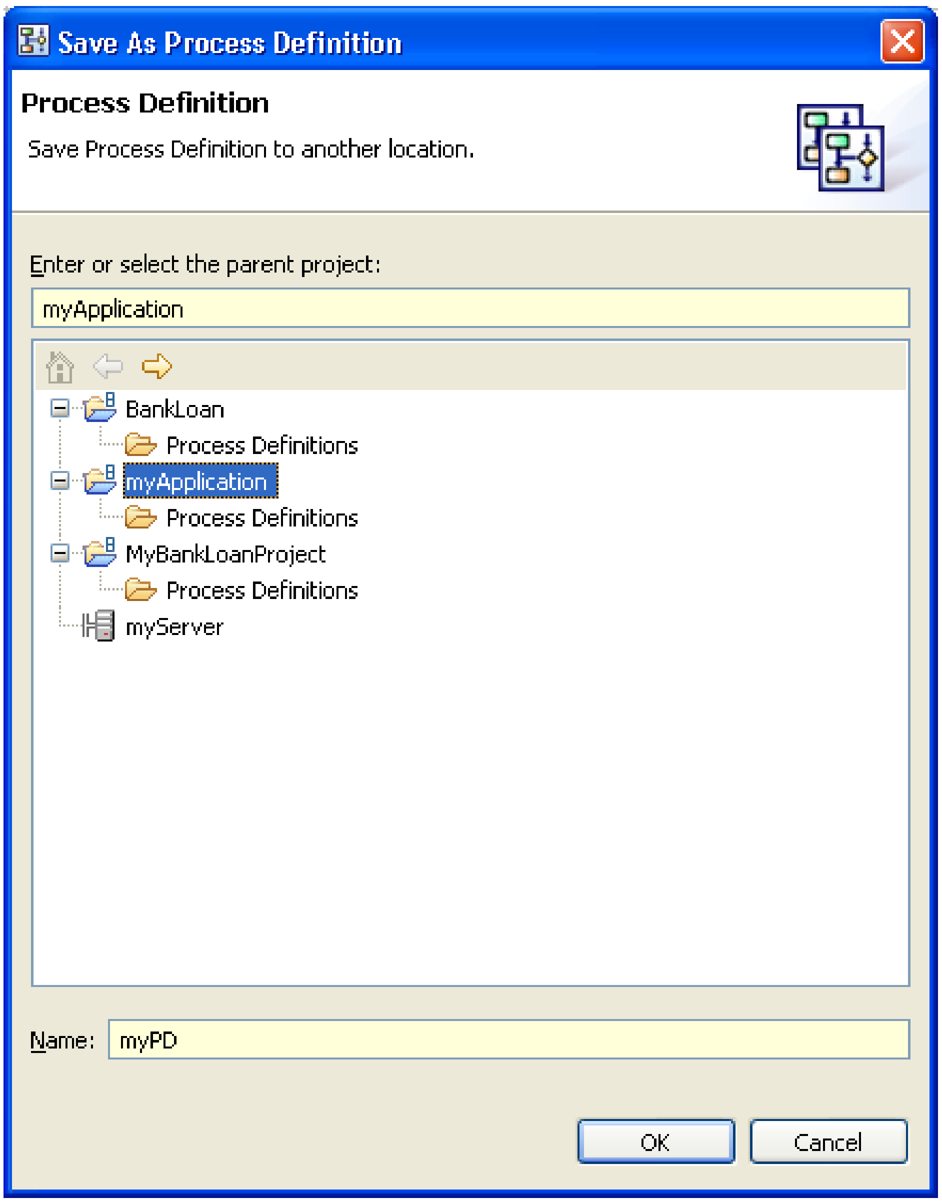
1. Click the Process Definition editor that displays the process definition to be saved.
2. Select **File** > **Save As**.
3. If the process definition is not valid, a message is displayed telling you so. You can display detailed information on the errors that have been found. You have the following options:
   - You can save the process definition anyway by clicking **Yes**.

- You can cancel saving 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.


Save As Process Definition

Process Definition

Save Process Definition to another location.

Enter or select the parent project: myApplication

- BankLoan
  - Process Definitions
- myApplication
  - Process Definitions
- MyBankLoanProject
  - Process Definitions
- myServer

Name: myPD

Figure 75: Saving a Process Definition to a Different Name or Project

6. Click **OK**.

5.5 Defining the Process Definition Name and Description

The name of a process definition is used to identify it. You can use a description to provide additional information on the process.

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

To define 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.

5.6 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.

5.7 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. If the parent process instance tries to call a subprocess definition with a version number that does not exist, an error is thrown.

Note: You can choose to use the same versions of only subprocesses.

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.

5.8 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 know the technical details of the Interstage BPM. Their need is to know 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 knowing 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 *Time and Day Codes for Advanced Due Dates and Timers* on page 176 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

You can access all the reports in Interstage BPM Studio. PDF and HTML reports open in their respective formats. MS-PowerPoint reports open in Studio.

Refer *Creating Process Definition Reports* on page 128 to know how to create reports.

5.8.1 Creating Process Definition Reports

Pre-requisites

- **Generate Process Documentation** option 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** command.

   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**.

Note: You can select more than one options 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, the **Generate Process Documentation** command cannot be used.

Note: When you use the Interstage BPM Studio as Eclipse Plug-In, the **Generate Process Documentation** command cannot be used.

Note: When the **Generate Process Documentation** command 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 the **Generate Process Documentation** command, 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.

5.9 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 an Interstage BPM 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**.

5.10 Importing Process Definitions

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

You can import process definitions in XPDL format into Interstage BPM Studio.

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, make sure that these files are available on the machine where you import it. For more information, refer to the following sections:

- *Using Forms* on page 191
- *Creating Your Own Business Calendars* on page 178
- *Integrating ILOG JRules* on page 318
- *Integrating Blaze Rules* on page 315
- *Using Generic Java Actions* on page 345
- *Integrating Decision Tables*

5.11 Exporting Process Definitions

You can export process definitions from Interstage BPM 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 an Interstage BPM server. Refer to section *Exporting Workflow Application Projects* on page 100 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 an Interstage BPM Server by creating a server project (refer *Creating Server Projects* on page 117 to know how you can create server projects) and transferring them to the server project, or by using the Interstage BPM Console. Refer to the *Interstage Business Process Manager User's Guide* for instructions.

Note: If the process definition references external files like forms, business calendars, cascading style sheets, Java classes, or rules files, make sure that these files are available on the machine to which you export it. For more information, refer to the following sections:

- *Using Forms* on page 191
- *Creating Your Own Business Calendars* on page 178
- *Integrating ILOG JRules* on page 318
- *Integrating Blaze Rules* on page 315
- *Using Generic Java Actions* on page 345
- *Integrating Decision Tables*

5.12 Sending Process Definitions to a Server

Prerequisite: You have locally created one or more process definitions in Workflow Application projects. One or more Interstage BPM Servers are connected, i.e. 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 an Interstage BPM 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.

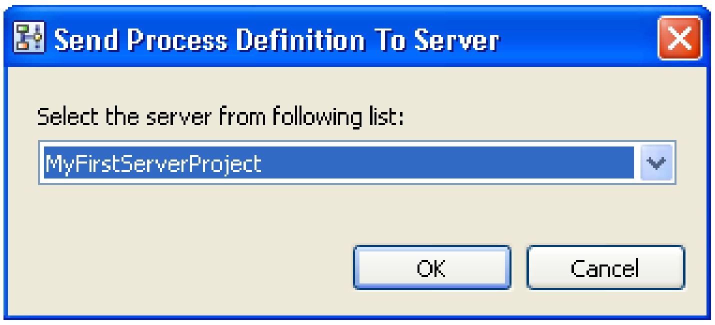

Send Process Definition To Server
Select the server from following list:
MyFirstServerProject
OK
Cancel

Figure 76: Sending Process Definitions to the Server

Note:	If you have created a server project in SaaS mode, the server project is created on the specified tenant which is on the Interstage BPM Server. The selected process definition is sent to the server project created on the tenant. Refer *Interstage BPM Studio Features* on page 17 for more information about SaaS mode.

Note:	If you have created a server project in non-SaaS mode, the server project is created on the default tenant.

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 *Creating Server Projects* on page 117 for details.

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, make sure that these files are available on the Interstage BPM Server. For more information, refer to the following sections:

- *Using Forms* on page 191
- *Creating Your Own Business Calendars* on page 178
- *Integrating ILOG JRules* on page 318
- *Integrating Blaze Rules* on page 315
- *Using Generic Java Actions* on page 345
- *Integrating Decision Tables*

5.13 Copying Process Definitions

You can copy process definitions easily 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.

5.14 Renaming Process Definitions

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.

5.15 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**.

5.16 Removing Process Definitions

When you remove a process definition, it is removed from the Interstage BPM 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.

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 section *Adding and Editing Nodes* on page 135.
2. Add the arrows to connect your nodes.

   For details, refer to section *Adding and Editing Arrows* on page 139.
3. Optional: Add swimlanes to visually group activities performed by the same Role.

   For details, refer to section *Adding and Editing Swimlanes* on page 142.
4. Optional: Add groups to visually group activities of the same category.

   For details, refer to section *Adding and Editing Groups* on page 145.
5. Optional: If you want to make comments on your process definition, add annotations.

   For details, refer to section *Adding Annotations* on page 148.
6. Optional: Define the owner of the process instances created from the process definition.

   For details, refer to section *Assigning Process Instance Owners* on page 150.
7. Assign all activities to Roles.

   For details, refer to section *Assigning Activities to Roles* on page 150.
8. Define the information that process participants need to access, modify, or add.

   For details, refer to section *Specifying User Defined Attributes* on page 151.
9. Create forms and associate them with activities.

   For details, refer to chapter *Using Forms* on page 191.
10. If you have added Voting Activity Nodes, define the voting rules.

    For details, refer to section *Defining Voting Rules* on page 182.
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 *Using Due Dates and Timers* on page 169.
12. If the process is to start at a particular date, define a timer for the process definition.

    For details, refer to section *Defining Timers* on page 171.
13. If you have added Delay Nodes, define a timer for them.

    For details, refer to section *Defining Timers* on page 171.
14. If you have added Conditional Nodes or Complex Conditional Nodes, define the required conditions.

    For details, refer to sections *Defining Conditions* on page 185 and *Defining Complex Conditions* on page 188.
15. If you need to add additional properties for process definitions, nodes, and arrows, define User Extended Attributes.

    For details, refer to section *Defining User Extended Attributes* on page 167.

Note: You can also use pre-defined process fragments to add pallette elements to your Process Definition. Refer to *Using Process Fragments* on page 189 for more information.

6.1 General Procedures

The following sections explain how to zoom, how to undo and redo changes, and how to print.

6.1.1 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 didn't want to undo the change, right click the empty space in the Process Definition editor and select **Redo**.

6.1.2 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.

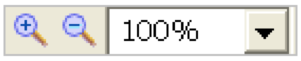


Figure 77: Zooming Tools in the Toolbar

6.1.3 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**.

6.2 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.

Note: You can also use pre-defined process fragments to add pallette elements to your Process Definition. Refer to *Using Process Fragments* on page 189 for more information.

6.2.1 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. Make sure to add at least one Exit Node.

To add a node:

1. Display the process definition in the Process Definition editor.
2. Display the palette.

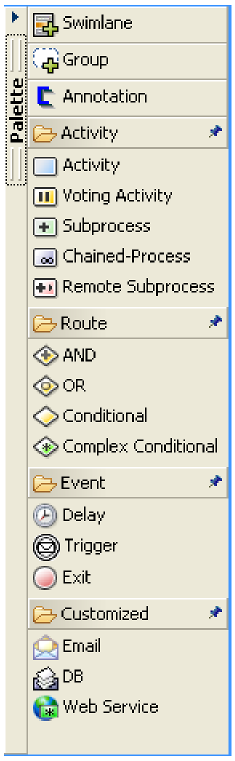


Figure 78: Using the Palette to Select the Type of Node

3. Click the type of node you want to add.
4. In the Process Definition editor, point to the area where you want to place the node. Click to add the node.

6.2.2 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.

6.2.3 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 *Setting Priority for an Activity* on page 288.

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.

6.2.4 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**).

Example

The following example shows how to align two nodes to the left edge.

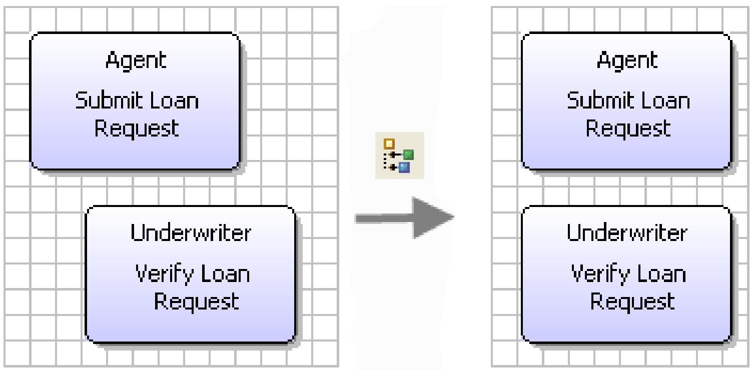


Figure 79: Aligning Nodes to the Left Edge

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 *Adding and Editing Swimlanes* on page 142; for information on how to work with groups, refer to section *Adding and Editing Groups* on page 145.

6.2.5 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.

6.2.6 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.

   Interstage BPM 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.

6.2.7 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.

6.3 Adding and Editing Arrows

Arrows connect the nodes in your process definitions. With arrows, you define the flow of events.

Note: You can also use pre-defined process fragments to add pallette elements to your Process Definition. Refer to *Using Process Fragments* on page 189 for more information.

The following sections tell you how to add and remove arrows, how to define their basic properties and how to change their shape.

6.3.1 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.

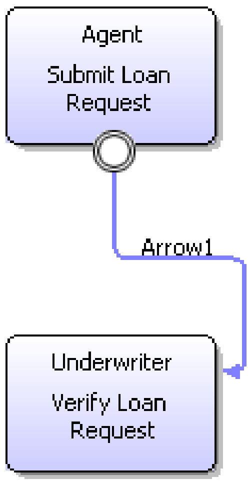


Figure 80: Adding an Arrow

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.

6.3.2 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
  - Remote Subprocess Node.
- Naming arrows is especially important for Voting Activity Nodes, Conditional Nodes, and Complex Conditional Nodes. These nodes use arrow names to route the process flow according to criteria that you set.

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.

6.3.3 Changing the Arrow Shape

You can change the shape of an arrow dragging its bend points. Interstage BPM Studio automatically adds additional bend points or removes bend points while you are dragging the arrow.

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.

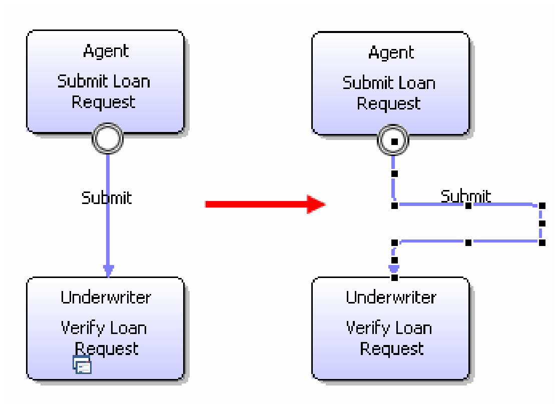


Figure 81: Reshaping an Arrow

6.3.4 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
- Remote Subprocess Node.

Therefore, you might need to change the arrow name before reconnecting the arrow.

6.3.5 Removing Arrows

To remove an arrow:

1. Select the arrow to be removed.
2. Right click and select **Delete**.

6.4 Adding and Editing Swimlanes

With Interstage BPM Studio, you can use swimlanes to visually group activities performed by the same Role.

Note: You can also use pre-defined process fragments to add pallette elements to your Process Definition. Refer to *Using Process Fragments* on page 189 for more information.

The following sections explain how to add and remove swimlanes, and how to change their appearance.

6.4.1 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.

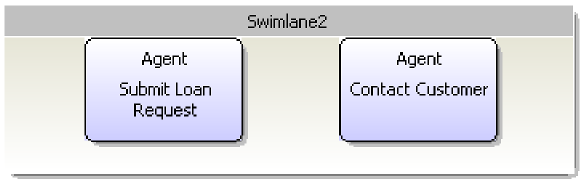


Figure 82: Adding Swimlanes

6.4.2 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.

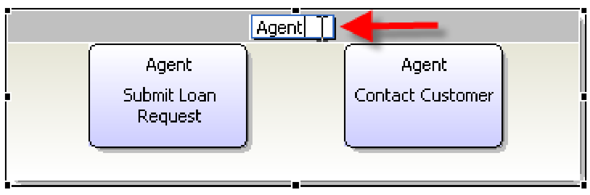


Figure 83: Defining the Swimlane Title

Alternatively, you can type the new title in the Properties view.

6.4.3 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.

6.4.4 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.

6.4.5 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.

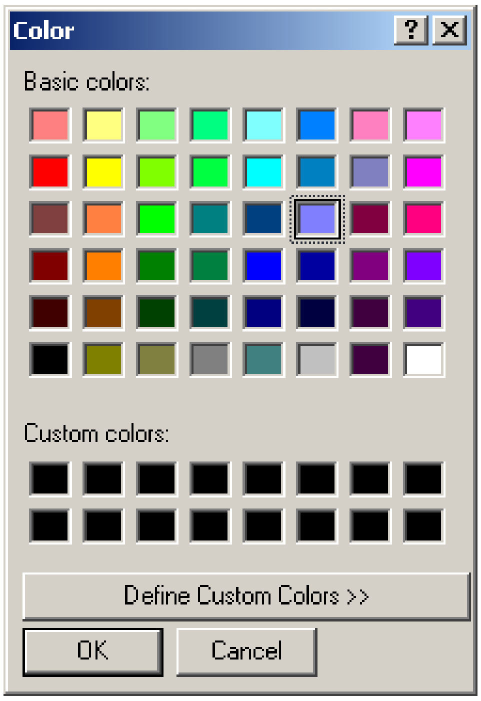


Figure 84: Picking a Color from the Palette

4. Pick a color from the palette.
5. Click **OK**.

6.4.6 Changing Swimlane Style

You can place the swimlane title on the left-hand side or on the upper side of a swimlane.

You can change this setting for all swimlanes of a process definition.

To change swimlane style for swimlanes of a process definition:

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**.

6.4.7 Cutting, Copying and Pasting Swimlanes

You can cut, copy, and paste a single swimlane at a time.

- **To cut a swimlane:**
  Select the swimlane. Right click and select **Cut** from the pop-up menu.
- **To copy a swimlane:**
  Select the swimlane. Right click and select **Copy** from the pop-up menu.
- **To paste a swimlane:**
  Select **Edit** > **Paste**.
  The swimlane is inserted near the original swimlane.

6.4.8 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.

6.4.9 Removing Swimlanes

To remove a swimlane:

1. Select the swimlane.
2. Right click and select **Delete** from the pop-up menu.

6.5 Adding and Editing Groups

With Interstage BPM 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.

6.5.1 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.

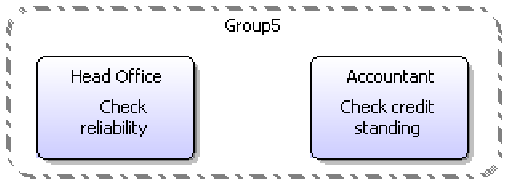


Figure 85: Adding Groups

6.5.2 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.

6.5.3 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.

6.5.4 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:
  a) Select the group.
  b) Point to one of the sizing handles and drag the group to the new size.
- To scale groups to the same width or height:
  a) Select the groups to be scaled. Keep the <Shift> or <Ctrl> key pressed while selecting the groups.
  b) 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.

6.5.5 Cutting, Copying and Pasting Groups

You can cut, copy, and paste a single group at a time.

- **To cut a group:**
  Select the group. Right click and select **Cut** from the pop-up menu.
- **To copy a group:**
  Select the group. Right click and select **Copy** from the pop-up menu.
- **To paste a group:**
  Select **Edit** > **Paste**.
  The group is inserted near the original group.

6.5.6 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.

6.5.7 Removing Groups

To remove a group:

1. Select the group.

2. Right click and select **Delete** from the pop-up menu.

6.6 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.

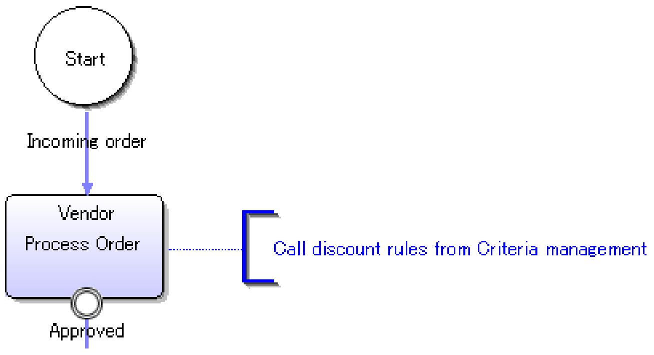


Figure 86: Adding Annotations and Connecting them to Nodes

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.

6.7 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, Delay and Trigger Nodes.

Note:
- Only completed work items can be recalled.
- Only single level recall of work item is possible. Multiple level recall of work items cannot be done.

To enable recall for an activity:

1. Select the Activity, Voting, Delay or Trigger 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.

6.8 Enabling Future Work Items on Activity Node

Future Work Items are the work items that the users may be assigned in future. As a user, Future Work Items help you to be aware of work items that may be assigned to you later so you can plan your tasks in advance.

Future Work Items generated for a user may not be accurate since task assignment can vary during the process execution.

Future Work Items are displayed as a list for a particular user. User cannot actually accept them or reject them or perform any kind of actions on them since these are displayed as a read-only list.

Note: Future Work Items are generated only on activity nodes.

To enable Future Work Items on an activity node:

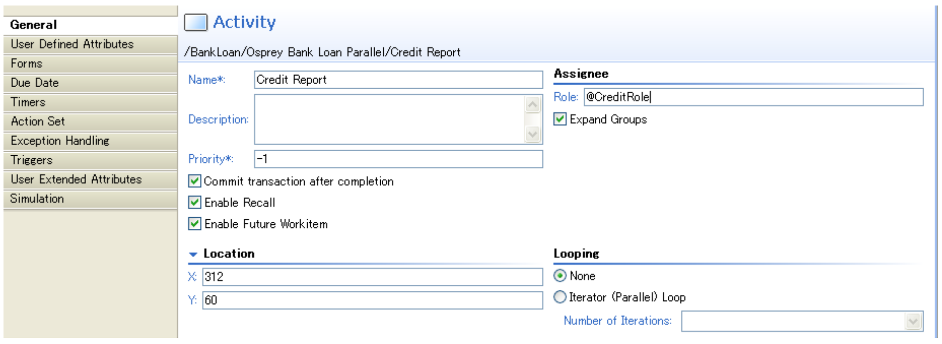
1. Open the process definition.
2. Select the activity node on which you want to enable Future Work Items.
3. In the **General** tab of **Properties** view of the activity, select the checkbox **Enable Future Workitem**.

Note: Future Work Items cannot be set for the activity nodes that have Agents defined on them.

Note: Future Work Items cannot be set for the activity nodes configured as Iterator Nodes.

4. Save the process.

   The following figure shows how Future Work Items can be enabled on an activity node:



/BankLoan/Osprey Bank Loan Parallel/Credit Report

Name*: Credit Report
Description:
Priority*: -1
[✓] Commit transaction after completion
[✓] Enable Recall
[✓] Enable Future Workitem

Location
X: 312
Y: 60

Assignee
Role: @CreditRole
[✓] Expand Groups

Looping
[✓ None]
Iterator (Parallel) Loop
Number of Iterations:

Figure 87: Enabling Future Work Items

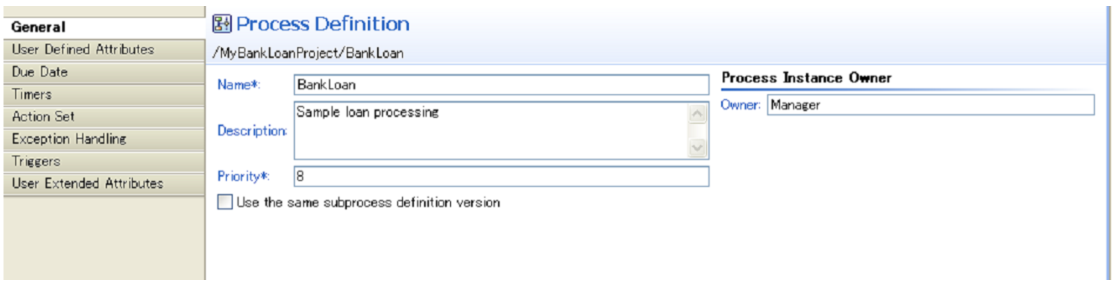
6.9 Assigning Process Instance Owners

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.



/MyBankLoanProject/BankLoan

Name*: BankLoan

Description: Sample loan processing

Priority*: 8

Use the same subprocess definition version

Process Instance Owner

Owner: Manager

Figure 88: Assigning Process Instance Owners

6.10 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 *Assigning an Activity to a User* on page 282 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.

6.11 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:

- Individual Work Items

  By default, one work item is generated for every user having the Role to which the Activity Node has been assigned. As soon as a user accepts the work item, the other work items associated with the activity are set inactive. This way, Interstage BPM prevents users from duplicating each other's work.

- Group Work Item

  As an alternative, one work item is generated for the entire group of users having the Role. Any user of the group can work on the activity by accepting the group work item.

  Group work items are especially useful in case of frequent changes to the users having a Role and when many users are having one and the same Role (for performance reasons).

To configure work item generation for an Activity Node:

1. Select the Activity Node or Compound Activity to display the **Properties** view for the node.
2. In the **Assignee** area, check the setting of **Expand Groups**:
   - If you want individual work items to be generated, make sure that the check box is selected.
   - If you want a group work item to be generated, clear the check box.

6.12 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 are unique across a process definition so that two different process definitions can have the same identifier.

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:

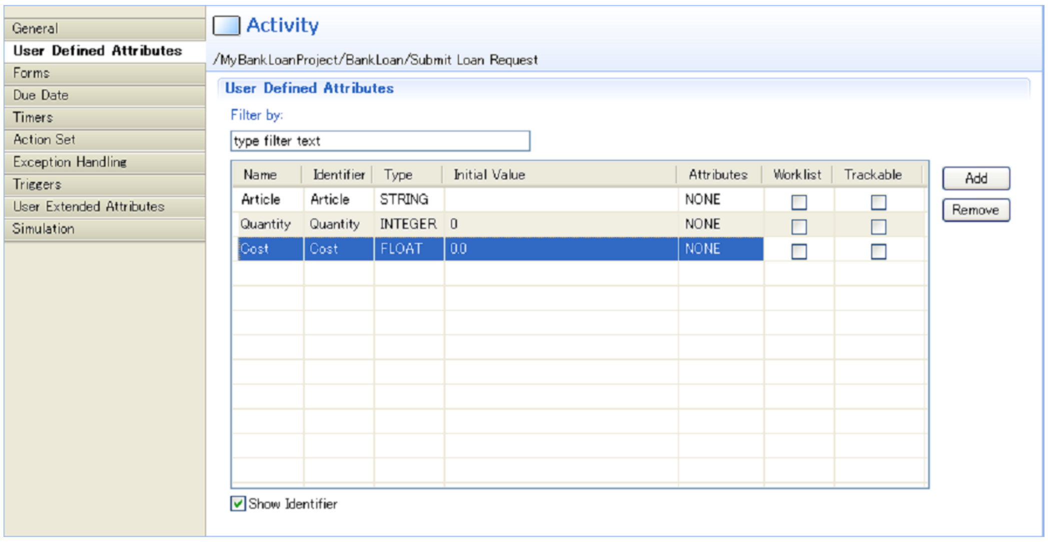
Field	Description
Name	Name of the UDA The UDA name is descriptive and user-defined. It can contain up to 64 characters. The name must not start with two underscores (__) because two underscores are used as a prefix of UDAs created and maintained by the system.

Field	Description
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. Refer to the *Interstage Business Process Management Server Administration Guide* for more details. **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, CUSTOM CUSTOM is the type which you can define arbitrarily. When you add a new type of CUSTOM UDA, you can see it in the drop-down list.
Initial Value	Initial value of the UDA Process participants can change this value. If you select a UDA of type XML or CUSTOM, two additional buttons are displayed. You can use these buttons to configure the XML data type. For more information, refer to section *Specifying User Defined Attributes of Type XML* on page 154 or *Specifying User Defined Attributes of Type CUSTOM* on page 157. 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.

Field	Description
Attributes	This column is only used with ARIS Process Performance Manager (PPM). It indicates whether the UDA will be exported to ARIS PPM and how it is used by ARIS PPM. If you want the UDA to be exported, specify whether the UDA is to be interpreted as a dimension or as a measure. For more information, refer to the *ARIS Process Performance Manager Integration Guide*. If you don't want the UDA to be exported, use the default value **NONE**. If you are not using ARIS PPM, you can ignore this column.
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:



/MyBankLoanProject/BankLoan/Submit Loan Request

User Defined Attributes

Filter by:

type filter text

Name	Identifier	Type	Initial Value	Attributes	Worklist	Trackable
Article	Article	STRING		NONE	☐	☐
Quantity	Quantity	INTEGER	0	NONE	☐	☐
Cost	Cost	FLOAT	0.0	NONE	☐	☐

☑ Show Identifier

Figure 89: Specifying UDAs

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, make sure 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, Interstage BPM automatically generates default identifiers (for example, `'Variable1', 'Variable2', 'Variable3'`).

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.

6.12.1 Specifying User Defined Attributes of Type XML

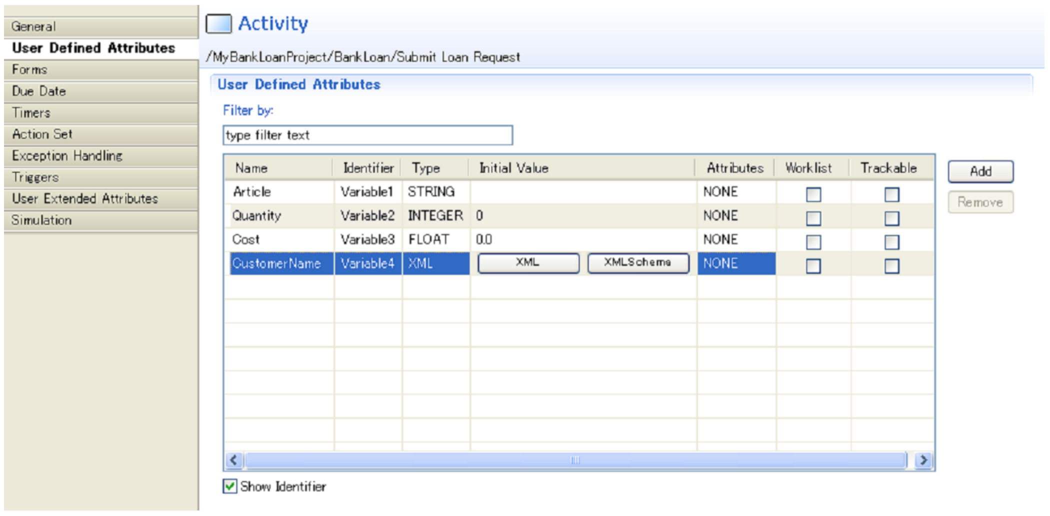
In Interstage BPM Studio, you can create User Defined Attributes (UDAs) of type XML to enhance the flexibility of the system.

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. You can use these buttons only if you select XML or CUSTOM type of UDA.



Activity

/MyBankLoanProject/BankLoan/Submit Loan Request

User Defined Attributes

Filter by:

type filter text

Name	Identifier	Type	Initial Value	Attributes	Worklist	Trackable
Article	Variable1	STRING		NONE	☐	☐
Quantity	Variable2	INTEGER	0	NONE	☐	☐
Cost	Variable3	FLOAT	0.0	NONE	☐	☐
CustomerName	Variable4	XML	XML XMLSchema	NONE	☐	☐

☑ Show Identifier

Figure 90: Displaying UDAs

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.

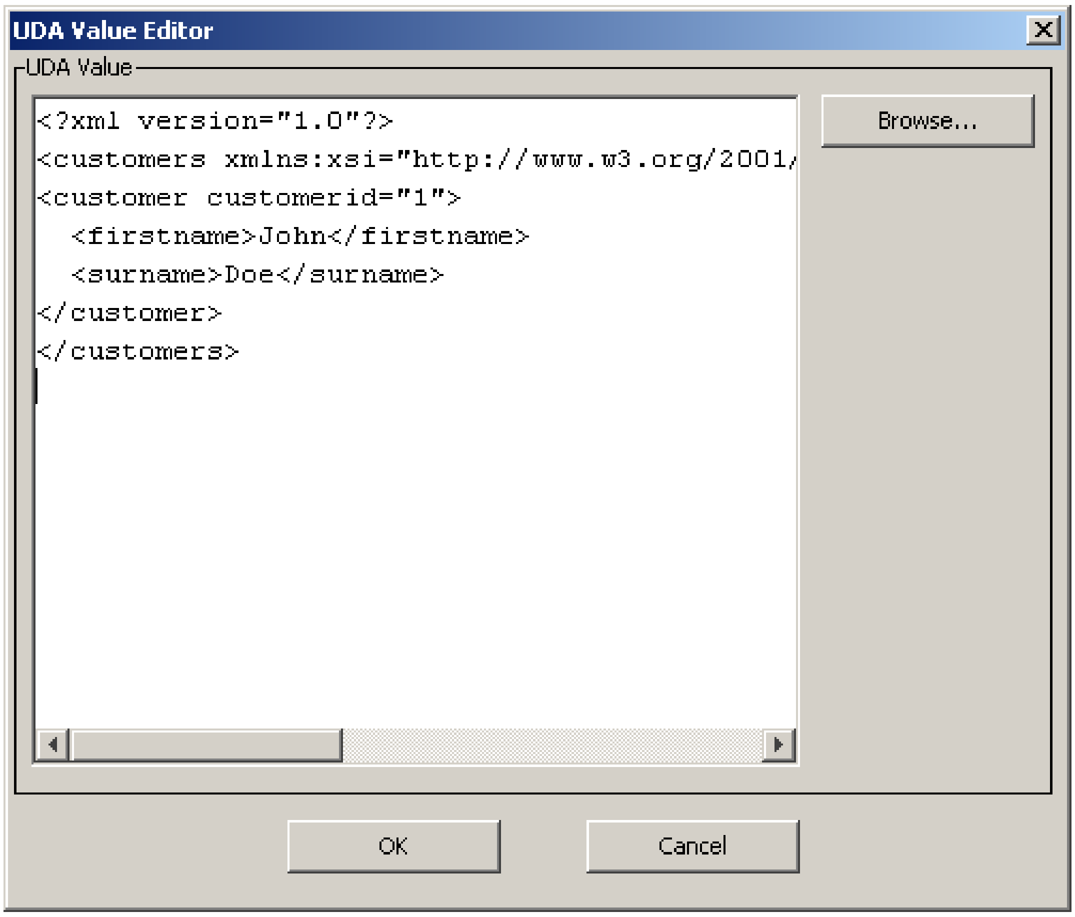
   Refer to section *Specifying User Defined Attributes* on page 151 for a detailed description of these parameters.

5. In the **Initial Value** column, select the **XML** button.

   The **UDA Value Editor** dialog 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 a UDA value in the **UDA Value Editor** dialog:



```
<?xml version="1.0"?>
<customers xmlns:xsi="http://www.w3.org/2001/
<customer customerid="1">
  <firstname>John</firstname>
  <surname>Doe</surname>
</customer>
</customers>
```

Figure 91: Specifying a UDA value

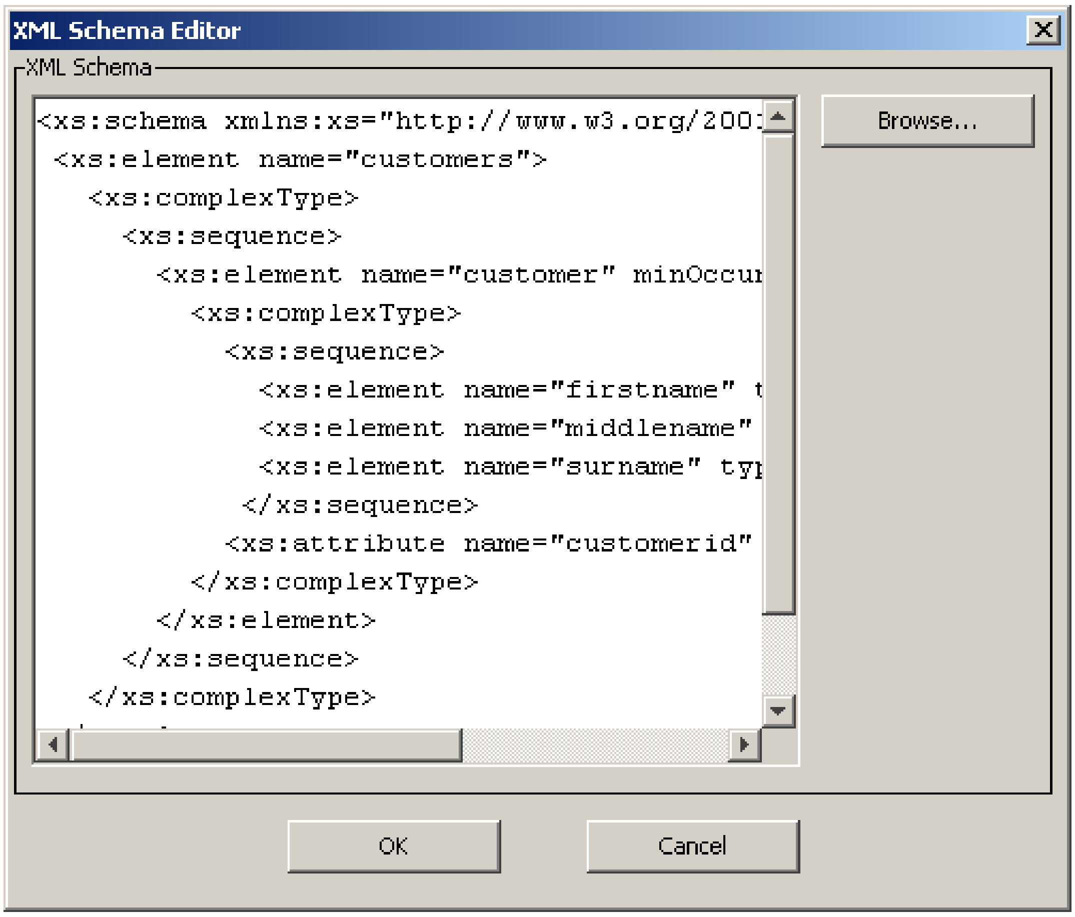
7. Optional: In the **Initial Value** column, select the **XML Schema** button.

   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:



```
<xs:schema xmlns:xs="http://www.w3.org/2001
 <xs:element name="customers">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="customer" minOccur
         <xs:complexType>
           <xs:sequence>
             <xs:element name="firstname" t
             <xs:element name="middlename"
             <xs:element name="surname" typ
            </xs:sequence>
           <xs:attribute name="customerid"
         </xs:complexType>
       </xs:element>
     </xs:sequence>
   </xs:complexType>
```

Figure 92: Specifying an XML 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**.

6.12.2 Specifying User Defined Attributes of Type CUSTOM

When using User Defined Attributes (UDAs) of type XML, you need to define the corresponding XML Schema each time you want to define a new UDA of type XML. If you intend to re-use part of a data structure of XML data type, you need to create many UDA definitions of the same structure, which is a drawback. In such a case, you can choose to work with UDAs of type CUSTOM.

A CUSTOM data type is basically XML, but its structure is pre-defined (with an associated XSD) and named in Interstage BPM Studio. This 'named structure' then serves as a new data type, which can be used to create new UDAs of that type.

Another advantage of using UDAs of type CUSTOM is that if the structure changes you can change the XSD file, and it is not needed to change each UDA.

XML Schema files (XSD files) can be created using the XML Schema Editor. You can also import existing XSD files into Interstage BPM Studio. For details, refer *Importing XML Schema File* on page 231.

CUSTOM data types are named using either of the following:

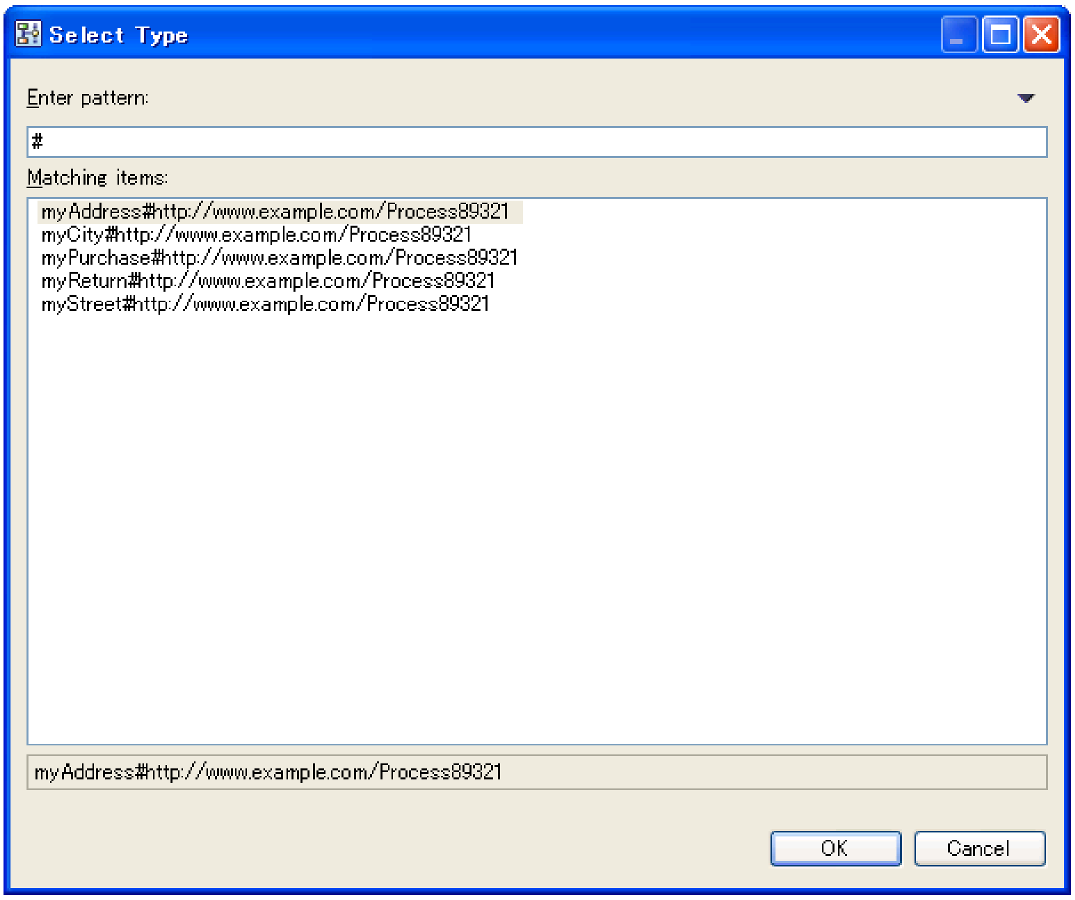
- Global elements and namespaces, in the format **`<global element name>#<namespaceURI>`**.
- XSDType and namespaces, in the format **`<XSDType name>@<namespaceURI>`**.

where `XSDType` means `xsd:complexType` and `xsd:simpleType`, and `<namespaceURI>` is the targetNamespace for the XSD in which the type is defined.

To specify a UDA of type CUSTOM:

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 **CUSTOM...** as type for the UDA.
4. In the **Select Type** dialog box, in **Enter pattern** field, begin to type the name of the CUSTOM UDA you want to choose.

Matching suggestions are displayed in the **Matching items** area.



```
#
```

```
myAddress#http://www.example.com/Process89321
myCity#http://www.example.com/Process89321
myPurchase#http://www.example.com/Process89321
myReturn#http://www.example.com/Process89321
myStreet#http://www.example.com/Process89321
```

```
myAddress#http://www.example.com/Process89321
```

Figure 93: Select Type

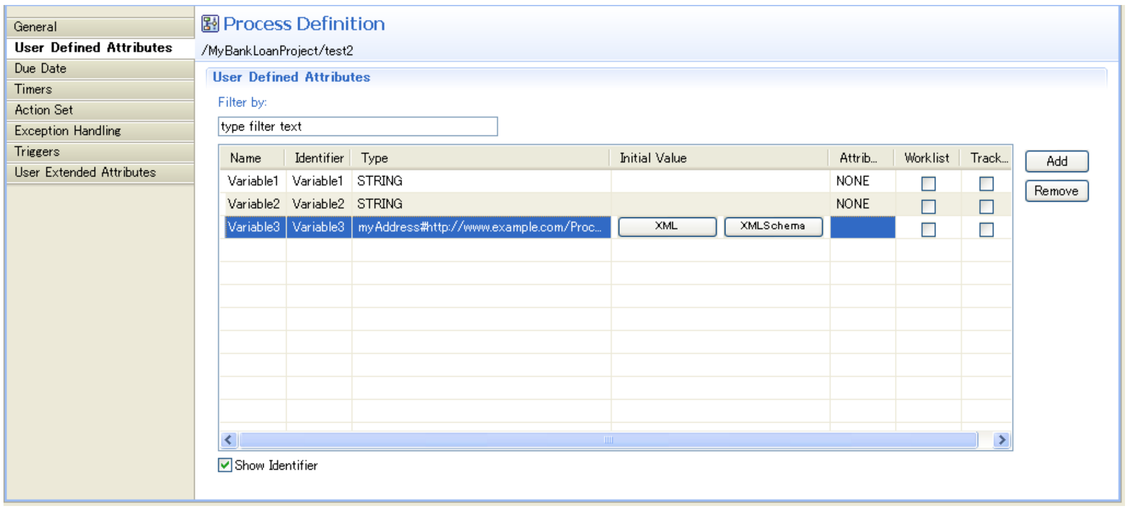
5. Select a CUSTOM UDA from the **Matching items** area.

   The selected item is displayed in a status box at the bottom of the **Select Type** dialog box.

Note: You can choose to show/hide the status box by using the toggle switch available at the top-right corner of the **Select Type** dialog box.

6. Click **OK**.

In the **User Defined Attributes** tab, the selected type is displayed against the UDA you are adding.



- General
- User Defined Attributes
- Due Date
- Timers
- Action Set
- Exception Handling
- Triggers
- User Extended Attributes

Process Definition

/MyBankLoanProject/test2

User Defined Attributes

Filter by:

type filter text

Name	Identifier	Type	Initial Value		Attrib...	Worklist	Track...
Variable1	Variable1	STRING			NONE	☐	☐
Variable2	Variable2	STRING			NONE	☐	☐
Variable3	Variable3	myAddress#http://www.example.com/Proc...	XML	XMLSchema		☐	☐

Add

Remove

☑ Show Identifier

Figure 94: Displaying UDA of Type Custom

7. In the **Name** and **Identifier** fields, specify the general parameters for the UDA of type XML.

   Refer to section *Specifying User Defined Attributes* on page 151 for a detailed description of these parameters.

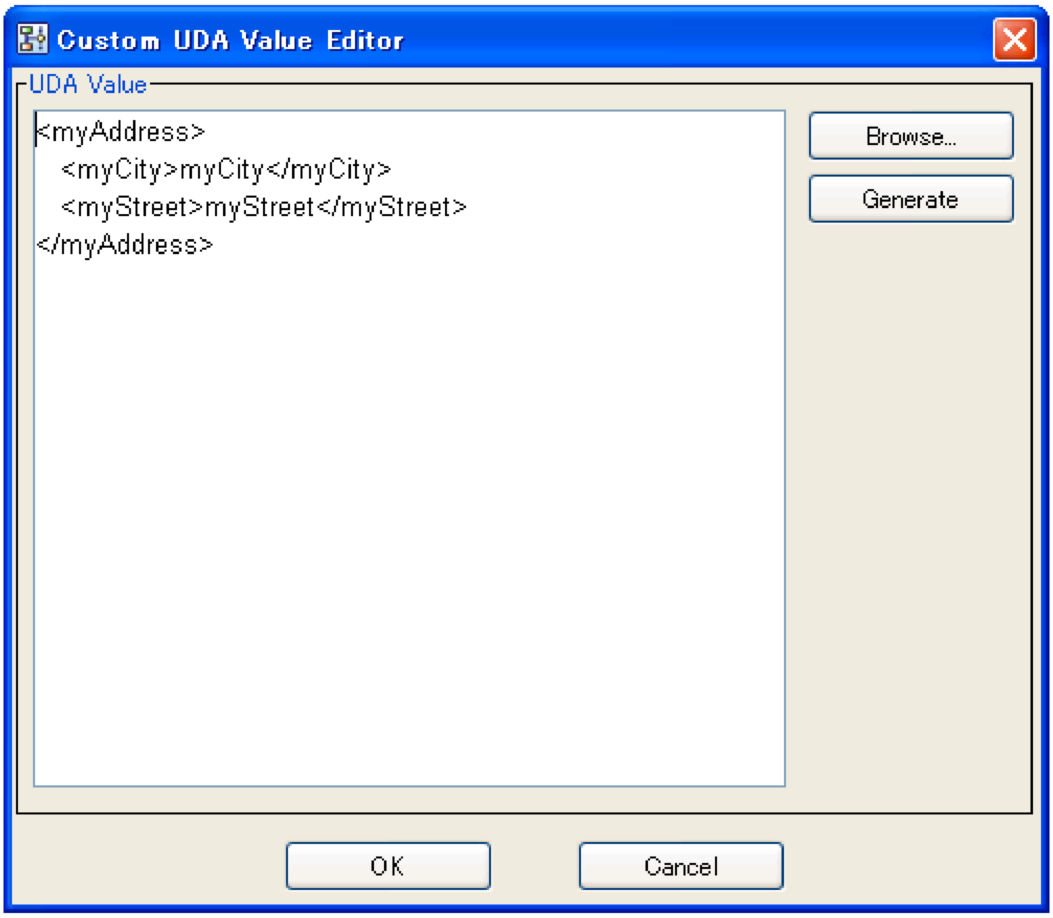
8. In the **Initial Value** column, select the **XML** button.

   The **Custom UDA Value Editor** dialog is displayed.

9. Type in a value for the UDA, browse for an XML file, or generate a UDA value.
   - **Type in a UDA value**: In the **UDA Value** field, type in a value for the UDA and click **OK**.
   - **Browse for a file**: Click the **Browse** button to select a file from the local file system. The **Custom UDA Value Editor** dialog displays the content of the file. Select a value from the content of this file, and click **OK**.
   - **Generate a UDA value**: Click the **Generate** button to generate an initial UDA value for the corresponding XML Schema. Change the value appropriately and click **OK**.

Note:
- Element node names are used as the default values of text nodes.
- The rules to control the number of occurrences of elements, such as `xsd:choice` or `minOccurs` etc., are not taken into account. Therefore, you will need to edit generated elements if necessary.
- The restrictions for attributes are also considered, thus, you will need to edit generated attributes if necessary.
- Namespaces are not supported.

The following example shows how to specify a UDA value in the **Custom UDA Value Editor** dialog:



```
<myAddress>
  <myCity>myCity</myCity>
  <myStreet>myStreet</myStreet>
</myAddress>
```

Figure 95: Specifying a UDA Value

10. Optional: In the **Initial Value** column, select the **XML Schema** button.

    The **XML Schema Editor** dialog is displayed. You can use this dialog to edit the XML Schema for the UDA. For details, refer *XML Schema Editor Views* on page 224.

11. Similar 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**.

When you select an element or a type of XML Schema, it is displayed in the **Type** column.

6.13 Looping Definition

In the process definition, two or more node instances might have to be generated without defining individual nodes for each of the node instances. The node instance generated when each node in the process definition is executed, is usually only one.

To enable looping behavior on a node, you can define the node behavior either as an Iterator (Parallel) Loop or Sequential Loop. In both the behaviors, two or more node instances are generated with one node definition. The node that has been defined as a loop node is either called the Iterator (Parallel) Loop node or the Sequential Loop node.

In Iterator (Parallel) Loop node, two or more node instances are concurrently generated at the same time. In Sequential Loop node, the node instance is generated one after another while it meets a certain requirement.

For example, in a typical purchase order processing scenario, it is required to repeat the same activity (for instance, **Approval**) for all the items in an **Order**. In this case, you do not need to define an Activity Node for the activity **Approval**, for each of the ordered items. You can define a single Activity Node. On this node, you can then set the number of node instances by using the User Defined Attributes while designing the process definition. Node instances are generated in parallel from the same Activity Node. This loop behavior is called Iterator (Parallel) Loop.

In Sequential Loop node, multiple node instances are generated sequentially 'while' a certain condition is being satisfied.

For example, in **Order Shipment** process, Sequential Loop Node enables an **Agent** to check the stock of ordered items. The **Agent** gets items (from **Order**) and automatically adds them to the shipment while there are items available in **Order**. This **Agent** repeats the same process sequentially until there are no more items in the **Order**.

The Sequential Loop condition and the number of the loop iterations can be specified, and the number of node instances can be set while designing the process definition.

Note: You cannot set Iterator (Parallel) Loop and Sequential Loop on the same node.

6.13.1 Configuring Node as Iterator (Parallel) Loop Node

In Process Definitions, a node may need to generate multiple node instances to eliminate the need of creating individual nodes. Multiple node instances can also be assigned to multiple users by defining only a single node.

For example, in a Process Definition for Order Management, it is required that the same activity (say Approval) be repeated for all items that are part of an Order. Configuring the ‘Approval’ activity node as an Iterator (Parallel) Loop node will help avoid the need to add a new activity node for each Order item in the Process Definition.

A node that generates multiple node instances is called an Iterator (Parallel) Loop node. While designing the Process Definition, you can specify the number of node instances to be generated for a node by selecting a UDA of type Integer.

Note: If a specified UDA for an iterator (parallel) loop node does not exist, process validation will fail and no process instance can be started from the process definition until the UDA is added.

The number of node instances the selected node generates is equal to the number value returned by the selected UDA. The maximum value that can be set to the UDA is related to the number of persons associated with the node. The maximum value that can be set for this UDA is 2000.

Note: You can configure a node as an Iterator (Parallel) Loop node only on Activity Node, Subprocess Node and Chained-Process Node.

Note: You can use only one outgoing arrow form an Activity Node configured as Iterator (Parallel)Loop node.

To configure an Activity and Subprocess Node as an Iterator (Parallel) Loop node:

1. Select the node that you want to configure as an Iterator (Parallel) Loop node.
2. Select the **Iterator (Parallel) Loop** radio button under **Looping** section in **General** tab of the **Properties** view.
3. Select the Integer type UDA for the value of iterations in **Number of Iterations** drop-down list.

To configure a Chained-Process node as an Iterator (Parallel) Loop node

1. Select the node that you want to configure as an Iterator (Parallel) Loop node.
2. In **General** tab of **Properties** view, select the Integer type UDA for the value of count in **Number of Iterations** drop-down list under **Iterator Setting**.

   The integer value returned by the selected UDA is considered as the count value for generating the node instances.

Note: If you do not select any UDA in the **Number of Iterations** drop-down list, the selected node will not be configured as an Iterator (Parallel) Loop node.

The Iterator (Parallel) Loop node is graphically represented as three vertical parallel lines in the node. The Activity Iterator (Parallel) Loop node and Subprocess Iterator (Parallel) Loop node are represented as shown in the figure below.

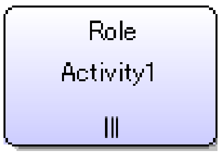


Figure 96: Activity Iterator (Parallel) Loop Node

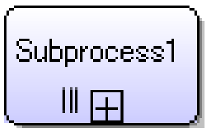


Figure 97: Subprocess Iterator (Parallel) Loop Node

Chained-Process Iterator (Parallel) Loop node is represented as shown in the figure below.

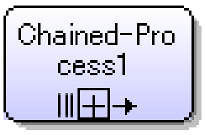


Figure 98: Chained-Process Iterator (Parallel) Loop Node

6.13.2 Configuring Node as Sequential Loop Node

Pre-requisite: Only 1 outgoing arrow is defined from the node configured as Sequential Loop Node.

Sequential Loop node generates node instances one after another (sequentially) while satisfying a certain requirement. You can configure an Activity Node and the Subprocess Node as Sequential Loop node.

Note: You cannot set **Timer** or **Due Date** on the Activity Node which has been configured as Sequential Loop Node.

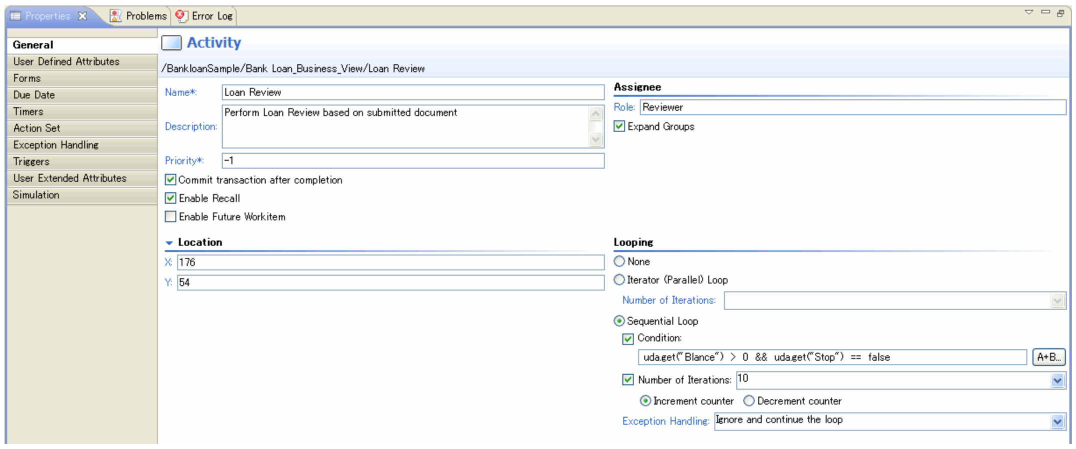
To configure a node as Sequential Loop node:

1. Select the node which you want to configure as Sequential Loop node.

   The **Sequential Loop** radio button in the **Looping** section is enabled in the **General** tab of the **Properties** view.

2. Select the **Sequential Loop** radio button.

Note: If you select a node and then select the **None** option, the node will behave as a normal node.



Activity

/BankloanSample/Bank Loan_Business_View/Loan Review

Name*: Loan Review

Description: Perform Loan Review based on submitted document

Priority*: -1

[✓] Commit transaction after completion

[✓] Enable Recall

[ ] Enable Future Workitem

Location

X: 176

Y: 54

Assignee

Role: Reviewer

[✓] Expand Groups

Looping

None

Iterator (Parallel) Loop

Number of Iterations:

[✓ Sequential Loop]

[✓] Condition:

uda.get("Blance") > 0 && uda.get("Stop") == false

[✓] Number of Iterations: 10

[✓ Increment counter] Decrement counter

Exception Handling: Ignore and continue the loop

Figure 99: Sequential Loop

Condition and **Number of Iterations** checkboxes are enabled.

3. Select the **Condition** checkbox or **Number of Iterations** checkbox or both.

   - If you select the **Condition** checkbox, 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. This expression must be a BOOLEAN expression.

Note: The **Condition** you set is evaluated before the generation of the node instance. The condition is generated and any node instance is not generated for the first time.

- If you select the **Number of Iterations** checkbox, you need to either enter a numerical value in the drop-down field or select the UDA of the INTEGER type from the drop-down list. The integral value of the numerical value or the selected UDA is considered to be the number of looped node instances. When you enter a numerical value or select a UDA in the drop-down list, **Increment counter** and **Decrement counter** radio buttons are enabled. Select an appropriate radio button.
  - If you select **Increment counter**, the number of iterations increases from 1 to the set number of iterations in **Number of Iterations** field.
  - If you select **Decrement counter**, the number of iterations start from the set value in the Number of Iterations field and reaches 1.

Note: If you do not set the **Condition** and **Number of Iterations**, the node behaves as a normal node.

4. Select the appropriate exception handling option from the **Exception Handling** drop-down list.
   - **None:** If you select this option, exception is handled by the node-level error handling setting. Refer section *Sequential Loop Node* in *Interstage BPM Developer's Guide* for more information.
   - **Ignore and continue the loop:** If you select this option, exception is ignored and loop is completed.
   - **Break the loop:** If you select this option, loop is terminated.

   The Activity Sequential Loop node is graphically displayed as below:

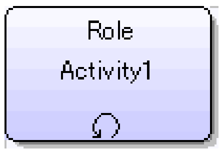


Figure 100: Activity Sequential Loop Node

The Subprocess Sequential Loop node is graphically displayed as below:

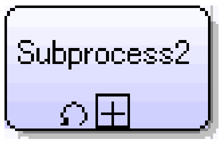


Figure 101: Subprocess Sequential Loop Node

6.14 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 is just like any other process definition. 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.
- You can add the following to a Compound Activity Node:
  1. **UDAs**: Refer *Specifying User Defined Attributes* on page 151 for more details.
  2. **Action Set**: Refer *Using Java Actions* on page 265 for more details.
  3. **Timers**: Refer *Using Due Dates and Timers* on page 169 for more details.
  4. **Due Date**: Refer *Using Due Dates and Timers* on page 169 for more details.
  5. **Forms**: Refer *Using Forms* on page 191 for more details.
  6. **Priority**: Refer *Setting Activity Level Priority* on page 137 for more details.

Refer *Creating Compound Activity Node* on page 166 to know how you can create a Compound Activity Node.

6.14.1 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.

Note: There should be at least one child Exit node inside the Compound Activity Node.

Note: The number of Exit nodes inside the Compound Activity Node and the number of outgoing arrows from the Compound Activity Node should be the same.

Note: 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:

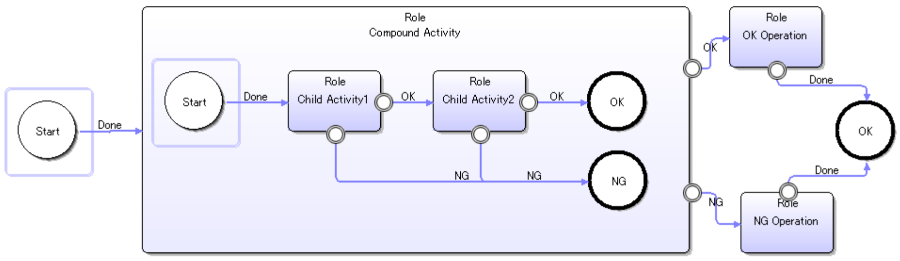


Figure 102: Compound Activity Node

6.15 Defining User Extended Attributes

You can define User Extended Attributes to add additional properties for process definitions, nodes (including Start, Exit, AND, and OR nodes), and arrows. The User Extended Attributes that you define, store the additional information defined by you, in the process definitions, nodes, and arrows.

To define User Extended Attributes:

1. Open the **Properties** view for process definitions, nodes, or arrows for which you want to add any additional properties.
2. Select the **User Extended Attributes** tab.
3. Click **Add**.

   Default attributes (for example, Attribute1, Attribute2, and so on) for the selected process definition, node, or arrow are added.

4. Click the appropriate row in the table to add/edit the name of the attribute in the **Name** column and add/edit value in the **Value** column.

Note: Do not use the following as User Extended Attribute name.

- Artifacts
- Associations
- Build
- ChildPlan
- ChildPlanId
- CommitJavaActionSet
- Coordinates
- CustomNodeType
- DataMapping
- DataMappings
- Description
- EnableFutureWorkItems
- EndPoint
- EpilogueJavaActionSet
- ExpandGroups
- ExposedField
- FormList
- FormsList
- IflowDataType
- InitJavaActionSet
- InTransaction
- IsWorkItemUDA
- IteratorCountUDA
- NodeSize
- NodeType
- Organization
- ParentVersion
- PrivateData
- ProcessDefinitionId
- ProcessOwnerRole
- ProcessOwnerRoleJavaActionSet
- ProcessTypeId
- PrologueJavaActionSet
- PrologueScript
- RecallDisabled
- RoleJavaActionSet
- SameSubPlanVersion
- Simulation
- StartPoint
- State

- subProcessDefinitionURI
- SWIM_LANES
- System
- TemplateIdentifier
- TimerDefSet
- Title
- TriggerDefSet
- VersionComment
- ViewerScript

5. (Optional) Select the appropriate row and click **Remove** to remove the selected User Extended Attribute.

   You can select two or more User Extended Attributes simultaneously by using **Ctrl** or **Shift** key and then remove all the selected User Extended Attributes by clicking the **Remove** button.

6.16 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 Delay Nodes, and for Trigger 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.

6.16.1 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.

  Interstage BPM determines business days and hours using a business calendar. Refer to section *Creating Your Own Business Calendars* on page 178 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 *Defining Timers* on page 171 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 *Defining Timers* on page 171 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.

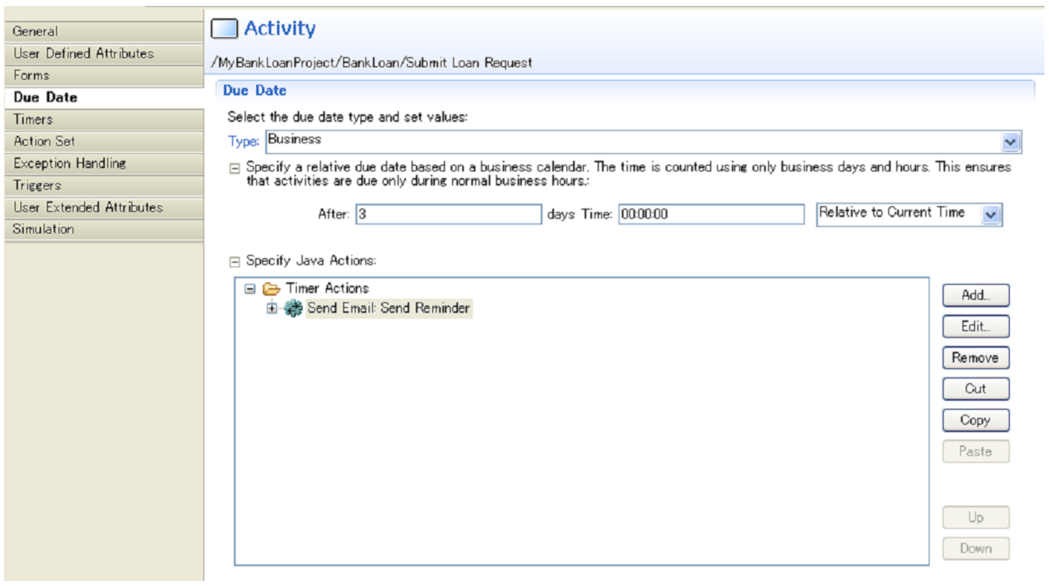
Note: If you set the expression in Advanced timer and if this expression is expressible as Business timer then this expression is displayed as the Business timer.

     For details on date codes that you can use in the expression, refer to section *Time and Day Codes for Advanced Due Dates and Timers* on page 176.

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 *Using Java Actions* on page 265.

The following dialog shows an example of a due date. If the activity is not completed after three business days, an email is sent.



/MyBankLoanProject/BankLoan/Submit Loan Request

Due Date

Select the due date type and set values:

Type: Business

Specify 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.:

After: 3 days Time: 00:00:00 Relative to Current Time

Specify Java Actions:

Timer Actions
Send Email: Send Reminder

Figure 103: Defining a Due Date

6.16.2 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, Compound Activity Nodes, and Trigger 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: 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**: Sets the absolute time when the timer will expire, for example, January 1, 2007, 00:00:00. 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. Java Script expression will automatically take the node activation date and time after the node is activated. 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. Java Script expression can be defined to return a value (for example, date) which will be taken as the reference for the time set in the days and hours fields (UDAs). 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. The timer will expire after the time set in the UDAs is elapsed starting from the date and time set in the Java Script expression. 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.

  Interstage BPM determines business days and hours using a business calendar. Refer to section *Creating Your Own Business Calendars* on page 178 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:

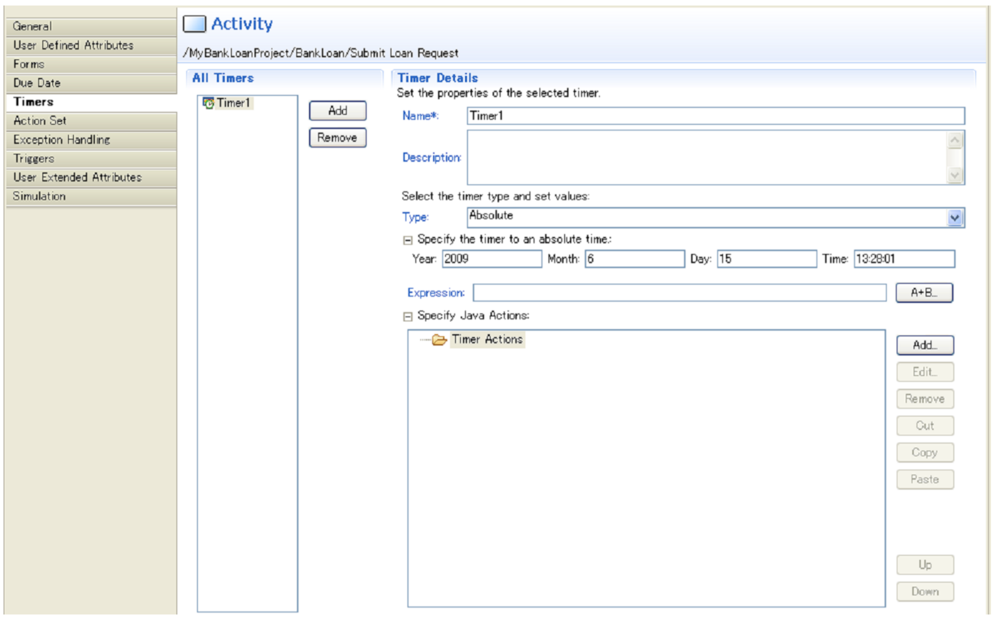
1. Do one of the following:
   - To define a timer for a process definition, click the empty space in the Process Definition editor to display the Properties view for the process definition.
   - To define a timer for a node, select the node in the Process Definition editor to display the Properties view for the node.

Note: 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 customise 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, make sure 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\")` 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.



/MyBankLoanProject/BankLoan/Submit Loan Request

Name*: Timer1

Description:

Type: Absolute

Year: 2009 Month: 6 Day: 15 Time: 13:28:01

Expression:

Figure 104: Defining an Absolute Timer

The following figure shows the **Interstage BPM Expression Builder**.

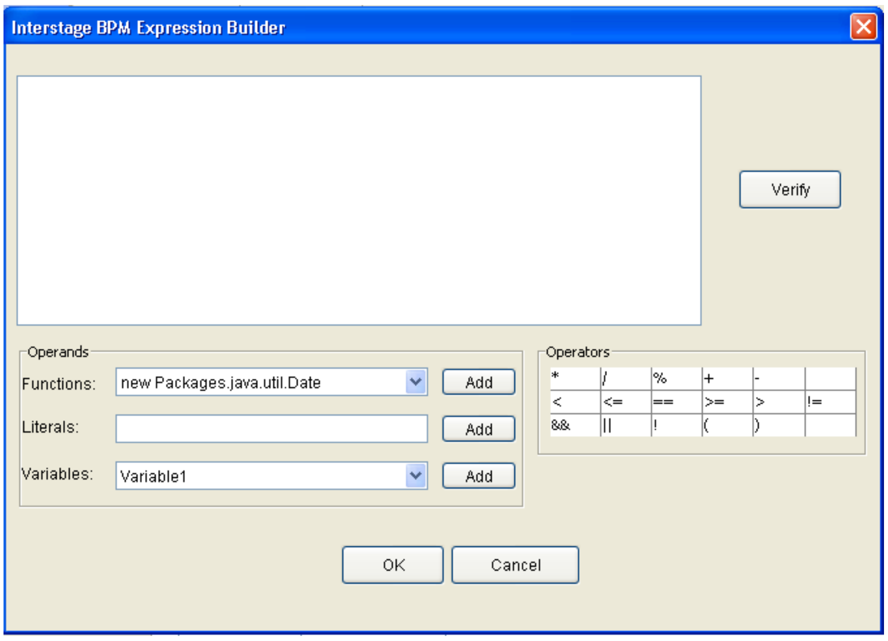


Figure 105: Building Java Script Expression

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.

- 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. These are the UDA values for the Calendar timer.
  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: 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: 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 and 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.
  2. 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 details on date codes that you can use in the expression, refer to section *Time and Day Codes for Advanced Due Dates and Timers* on page 176.
  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.

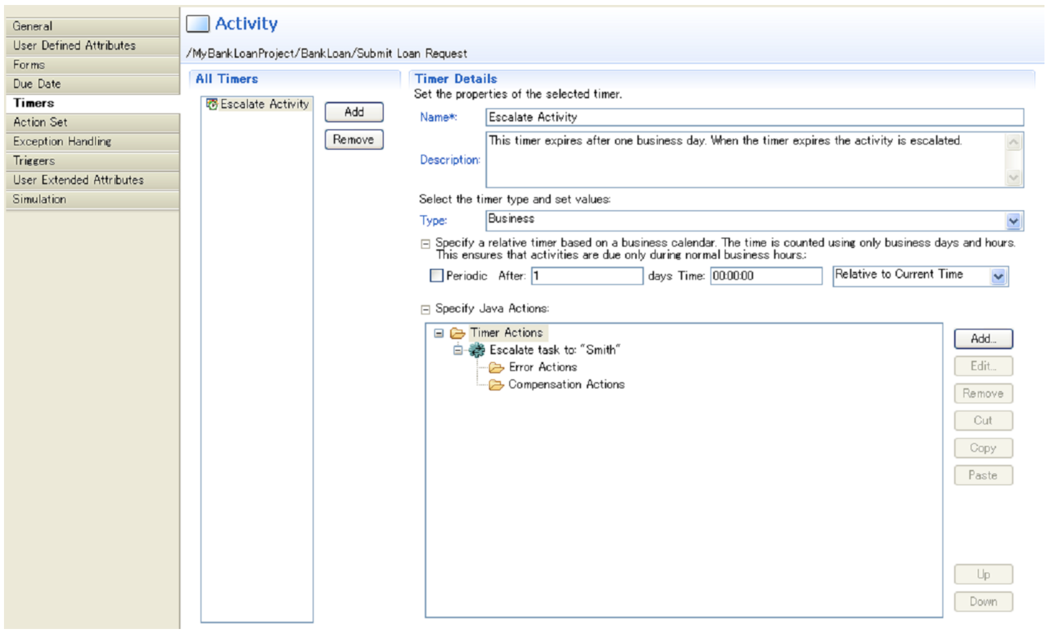
Note: If you set the expression in Advanced timer and if this expression is expressible as Business timer then this expression is displayed as the Business timer.

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 *Using Java Actions* on page 265.

The following figure shows a timer that expires after one business day. When the timer expires, the activity is escalated.



Activity

/MyBankLoanProject/BankLoan/Submit Loan Request

All Timers

Escalate Activity

Timer Details

Set the properties of the selected timer.

Name*: Escalate Activity

Description: This timer expires after one 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: 1 days Time: 00:00:00 Relative to Current Time

Specify Java Actions:

- Timer Actions
  - Escalate task to: "Smith"
    - Error Actions
    - Compensation Actions

Figure 106: Defining a Timer

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.

You can check which UDAs belong to a timer by clicking **Advanced** on the **Timers** tab.

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: .

6.16.3 Time and Day Codes for Advanced Due Dates and Timers

Advanced due dates and timers are defined using expressions. In these expressions, you can use codes to specify the time and/or day. This section describes the time and day codes.

Time Codes

Code	Meaning	Example
`AT`	Sets the absolute time of the day.	`AT(16:30:00)`: 4:30pm on that day.

Code	Meaning	Example
`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.
`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 business day. `BD(-0)`: Same day if it is a business day, else the previous business 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.
`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.

Code	Meaning	Example
`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.

6.16.4 Creating Your Own Business Calendars

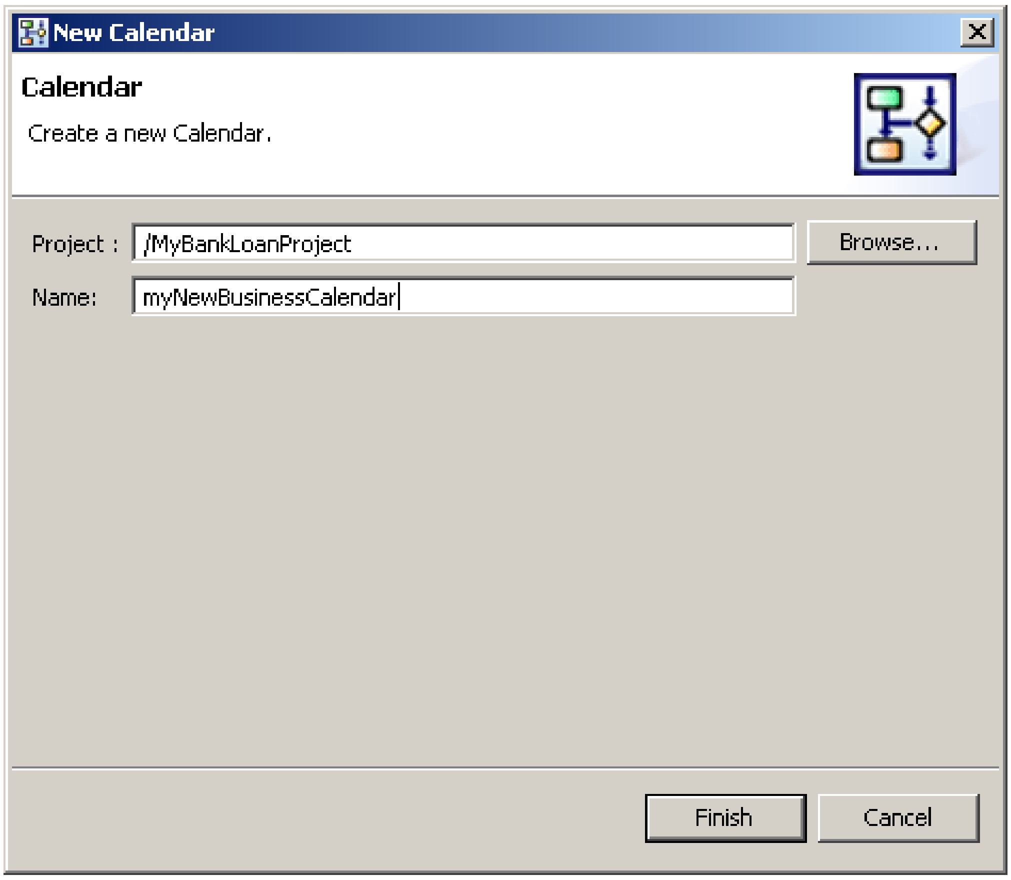
When defining due dates and timers, you can use a business calendar instead of a regular calendar. A business calendar defines the business hours and days of an organization. Using a business calendar ensures that timers expire during business hours only. The same applies for due dates; business calendars ensure that activities are due during business hours only.

Interstage BPM 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.


New Calendar

Calendar

Create a new Calendar.

Project : /MyBankLoanProject

Name: myNewBusinessCalendar

Figure 107: Displaying the New Calendar dialog

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:

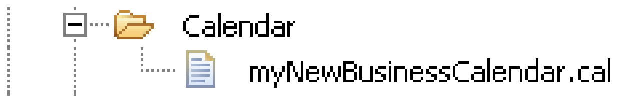


Figure 108: Location of a new calendar file

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\InterstageBPM_studio\ibpm\Data\calendar` directory. The file looks as follows:

```
EVERYDAY=8:00,18:00;
   SAT=;
   SUN=;
   2003/01/01=;
   2004/01/01=;
   2005/01/01=;
   2006/01/01=;
   2007/01/01=;
   2008/01/01=;
   2009/01/01=;
   2010/01/01=;
   CALENDAR_END=2010/12/31;
   CALENDAR_BEGIN=2003/01/01;
   TIMEZONE=-8:00;
```

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 *Interstage Business Process Manager Server Administration Guide*.

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.

6.16.5 Assigning Business Calendars to Process Definitions

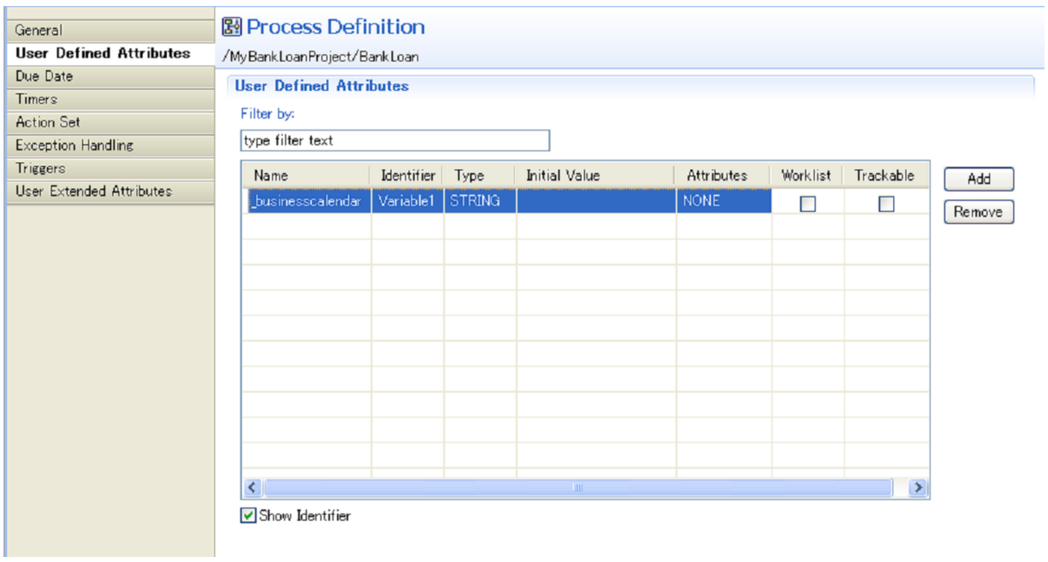
Prerequisite: You have created your own business calendar.

You can assign different business calendars for every process definition.

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:



Name	Identifier	Type	Initial Value	Attributes	Worklist	Trackable
_businesscalendar	Variable1	STRING		NONE	☐	☐

Figure 109: Assigning a Business Calendar

6.16.6 Assigning Business Calendars to Due Dates or Timers

Prerequisite: You have created your own business calendar.

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.

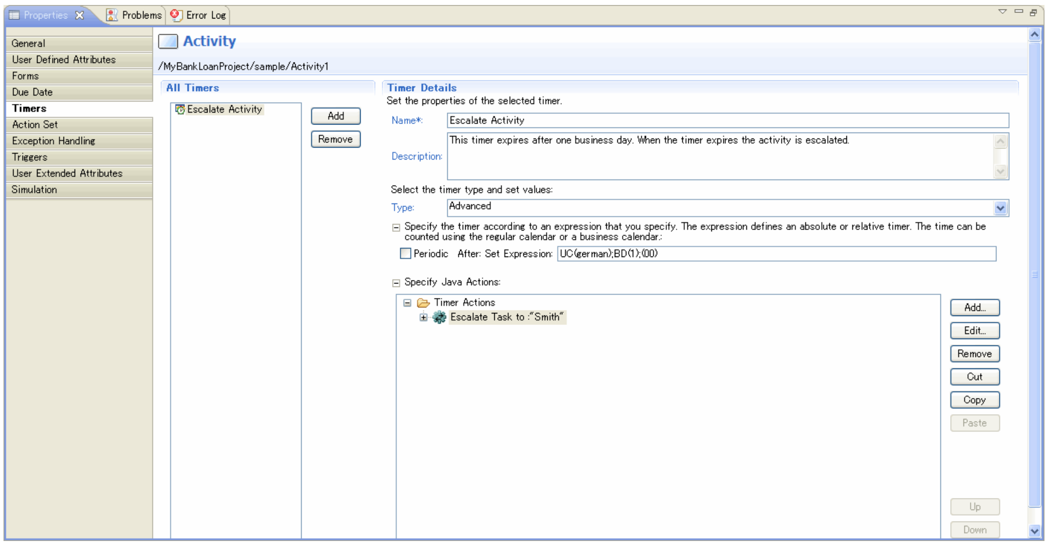
   Refer to sections *Defining Due Dates* on page 169 or *Defining Timers* on page 171 for instructions.

2. In the **Set Expression** field, enter the following expression:

   ```
   UC(<business calendar>);
   ```

   For `<business calendar>`, specify the name of your business calendar without the `.cal` extension.

In the following example, a business calendar called `german.cal` is assigned to a timer:



Activity

/MyBankLoanProject/sample/Activity1

All Timers

Escalate Activity

Timer Details

Set the properties of the selected timer.

Name*: Escalate Activity

Description: This timer expires after one business day. When the timer expires the activity is escalated.

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(german);BD(1);(00)

Specify Java Actions:

Timer Actions

Escalate Task to :"Smith"

Figure 110: Assigning a Business Calendar to a Timer

6.17 Defining 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.

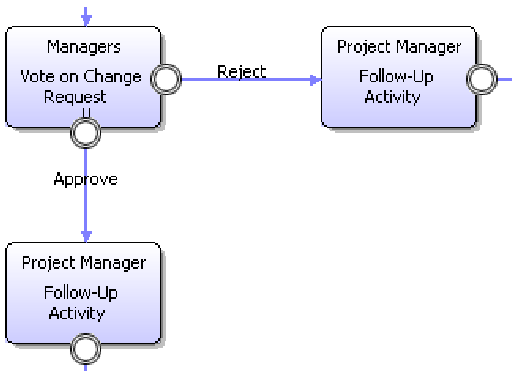


Figure 111: Voting on a Change Request

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 make 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:

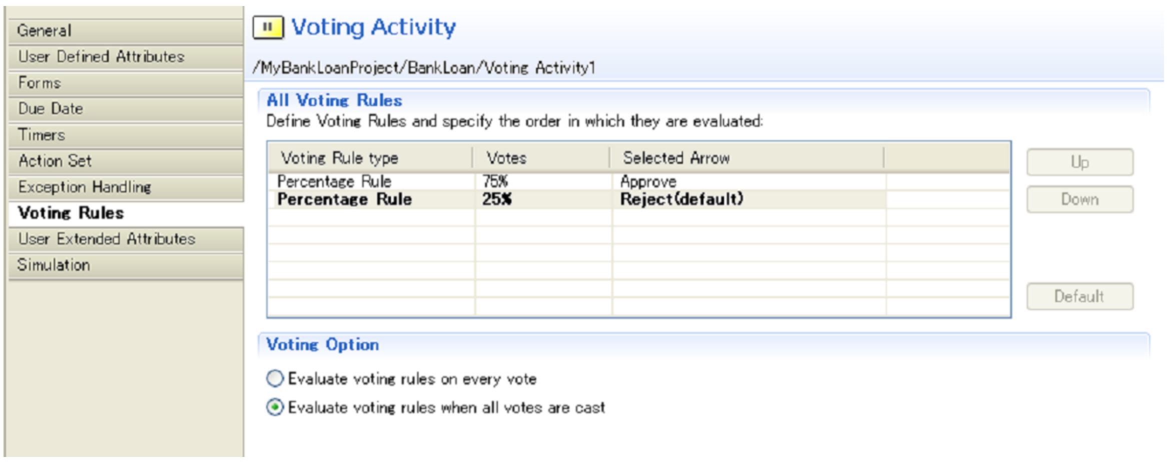
1. Select the Voting Activity Node to display the Properties view for it.
2. Select the **Voting Rules** tab.

   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 make sure 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.



/MyBankLoanProject/BankLoan/Voting Activity1

All Voting Rules

Define Voting Rules and specify the order in which they are evaluated:

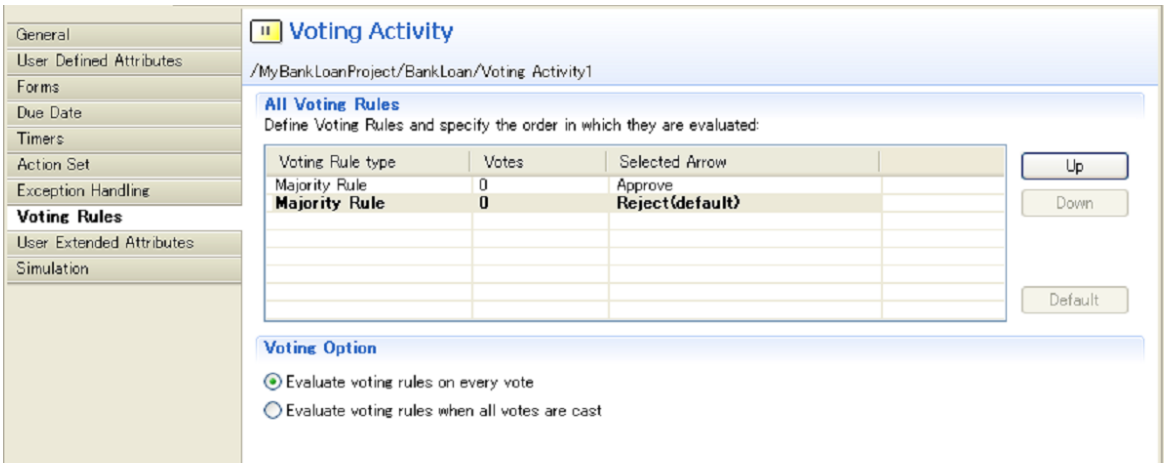
Voting Rule type	Votes	Selected Arrow
Percentage Rule	75%	Approve
Percentage Rule	**25%**	**Reject(default)**

Voting Option

( ) Evaluate voting rules on every vote
[✓ Evaluate voting rules when all votes are cast]

Figure 112: Percentage Rules

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.



/MyBankLoanProject/BankLoan/Voting Activity1

All Voting Rules

Define Voting Rules and specify the order in which they are evaluated:

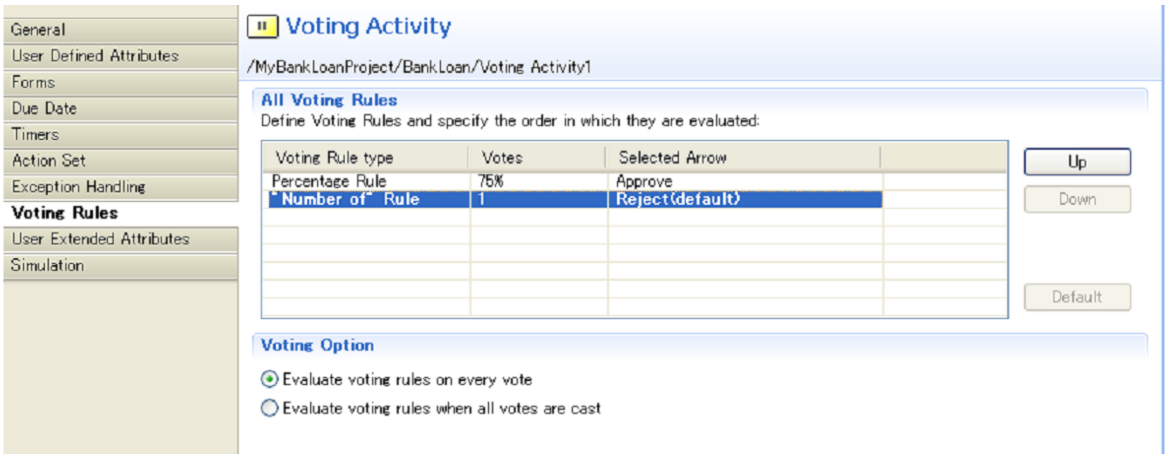
Voting Rule type	Votes	Selected Arrow
Majority Rule	0	Approve
Majority Rule	**0**	**Reject(default)**

Voting Option

[✓ Evaluate voting rules on every vote]
( ) Evaluate voting rules when all votes are cast

Figure 113: Majority Rules

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.



Voting Activity

/MyBankLoanProject/BankLoan/Voting Activity1

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
"Number of" Rule	1	Reject(default)

Voting Option

(●) Evaluate voting rules on every vote

( ) Evaluate voting rules when all votes are cast

Figure 114: Number of Rule

6.18 Defining 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.

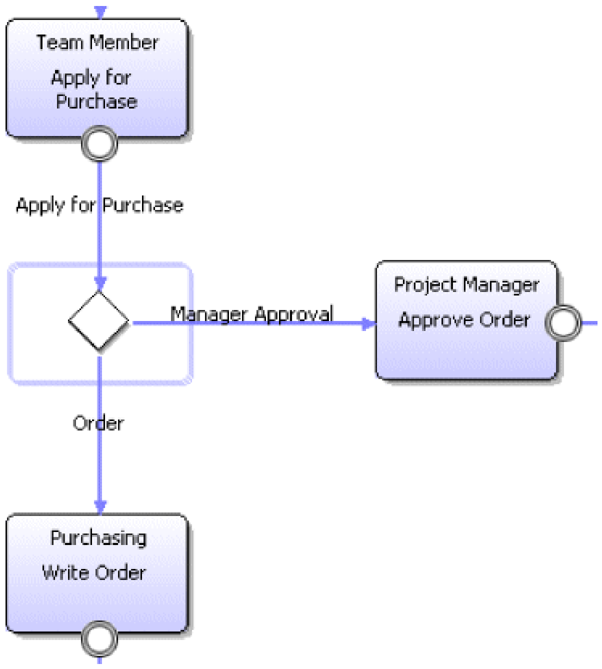


Figure 115: Using Conditional Nodes

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 or CUSTOM, the **Properties** view displays an additional **XPath** field.

Note: If you select a non-XML or non-CUSTOM UDA type, the XPath field will be disabled.

4. If you select a UDA data type as XML, click in the field in the **XPath** field. 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. **XPath** field is active only if you have selected a UDA of type XML in **UDA name** field.
5. 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 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.

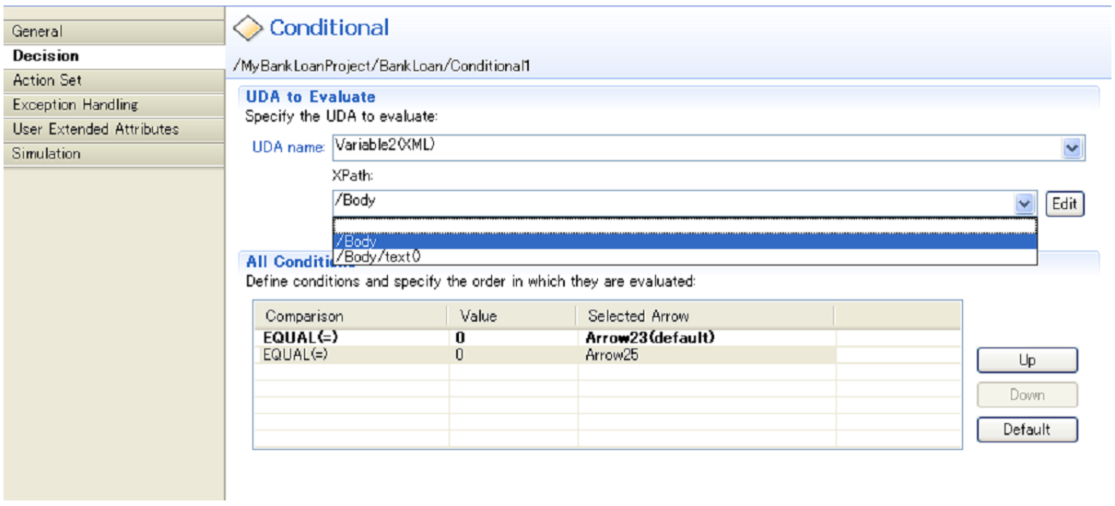
Note: The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note: The XPath expressions related to the selected XML or CUSTOM 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 XML 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: If a Custom UDA is selected, XPath expressions specifying only leaf nodes (such as text or attribute nodes) will be shown in the **XPath** drop-down list.

Note: **XPath Editor** only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

6. Select the UDA from the **UDA name** drop-down list. The names listed in this list are based on the UDA data type chosen by you.
7. For each arrow that originates from the Conditional Node, specify the criteria to compare with the value of the UDA.



/MyBankLoanProject/BankLoan/Conditional1

UDA to Evaluate
Specify the UDA to evaluate:
UDA name: Variable2(XML)
XPath: /Body
/Body
/Body/text()

All Conditions
Define conditions and specify the order in which they are evaluated:

Comparison	Value	Selected Arrow
EQUAL(=)	0	Arrow23(default)
EQUAL(=)	0	Arrow25

Figure 116: Defining Conditions

8. 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.
9. 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.

6.19 Defining Complex Conditions

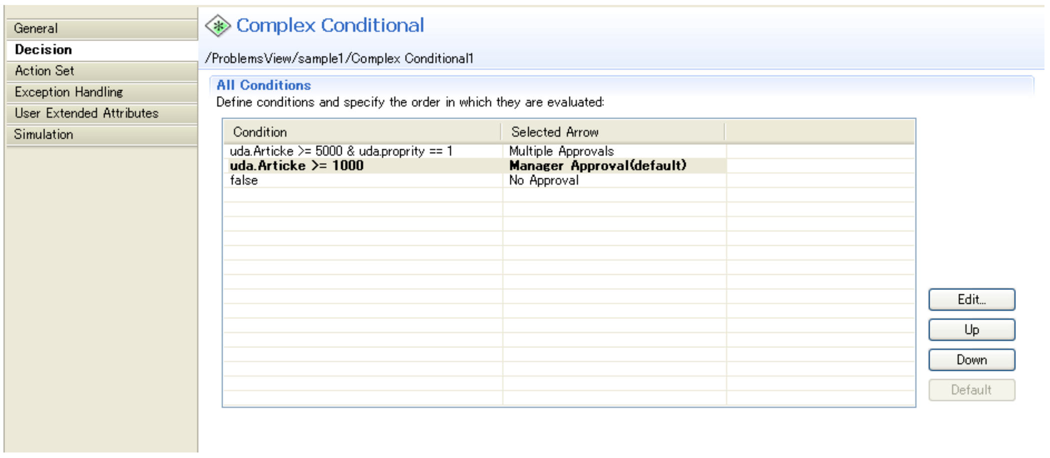
Prerequisites:

- 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.



/ProblemsView/sample1/Complex Conditional1

All Conditions

Define conditions and specify the order in which they are evaluated:

Condition	Selected Arrow
uda.Articke >= 5000 & uda.proprity == 1	Multiple Approvals
uda.Articke >= 1000	**Manager Approval(default)**
false	No Approval

Figure 117: Defining Complex Conditions

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 *Defining JavaScript Expressions* on page 371.

   If the JavaScript expression for an arrow evaluates to true, the arrow is chosen.

4. If you want to define a default conditon, select the condition in the **All Conditons** 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.

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.

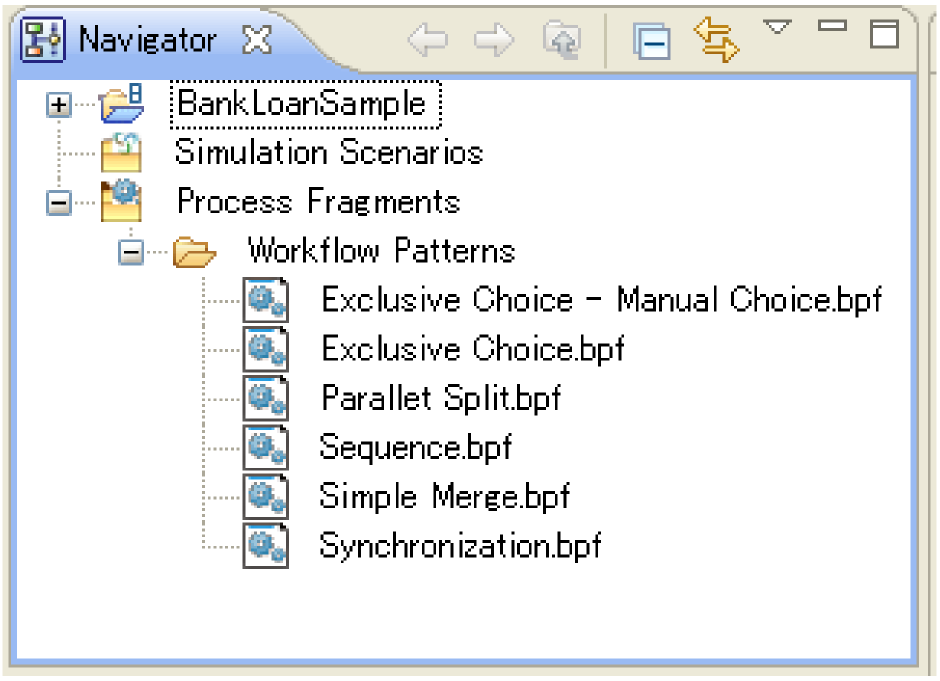


Figure 118: Process Fragments Navigator View

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

- *Creating Process Fragments* on page 189
- *Adding Process Fragments to Process Definitions* on page 190

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.

Note: To copy arrows, the source and target nodes of the arrow must also be selected.

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: Make sure 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.

8 Using Forms

Most business processes involve decisions and actions that are based on information. Users can provide and access information through forms.

Forms are structured, field-based HTML files. They serve as an interface for data exchange between Interstage BPM and structured data repositories.

Forms can be used:

- For displaying or reporting data stored in User Defined Attributes (UDAs)
- For enabling users to add or modify data stored in UDAs.

Forms can be associated with Start Nodes, Activity Nodes, Voting Activity Nodes, and Compound Activity Nodes. Depending on your needs, you may associate a single form with multiple nodes. Vice versa, you may associate one or multiple forms with a node. If you associate multiple forms with a node, the Interstage BPM Console displays them as the tab of each related form. You can click the tab to select the form.

You can use the following two types of QuickForms:

- **QuickForm (to display or update the UDA Value)**:

  This is the Form to display or update the UDA value. You can create this form graphically by using Ajax Page Editor. In this Editor, you can freely lay out Item (TextInput, Select, RadioButton, etc.). Each Item is related to UDA. This file extension is **.jsp**.

- **QuickForm (for compatibility)**:

  This QuickForm is a structured, field-based HTML file. You can create these QuickForms using the Interstage BPM Studio as follows: Based on a few settings, Interstage BPM Studio generates an HTML file that contains fields for all UDAs that are to appear on the form. The QuickForm is generated with attributes that make it work with Java Server Pages (JSPs).

  Once you have generated the form, you can customize it according to your own particular needs as long as you do not edit the HTML components needed by Interstage BPM for data and process information display. You can modify the layout using any HTML editor or by modifying the HTML code directly. This file extension is **.qf**.

Both QuickForms enable users to add or modify data stored in UDAs.

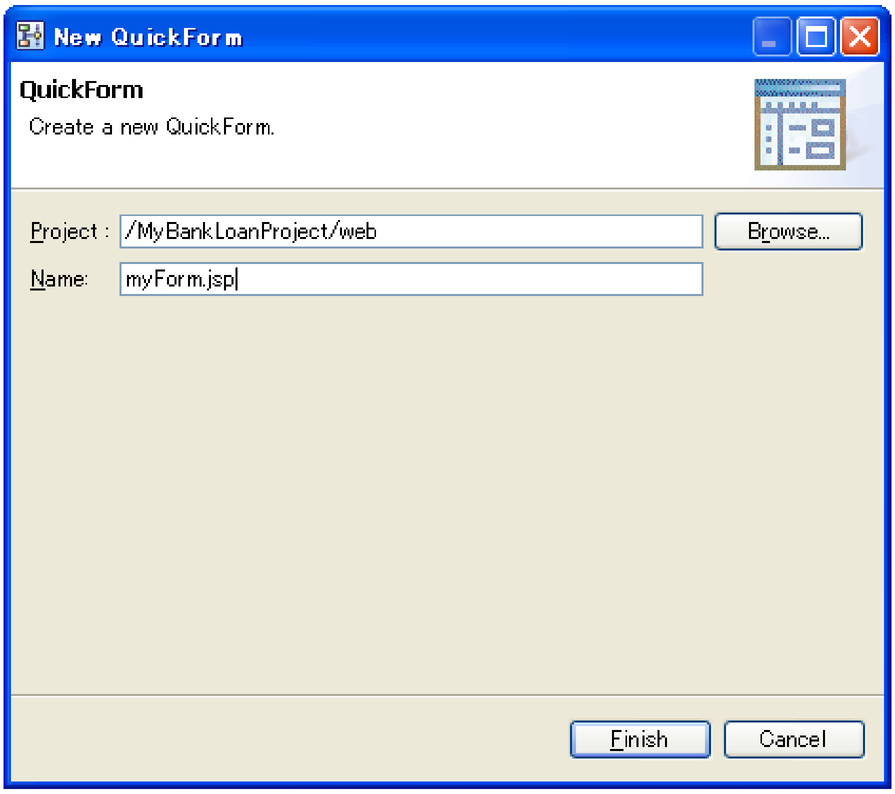
8.1 Creating QuickForms

You create QuickForms using the Ajax Page Editor.

To create a QuickForm:

1. There are two ways to open the Form editor:
   a) In the Navigator view, select the Workflow Application project for which you want to create a QuickForm, and select **New > QuickForm** from the **File** menu.
   b) Open the process definition of your Workflow Application project for which you want to create a QuickForm in the Process Definition editor, click the relevant Start Node, Activity Node, Voting Activity Node, or Compound Activity Node, and select **QuickForm > New** from the popup menu.

The following window will be opened:



Project : /MyBankLoanProject/web

Name: myForm.jsp

Figure 119: Creating a QuickForm

2. **web** folder name of the project is displayed in the **Project** field.
3. Type a name for your QuickForm in the **Name** field, and click **Finish**.

   The new QuickForm (**.jsp**) file is automatically stored in the selected folder.

Note: The following special characters are not supported in the names of the subfolders of the forms as well as in the .jsp form names: **$**, **&**, **<**, **>**, **?**, **;**, **#**, **:**, **=**, **,**, **"**, **'**, **~**, **+**, **%**

Note: The name of the form or the name of the subfolder containing forms created inside the `<Application Project>\web` folder of an application should not start with the substring **web**.

8.2 Ajax Page Editor

You can use the Ajax Page Editor to create QuickForm.

Ajax Page Editor has the following features:

- While confirming the runtime image, you can create a QuickForm.
  - The Screen layout and widget size, and position are displayed by runtime image.
  - The screen is displayed according to the setting of the attribute and CSS.
  - UI Widgets and HTML tag items are displayed by runtime image.

- You can edit a QuickForm by using the text mode. This edit operation is automatically reflected with the visual screen.

8.2.1 Overview

It explains the outline of the view of Ajax Page Editor as follows:

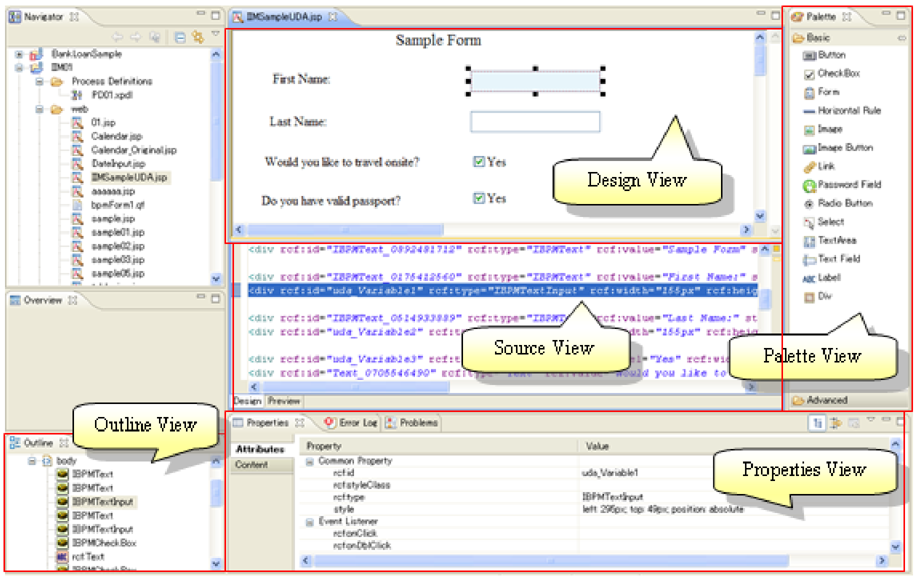


Figure 120: Views of Ajax Page Editor

***Design View* on page 195**

It is a view to edit the QuickForm using UI widgets with WYSIWYG. This view has the following features:

- You can edit a QuickForm using the absolute coordinate.
- You can easily define a UI widget.
    - Drop a widget from the **Palette** view to create this widget.
    - Move and resize a widget by mouse control.
- You can use the clipboard to copy a widget.

***Source View* on page 198**

It is a view that edits the source code of a QuickForm using UI widgets. The following functions are provided:

- The edit operation is automatically reflected in the **Design** view, the **Property** view, and the **Outline** view.

***Palette View* on page 199**

It is a view to display the HTML tags and UI widgets in the palette, and to arrange it on the **Design** view.

***Outline View* on page 202**

It is a view that displays the structure of the tag in the **Source** view.

***Properties View* on page 202**

It is a view that displays the following properties:

- Selected UI widget in the **Design** view
- Caret on a Tag or UI widget in the **Source** view
- Selected Tag or UI widget in the **Outline** view

This view has the following two tabs:

- **Attribute** tab: Displays the property of the selected Tag or UI widget
- **Content** tab: Displays the contents of the selected Tag or UI widget

Preview

You click the **Preview** tab in the Ajax Page Editor to display this view. In the preview, you confirm UI widget by runtime image using IE component of Internet Explorer.

QuickForm that can be edited

You can edit the following QuickForms only.

- QuickForms created by the rule of XHTML 1.0
- QuickForms in which the following DOCTYPE exists

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- This uses the following files:
  - Ajax Framework Config file (`rcf_config.js`)
  - AjaxFramework initial processing (`rcf.js`)

    When there is a QuickForm under the **web** folder in an Application Project, these files are defined below:

    ```
    <script type="text/javascript" src="../rcf_config.js"></script>
    <script type="text/javascript" src="../acf/file/rcf/rcf.js"></script>
    ```

    When there is a QuickForm under the subfolder of **web** folder (web/sub1) in an Application Project, these files are defined below:

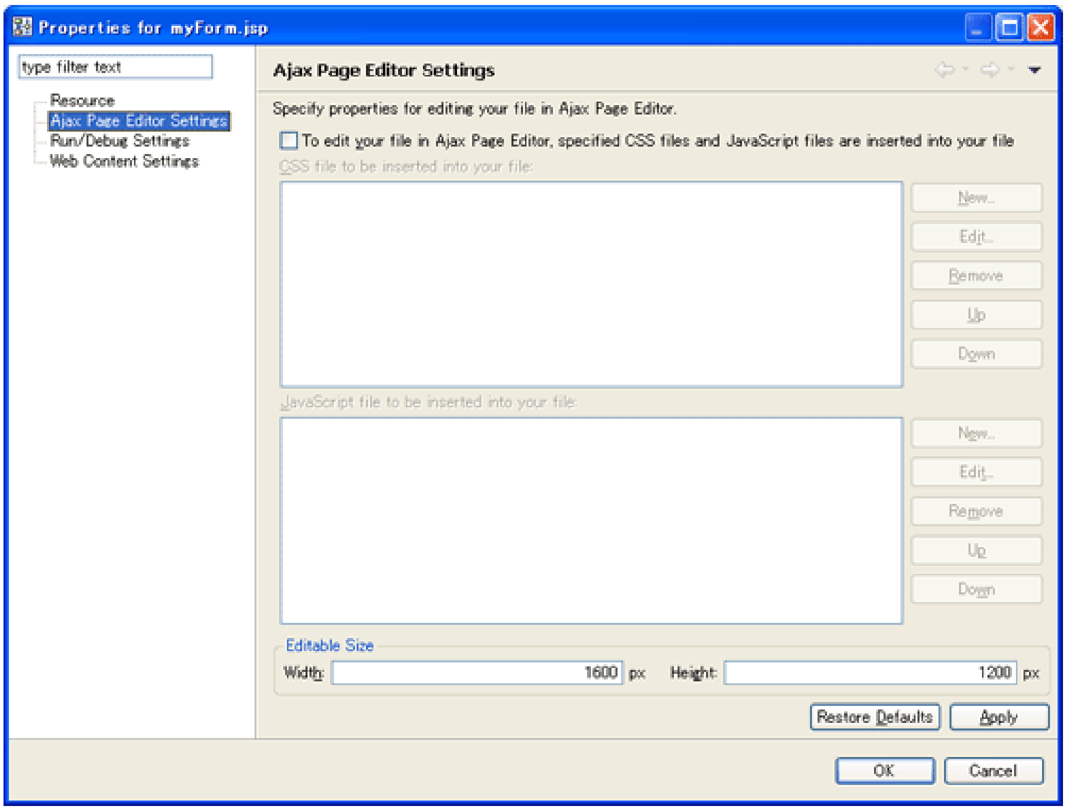
    ```
    <script type="text/javascript" src="../../rcf_config.js"></script>
    <script type="text/javascript" src="../../acf/file/rcf/rcf.js"></script>
    ```

Note: When you open a QuickForm, Studio updates these paths automatically. But if you save as a QuickForm to another folder, you have to update these paths by this rule.

Note: When you edit a QuickForm, only **< body >** tab is basically edited. You need not change other tabs as long as there is no special explanation.

Ajax Page Editor Settings

When you select a QuickForm and click the **Properties** command in the **Navigator** view; **Properties for <form name>** dialog is displayed.



Properties for myForm.jsp

type filter text

Resource
Ajax Page Editor Settings
Run/Debug Settings
Web Content Settings

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

Figure 121: Ajax Page Editor Settings

In **Ajax Page Editor Settings** tab, you do not have to click the following item:

- **To edit your files in Ajax Page Editor, specified CSS files and JavaScript files are inserted into your file** checkbox.

You can define the following items:

- **Editable Size:** The size (width and height) that can be edited in Ajax Page Editor is specified by the pixel for the selected file. UI widgets can be edited within the specified size range in the **Design** view of Ajax Page Editor.
- **Restore Defaults:** It returns it to the default setting.

8.2.2 Design View

The **Design** view displays the same image as the runtime of the HTML tag and the UI widgets. You can move, resize, and delete it. You can drop item in the **Palette** view to add it.

Note: When an error occurs in the source code, you cannot correct it using the **Design** view. In this case, you must fix this error using the **Source** view.

The **Design** view has the following popup menu commands.

Command	Description
Refresh	Refreshes the **Design** view
Cut	Cuts the selected widget to copy it to the clipboard
Copy	Copies the selected widget to the clipboard
Paste	Pastes cut or copied widget. This widget is cut or copied in the **Design** View.
Delete	Deletes the selected widget
Align	Aligns the widget
Define model binding	You cannot use this command.
Define event processing	You cannot use this command.
Define additional functional widget binding	You cannot use this command.
Reflect Automatically While Editing Source View	Sets whether to update the **Design** view automatically while you edit it using the **Source** view
Properties	Show the **Properties** view

Note: When you do not select the **Reflect Automatically While Editing Source View** command and you edit a QuickForm using the **Source** view, the **Design** view does not synchronize with the **Source** view. At this point, the **Design** view displays a gray screen. This **Design** view is not able to display anything and does not allow you to edit it. This **Design** view is refreshed when you do any of the following actions:

- Focus the **Design** view.
- Execute **Edit > Refresh**.
- Click any of the following buttons in the **Toolbar**:

When you click the **Preview** tab under such a condition, the latest screen is displayed.

How to Edit Widgets

There are the following two types of operations in the **Design** view:

- Common operations to widgets
- Peculiar operations to widgets

Common Operations on Widgets

Common operations comprise the following operations:

- **Locate a widget:**
  1. Select a widget in the **Palette** view.

2. Click the position where you want to locate this widget.

   When this widget is located, the ID's (`rcf:id` property) value is automatically set. Press the **Esc** key to cancel the selected widget.

   For details on the Palette view, refer to section *Palette View* on page 199. To locate a widget in a Container widget, refer *Peculiar Operations on Container Widget* on page 198.

- **Select widget:** You can select the widget using any one of the following:
  - Click the widget on the area that can be selected.
  - Click the widget on the **Outline** view.
  - Set a caret tag on the **Source** view.
- **Move widget:** You can move the widget using any one of the following:
  - Select the widget and drag.
  - Select the widget and press cursor keys.
- **Cut widget:** You can cut the widget using any one of the following:
  - Select the widget and execute the **Cut** command of the popup menu.
  - Select the widget and press Ctrl + X.
- **Copy widget:** You can copy the widget using any one of the following:
  - Select the widget and execute **Copy** command of the popup menu.
  - Select the widget and press Ctrl + C.
- **Paste widget:** You can paste the widget using any one of the following:
  - Execute **Paste** command of the popup menu.
  - Press Ctrl + V.

    The pasted widget is located on the lower right of the widget that has been cut out or copied. If a widget is pasted in which `rcf:id` is defined, a new value is set to `rcf:id` of the pasted widget.
- **Delete widget:** You can delete the widget using any one of the following:
  - Select the widget to be deleted and execute **Delete** command of the popup menu.
  - Select the widget and press Ctrl + Delete.

    When you delete the widget which have child widgets, the child widgets are also deleted.
- **Resize widget:** You can resize the widget using any one of the following:
  - Drag the markers of the widget to resize them.

    When you resize a Container widget, the position of its child element is not updated by this operation.

    While resizing the Container widget by dragging its markers, if its edges overlap other widgets then such overlapped widgets are not the child widgets of this Container widget.
- **Undo:** You can undo any action on the widgets using any one of the following:
  - Execute the **Undo** command of **Edit** menu.
  - Press Ctrl + Z.

    You can undo the following commands:
    - Locate widget
    - Move widget

    - Cut widget
    - Paste widget
    - Delete widget
    - Resize widget
    - Align widget(s)

- **Redo:** You can redo any action on the widgets using any one of the following:
  - Execute the **Redo** command of **Edit** menu.
  - Press Ctrl + Y.
- **Align widget(s):**
  - Select the widgets and execute the **Align** command of the popup menu.

    You can align widgets as under:
    - Align Left
    - Align Center
    - Align Right
    - Align Top
    - Align Middle
    - Align Bottom

Peculiar Operations on Container Widget

The widget that contains another widget on which you can perform operations in the **Design** view as a child element, is called Container widget. There are the following widgets in the Container widget:

- Container widget of UI widget (FragmentContainer is excluded)
- HTML form tag

The peculiar operations explain the following operations:

- Move a Container widget

  If you move a Container widget, its child widget(s) are also moved.
- Add or remove a child element to a Container widget.

  If you move a widget to a Container widget, it is added to this Container widget as a child element.

Peculiar Operations on Calendar Widget

A PopupCalendar is used by a CalendarButton together. If you define a CalendarButton, a popupCalendar is automatically defined.

8.2.3 Source View

In Interstage Business Process Manager, you use the **Design** view for basic operations. You use the **Source** view for additional operations. The **Source** view has the following popup menu commands.

Command	Description
Undo	Undoes the last change

Command	Description
Revert file	Reverts this file
Cut	Cuts the selected data to copy it to the clipboard
Copy	Copies the selected data to the clipboard
Paste	Pastes cut or copied data
Format	Formats the editor content
Format Active Elements	Format the active elements
Properties	Show the Properties View
Preferences	You cannot use this command.

8.2.4 Palette View

The **Palette** view is used to add a widget to the **Design** view. The **Palette** view has the following two categories:

- Basic
- Advanced

When you use Interstage BPM Studio as Eclipse Plug-In, the **Palette** view has the following three categories:

- **XHTML1.0:** This category is the same as Basic category.
- **Ajax:** You cannot use this category.
- **Advanced:** This category is the same as Advanced category.

Basic Category

It explains the Basic category (In Eclipse Plugin, this category is shown as XHTML1.0).

This category has the following widgets:

Display Name	HTML Tag	Description
Button	Input	The input element that provides with "type=submit" shows the input option and the button to inform the user agent that the form is executed.
CheckBox	Input	The input element in which type= checkbox is defined, is used as Check Box. User can select some items in one set (group).
Form	Form	The form element contains the sequence of the input factor in addition to the document that assembles the element.
HorizontalRule	hr	The hr element is a divider between sections of the text. It is a full ruled line or equal graphic usually.
Image	img	Refer to the image or the icon by the hyperlink for the img element.

Display Name	HTML Tag	Description
Image Button	input	The input element that provides with "type=image" specifies the displayed image resource, and specifies X and Y coordinates of the pixel selected from the image.
Link	a	a element enables the user to navigate the content of the document.
Password Field	input	The input element in which type=password is defined, is used as a Password field.
Radio Button	input	The input element in which type=radio is defined, is used as a Radio button. User can select one item in one set (group).
Select	select	The select element controls the form field to the list that enumerates the value.
TextArea	textarea	The TextArea Field element provides multi-line text input field.
Text Field	input	The Text Field element provides single-line text input field.
Label	label	The label element shows the label.
Div	div	The div element shows the block element.

Advanced Category

This category has the following widgets:

Display Name	Description
Text	The widget to display text. This corresponds to `<span>text</span>` in HTML.
TextInput	The widget to input and edit a single line of text. This corresponds to `<input type="text" value="value">` and `<input type="password" value="value">` in HTML.
CheckBox	The widget to display an "on/off" (selected or cleared) check box. This corresponds to `<input type="checkbox" value="value ">` in HTML.
RadioButton	The widget to display a radio button. This corresponds to `<input type="radio" value="value">` in HTML.
Button	You cannot use this widget.
TextArea	The widget to input and edit one or more lines of text. This corresponds to `<textarea>` in HTML.

Display Name	**Description**
Select	The widget to display a list from which single or multiple items can be selected. This corresponds to `<select> <option>` in HTML.
ComboBox	The widget to display a combo box (sometimes called dropdown box), which consists of an input field and a list of items from which a single item can be selected. It is also possible to enter text directly in the input field.
DateInput	A type of TextInput widget to input and edit the date and time.
NumberInput	A type of TextInput widget to input and edit numerical values.
MaskedTextInput	You cannot use this widget.
MaskedDateInput	You cannot use this widget.
SelectList	The widget to display a list from which single or multiple items can be selected.
CheckList	The widget to display a list with check box items.
ViewContainer	The general container widget. This widget can contain HTML elements and the whole can be handled as one widget. ViewContainer itself does not have a view component.
Panel	The widget to display a container with a title bar. This consists of title part and body part. This widget can contain HTML elements for the body part.
ViewStack	The widget to switch between containers in the same position.
TabPanel	The widget to display containers on tabbed pages. Users can switch between pages by clicking the tabs.
FragmentContainer	The container widget to load and display contents from an external source after the page has been displayed.
TableView	The widget to display two-dimensional data in table format. Data can be sorted in ascending or descending order.
TableEdit	The widget to display two-dimensional data in table format and edit the data.
DataGrid	You cannot use this widget.
Calendar	The widget to display a calendar and select a date
CalendarButton	If you define DateInput widget the CalendarButton is defined together. This widget is used for displaying the PopupCalendar.
TreeView	You cannot use this widget.
ScrapingView	You cannot use this widget.

Popup Menu Commands

The **Palette** view has the following popup menu commands.

Command	**Description**
Hide (*1)	The category is non-displayed.
Layout	Changes the layout of palette
Use Large Icon	Displays palette items using the large icon
Customize	Displays the **Customize Palette** dialog and customizes the category.
Settings	Displays the **Palette Settings** dialog and sets the display format of the item.
Pinned (*1)	It opens without closing the category when the pin is detained.

(*1) Hide and the Pinned command are displayed only when right-clicking by the category.

8.2.5 Outline View

In the **Outline** view, widgets list on the Ajax Page Editor are displayed as a tree view. Moreover, you can select a widget in this list. The **Outline** view has the following popup menu commands:

Command	**Description**
Delete	Deletes the selected node
Define model binding	Defines the model binding. You cannot use this command.
Define event processing	Defines the event processing. You cannot use this command.
Define additional functional widget binding	Defines the binding a widget and an additional functional widget. You cannot t use this command.
Properties	Displays the selected node's property

8.2.6 Properties View

When you select a widget or a tag in **Design** view, the **Source** view and the **Outline** view, the properties of the selected widget or tag are displayed in the **Properties** view. The **Properties** view has the following two tabs:

- **Attributes** Tab: The properties of the selected widget or tag are displayed.
- **Content** Tab: The content of the selected widget or tag is displayed.

Attributes Tag

You can edit the properties of the selected widget or tag in this tab. This tab has the following two columns:

Column	Description
Property	The property has the following categories: **Common Property:** Displays the common property of a UI widget **Control Property:** Displays the control property of a UI widget **Style Property:** Displays the style property **Event Listener:** Displays the event listener. You cannot use this property **Validation Property:** Displays the validation property
Value	Displays the value of this property

Note: If you delete the value in the **Properties** view, the property itself is not deleted. The value of the property is set to `""`.

If you delete the value of the property (the default value of this property is not `""`), the error occurs in the **Design** view.

You should operate either as follows:

1. Delete this property in the **Source** view or execute **Restore Default Value** of popup menu in this tag.
2. Define appropriate value to this property.

When this property is mandatory, you should use the second option above.

Content Tag

You can edit the content of the selected widget or tag in this tab.

You can edit the HTML tags except the following HTML tags (empty element).

- area
- base
- br
- col
- hr
- img
- input
- link
- meta
- param

Moreover, it is possible to edit HTML tags other than the above.

To set your preferences:

1. Select **Window > Preferences**.
2. Select **Property** view of Ajax Page Editor in the **Preferences** dialog.
3. Specify an element name of HTML tag assumed to be edited to **Specify tags that have no content** of the **Content** tab. If you specify multi element names, the character to separate the elements is ','.

Return Code

You enter the return code with either of the following keys:

- Ctrl + Enter
- Shift + Enter

You can select above keys. Default is Ctrl + Enter.

To set your preferences:

1. Select **Window > Preferences**.
2. Select **Property** view of Ajax Page Editor in the **Preferences** dialog.
3. Select above keys in the **Set the return key** group.

Conversion of Escape Character

When the escape is set, the following escape characters are converted; however, when the value has already been set to the content of contents, escape characters are not converted.

Original Character	Converted Character
"	&quot;
&	&amp;
<	&lt;
>	&gt;

To set your preferences:

1. Select **Window > Preferences**.
2. Select **Property** view of Ajax Page Editor in the **Preferences** dialog.
3. Check [Convert escape characters].

8.3 QuickForm Tutorial

This section teaches you how to create a QuickForm which allows a user to start a process instance. The following figure shows the corresponding section in the process definition:

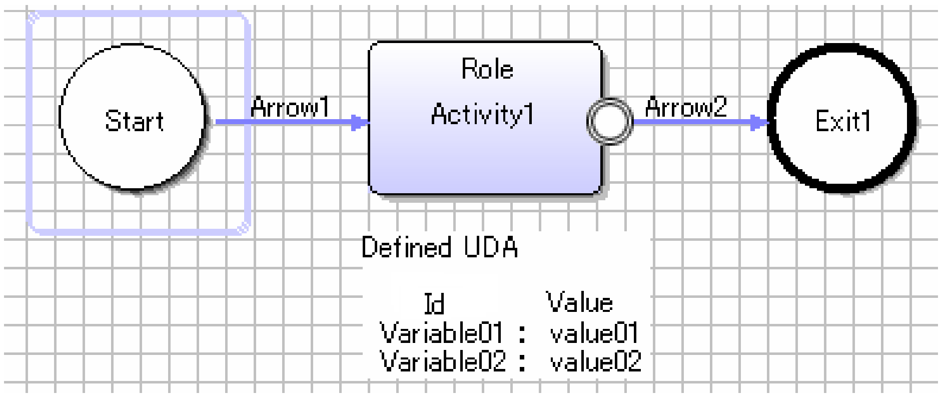


Figure 122: Process Definition for Creating a QuickForm

Step 1: Create the QuickForm

Open the process definition of your Workflow Application project in the Process Definition editor, click the relevant Activity Node, and select **QuickForm > New** from the popup menu.

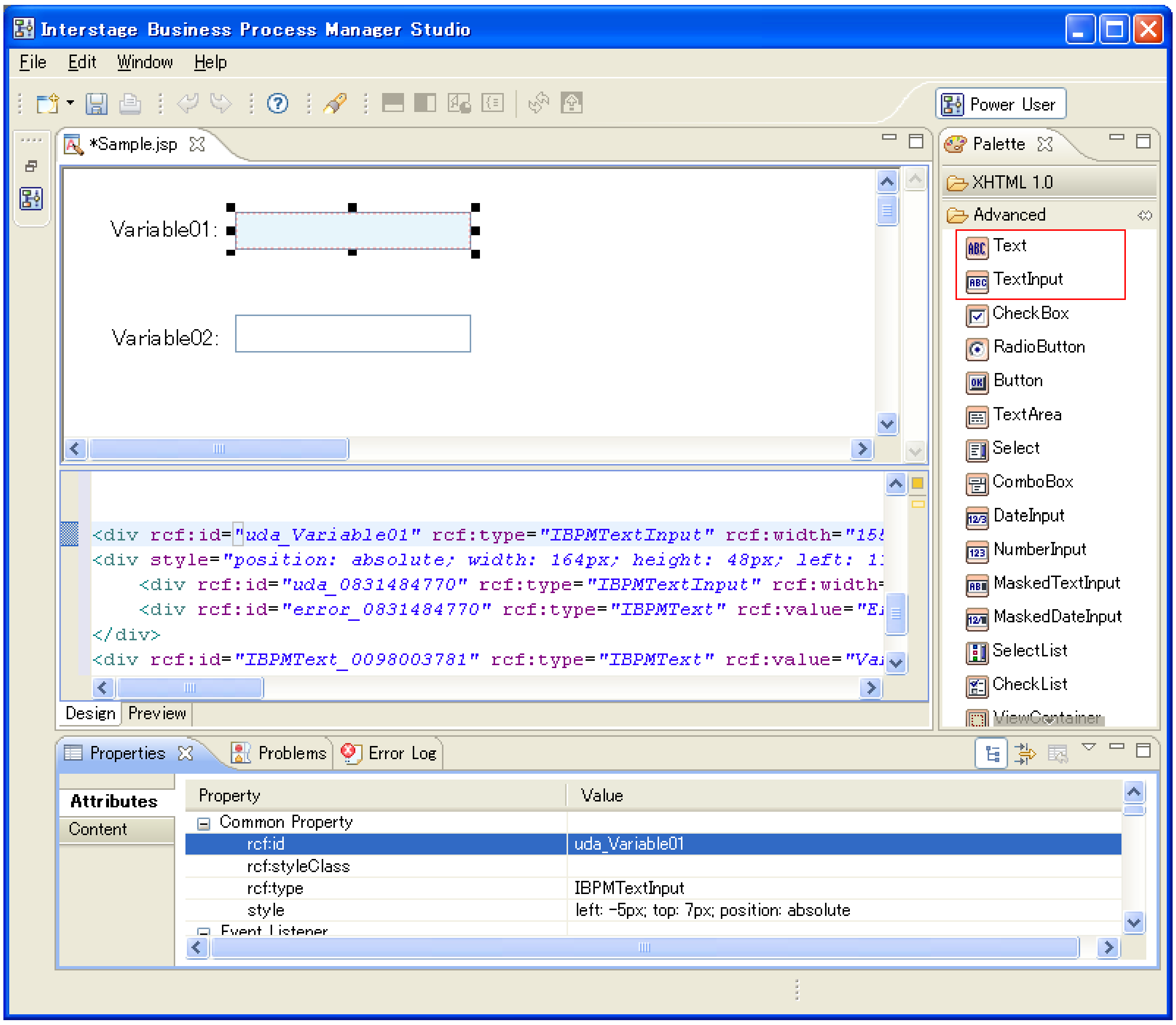
Enter a File Name. Refer to section *Creating QuickForms* on page 191 for details.

The QuickForm (file.jsp) is created in your Workflow Application project.

Step 2: Define the UI Widgets

Proceed with defining the UI widgets for displaying the value of a User Defined Attribute (UDA).

1. Define the UI widget for displaying a UDA value:



```
<div rcf:id="uda_Variable01" rcf:type="IBPMTextInput" rcf:width="15
<div style="position: absolute; width: 164px; height: 48px; left: 1
    <div rcf:id="uda_0831484770" rcf:type="IBPMTextInput" rcf:width=
    <div rcf:id="error_0831484770" rcf:type="IBPMText" rcf:value="E
</div>
<div rcf:id="IBPMText_0098003781" rcf:type="IBPMText" rcf:value="Va
```

Property	Value
⊟ Common Property	
rcf:id	uda_Variable01
rcf:styleClass	
rcf:type	IBPMTextInput
style	left: -5px; top: 7px; position: absolute
⊟ Event Listener	

Figure 123: Defining UI Widgets for Displaying a UDA Value

a. Select the **Text** item of **Advanced** category in the **Palette** view, and click it to the **Design** view in the Ajax Page Editor. Text item is used to distinguish UDAs.

b. Select the **TextInput** item of **Advanced** category in the **Palette** view, and click it to the **Design** view in the Ajax Page Editor. The TextInput and Text are defined in this editor. TextInput item is used to display the value of a UDA. This text item is used to display an error message, when you execute the following operations:

a. You define `true` to this mandatory property but the user does not enter this field in Interstage BPM Console.
b. You relate a UDA which does not support type of this UI Widget; etc

c. Repeat step 1 and 2 to insert another Text and TextInput item.
d. Select the first Text in this editor. In the **Properties** view, in **rcf:value** field, enter **Variable01:**.
e. Select the second Text in this editor. In the **Properties** view, in **rcf:value** field, enter **Variable02:**.
f. Select the first TextInput in this editor. In the **Properties** view, in **rcf:id** field, enter **uda_Variable01**. This value format is **uda_** plus **<identifier of the UDA>**.

   If you need the Text to display an error message, you select this Text and in the **Properties** view, in **rcf:id** field, enter **error_Variable01**. This value format is **error_** plus **<identifier of the UDA>**.
g. Select the second TextInput in this editor. In the **Properties** view, in **rcf:id** field, enter **uda_Variable02**.

   If you do not need the Text to display an error message, you delete this Text.

Note: The Text to display an error message even if you delete TextInput is not deleted together. You need to select and delete this Text.

If you use items of Basic (XHTML1.0), you have to complete the following steps:

1. When you create a new QuickForm, the following tags are defined:

```
<form method="post" action="servlet/IBPMEntryServlet" style="width:
650px; height: 48px">
 <input type="hidden" name="command" value="updateHTMLUDA"/>
 <input type="hidden" name="workItemID" value="<%=wid %>"/>
 <input type="hidden" name="processDefinitionID" value="<%=procDefId
%>"/>
 <input type="hidden" name="requestURL" value="<%=requestURL %>"/>
 <input type="hidden" name="formName" value="<%=formName %>"/>
 <!-- Add HTML controls here -->

 <input type="submit" value="Save" style="position: absolute; width:
80px; height:
 24px; left: 295px; top: 30px"/>
</form>
```

2. A 'Save' button will be provided inside the form which will be used to submit the values in those Basic items . 'Save' action will save both Basic item values as well as the Advanced item values in that form to the server.
3. Select the UI widget of Basic category in the **Palette** view, and click it on this form tab in the **Design** view of the Ajax Page Editor.

4. Select this UI widget. In the **Properties** view, in name field, enter `"uda_ plus <identifier of the UDA>"` and define each UI widgets as follows.

Common Basic Items User Can Define in QuickForm

```
[Text Field] UI widget.  Identifier of the UDA  is FirstName.
<input type=”textfield” name=”uda_FirstName”
value=”<%=ibpmUDAprop.getProperty(“uda_FirstName”)%>” />
```

```
[Select] UI widget. Identifier of the UDA is Selection.
<select name="uda_Selection" >
<%
// Options item string is stored in the array.
// The identifier of UDA that stores option item string is "Options".
// In that case, the separator of this uses the comma.

String[] opts = ibpmUDAprop.getProperty("uda_Options").split(",");

// The identifier of UDA that stores the value that has been selected
// is "Selection".

String selected = ibpmUDAprop.getProperty("uda_Selection");
for(int I=0;i<opts.length;i++){
if(opts[i].equals(selected)){
%>
<option value="<%=opts[i]%>" selected><%=opts[i]%></option>
<%
}else{
%>
<option value="<%=opts[i]%>"><%=opts[i]%></option>
<%
}
}
%>
</select>
// Above code should be modified for multiple selection.
```

```
[CheckBox]  UI  widget.   Identifier  of  the  UDA  is  Checked.
<input  type="checkbox"  name="uda_Check"
value=”<%=ibpmUDAprop.getProperty
(“uda_Checked”)%>”  >
<%=ibpmUDAprop.getProperty(“uda_Checked”)%>
// Checkbox  cannot  be  initial  set  as  checked  depending  on
// the  UDA  value.
```

```
[Radio  Button]UI  widget.   Identifier  of  the  UDA  is  Radio.
<input  TYPE="checkbox"  name="uda_Radio"  ”<%=ibpmUDAprop.getProperty
(“uda_Radio”)%>”  >
<%=ibpmUDAprop.getProperty(“uda_Radio”)%>
//  Radio  Button  cannot  be  initial  set  as  selected
//  depending on the  UDA value.
```

Common Basic Items User Can Define in QuickForm

`[TextArea]UI widget.  Identifier of the UDA  is  Address.`
`<textarea name="uda_Address" rows="10" cols="20">`
`<%=ibpmUDAprop.getProperty`
`("uda_Address")%>`
`</textarea>`

`[Password Field]UI widget.  Identifier of the UDA  is  Password.`
`<input type=”password” name=”uda_Password”`
`value=”<%=ibpmUDAprop.getProperty`
`(“uda_Password”)%>” />`

Note: The following behavior is observed by an Interstage BPM Console user for special characters in Basic item controls:

For a Basic item control mapped to any UDA other than the DATE type UDA, when you enter the special character (",>,< etc) in the textbox on the Process Start page or WorkItem page in Console, the entered string is not displayed correctly. But the UDA is updated correctly.

To avoid these issues, while creating the form, update the form as given below:

1. Use `value="<%=HTMLUtility.escapeHTML(ibpmUDAprop.getProperty("uda_int"))%>"` instead of `value="<%=ibpmUDAprop.getProperty("uda_int")%>"`.
2. Import the `<%@page import="com.fujitsu.ibpmconsole.common.HTMLUtility"%>` class in the .jsp form.
3. Use `HTMLUtility.escapeHTML` so that the HTML characters (",>,< etc) behave as expected.

Note:

- Advanced item control values will have preference over Basic items in a form, hence, if there is an Advanced item and a Basic item both mapped to the same UDA; the final UDA value updated to the server will be that of the Advanced item.
- You can delete the 'Save' button in case you are not using any Basic items.
- The 'Save' button will be available even if there are no UDAs on the form to be saved.
- When you have added multiple forms to a node; the forms are displayed in different subtabs, and each subtab will have a 'Save' button for saving any UDA data in Basic items.
- When you have added multiple forms to a node and the same UDA is used in each form, you need to note that in Interstage BPM Console, you have to enter the same value for these widgets in all the forms. Therefore, it is recommended that you do not map different widgets to the same UDA.
- UDA values entered in Basic items are NOT validated at client side. Any value entered by the user is saved to the DB, without any error.
- When you enter a wrong UDA id or a UDA id which does not exist, the values entered by the user will be lost, as there are no UDA in the process where the value can be stored.

Step 3: Save the QuickForm

By saving the QuickForm, an JSP file is created in your Workflow Application project.

Step 4: Associate the QuickForm with a Process Definition

As a last step, you need to associate the newly QuickForm with a process definition. You can associate the form to any process definition:

1. Open the process definition in the Process Definition editor.
2. In the Navigator view, select the newly created QuickForm.
3. Drag it and drop it on the Start Node of the process definition.

Refer to section *Managing QuickForms* on page 209 for details.

8.4 Managing QuickForms

This section explains the functions available for managing QuickForms.

8.4.1 Updating QuickForms

You can make changes to an existing QuickForm. For example, you can add additional User Defined Attributes (UDAs) to the form, change widget types, and so on.

To update a QuickForm:

1. To open an existing QuickForm in the Ajax Page Editor, there are two ways:
   a) In the Navigator view, double click the relevant QuickForm, or right click the form and select **Open** from the popup menu.
   b) In the Process Definition editor, right click the node with which the QuickForm has been associated and select **QuickForm > Edit** from the popup menu.
2. In the Ajax Page Editor, make the required changes.

   For more information, refer to section *Ajax Page Editor* on page 192.

3. Click the Save button of the Workbench window to save your changes.

8.4.2 Associating QuickForms with Nodes

When you started creating a QuickForm from within the Process Definition editor for a particular node, it is automatically associated with the node for which you created it. You can reuse existing QuickForms and associate them with additional nodes.

To associate an existing QuickForm with a node:

1. Open the relevant process definition of your Workflow Application project.
2. In the Navigator view, click the QuickForm that you want to associate with a Start Node, Activity Node, Voting Activity Node, or Compound Activity Node of the process definition.
3. Drag the form to the respective node.

   In the **Forms** tab of the Properties view for the node, you will see the **Form Title** and the **Form Path** of the form associated with the node.

To reuse an existing QuickForm:

1. Open the relevant process definition of your Workflow Application project.
2. Right click the node to which you want to associate a QuickForm and select **QuickForm** > **Add reference** from the popup menu. The following dialog is opened:

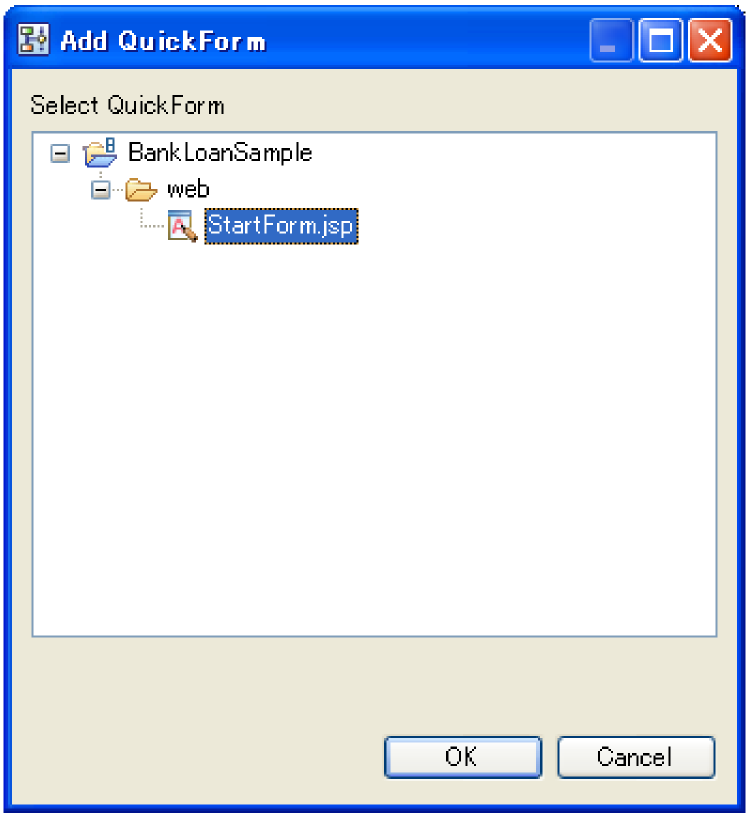


Figure 124: Creating a QuickForm

3. From the list of available QuickForms, select the one that you want to associate with the given node, and click **OK**.

Note: **Browse** button in the Forms tab of the Properties View of Start Node, Activity Node, Voting Activity Node, or Compound Activity Node provides the function which associates the node with the form, besides the QuickForm. Associate the QuickForm and the node in the operation as shown above.

8.4.3 Dissociating QuickForms from Nodes

You can dissociate a QuickForm from a node. This removes the association, but does not remove the form itself.

To dissociate a QuickForm from a node using the node Properties view:

1. Select the node to display the Properties view for it.
2. Select the **Forms** tab.
3. Select the form to be dissociated and click **Remove**.

To dissociate a QuickForm from a node using the node popup menu:

1. Right click the respective node in the Process Definition editor.
2. Select **QuickForm > Delete Reference** from the popup menu.
3. Select the form to be dissociated and click **OK**.

8.4.4 Renaming QuickForms

You can rename a QuickForm and take care that all references to this form are also taken into account.

To rename a QuickForm:

1. In the Navigator view, right click the respective QuickForm and select **Rename** from the popup menu.
2. Enter a new name for the form in the **New name** field.
3. Decide whether all references to this QuickForm should also be updated so that they use the new name.
4. Click **OK**.

8.4.5 Deleting QuickForms

You can delete a QuickForm and take care that all references to this form are also taken into account.

To delete a QuickForm:

1. In the Navigator view, right click the respective QuickForm and select **Delete** from the popup menu.
2. Decide whether all references to this QuickForm should also be deleted.
3. Confirm the deletion of the form by clicking **Yes**.

8.4.6 Importing QuickForms

Prerequisite: You have created a Workflow Application project where the QuickForms can be imported.

You can import QuickForms into Interstage BPM Studio.

To import a QuickForm:

1. In the Navigator view, right click the project where you want to import the form. Select **Import** from the pop-up menu.
2. Navigate to the location where the QuickForm is stored.
3. Select the form and click **Finish**.

   Refer section *Importing Folders and Files* on page 91 for details.

8.4.7 Exporting QuickForms

You can export QuickForms from Interstage BPM Studio to the file system. You can use the exported files, for example, to import them into other systems.

Note: You can export entire Workflow Application projects including all process definitions, forms, attachments, etc. in one step, and later deploy the application on an Interstage BPM server. Refer to section *Exporting Workflow Application Projects* on page 100 for details.

To export a QuickForm:

1. In the Navigator view, right click the QuickForm. Select **Export**.
2. Navigate to the location where the exported form is to be stored.
3. Click **Save**.

8.5 Creating QuickForms (for compatibility)

Prerequisite: You have specified User Defined Attributes (UDAs) that are to appear on the QuickForms. Otherwise, Interstage BPM Studio will not allow you to create a QuickForm.

Note: This function is provided for compatibility. If you want to create a new QuickForm, please refer *Creating QuickForms* on page 191.

You need to differentiate whether you create a QuickForm in a Workflow Application project or whether you create one in a project that has been created with a previous version of Interstage BPM Studio (local or server projects):

The subsequent sections describe this two cases.

8.5.1 Creating QuickForms in Workflow Application Projects

To create a QuickForm in a Workflow Application project:

1. Select the Start Node, Activity Node, Voting Activity Node, or Compound Activity Node for which you want to create a form, to display the Properties view for those nodes.
2. Select the **Forms** tab.
3. Click **Create New** to open the **Form Properties** dialog.
4. In the **Destination Path** field, replace the default file name of the form with a file name that describes the form and is easily recognized.

   If you change the file name, make sure to specify `.qf` as the extension.

   As a default, QuickForms are stored in the `web` folder of your Workflow Application project. If you want to organize your forms in different folders, you can create subfolders under the `web` folder. You can select these subfolders using the **Browse** button.
5. In the **Form Title** field, provide a descriptive title that is to appear on the form.

The title cannot be longer than 64 characters.

6. To define the look and feel of your form, do the following:
    a) If you already created a form template that you want to use, click **Browse** next to the **Use Template** field and select the form template.

       For more information on form templates, refer to section *Creating Templates for QuickForms* on page 219.
    b) Select the cascading style sheet that you want to use.

       You can select from a set of predefined style sheets that come with Interstage BPM. If you want to use your own style sheets, refer to section *Integrating your own Cascading Style Sheets in QuickForms* on page 220.
7. Specify which of the following components are to be included in your form: the work item header, process panel, and attachment panel.

   Use the **Work Item Header** if you want the form to look the same as the other Interstage BPM Console pages, including the navigation tabs and other buttons that appear. Without this, you can make the form look any way you want to.

   The **Process Panel** displays details about the current state of the work item including who it is assigned to, who started the process instance, when it was assigned, when it is due to be completed, what the choices are, and buttons to accept, decline, or reassign the work item.

   The **Attachment Panel** lists the documents currently attached to the process instance, and provides buttons for adding, removing or editing attachments. The attachment panel depends on the process panel. Therefore, when using the attachment panel you must also use the process panel.

Note: If you are using the Interstage BPM Console to start process instances and respond to work items, you must use the process panel. Otherwise, users will not be given any choices on the form and users will not be able to complete work items. Only deselect the process panel if you provide a custom Interstage BPM client.

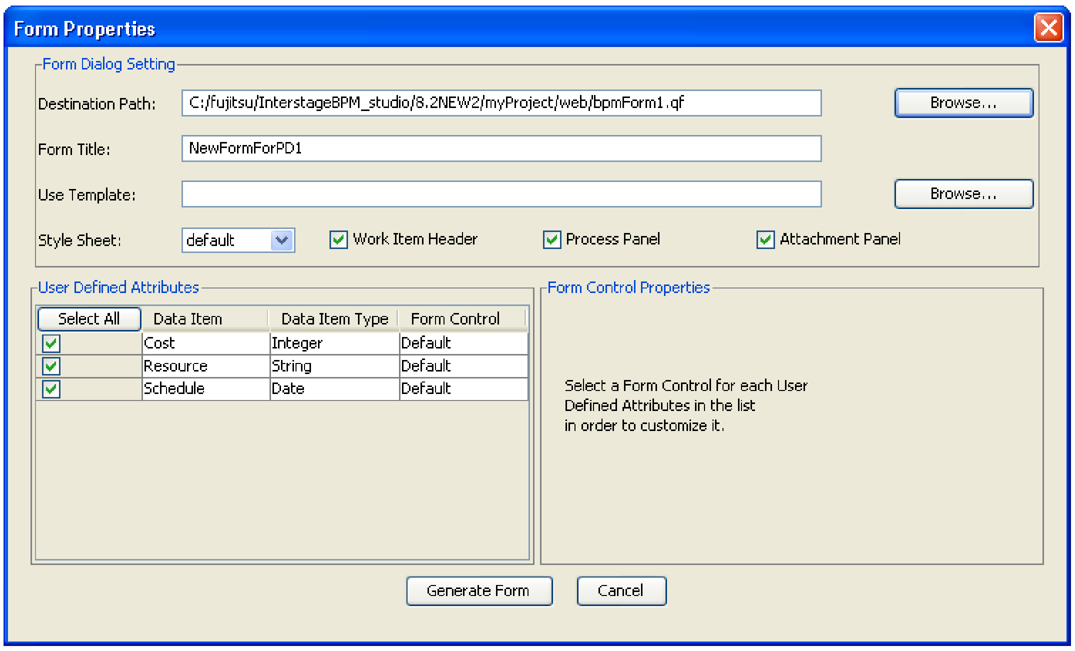
8. For each UDA that you want to appear on the form, do the following:
    a) Make sure that it is selected in the **User Defined Attributes** area.
    b) In the **Form Control** column, select a control type for the UDA.
    c) Check the properties of the control and change them if required. For more information, refer to section *Form Controls for QuickForms* on page 217.
    d) In the **Form Control Properties** area, click **Save** to save your changes.

Note: You must click **Save** after making changes to the form control's properties, or they will not take effect.

   e) For text fields, text areas, and read only fields: If you want to use your current settings as the default for that form control type, click **Set Default**.

      The settings are applied to all new fields of that form control type.
   f) For text fields, text areas, and read only fields: If you want to use the default settings that Interstage BPM Studio has defined for that form control type, click **Use Default**.

The default settings are applied to the field that you are currently editing.


Form Properties

Form Dialog Setting

Destination Path: C:/fujitsu/InterstageBPM_studio/8.2NEW2/myProject/web/bpmForm1.qf

Form Title: NewFormForPD1

Use Template:

Style Sheet: default

✓ Work Item Header ✓ Process Panel ✓ Attachment Panel

User Defined Attributes

Select All	Data Item	Data Item Type	Form Control
✓	Cost	Integer	Default
✓	Resource	String	Default
✓	Schedule	Date	Default

Form Control Properties

Select a Form Control for each User Defined Attributes in the list in order to customize it.

Figure 125: Generating a QuickForm

9. Click **Generate Form**.

   The form is generated and stored in the folder that you specified. The **Forms** tab displays the form title and its location.

10. To check the appearance, open the form in your Web Browser. You can customize the layout using any HTML Editor or by modifying the HTML code directly.

Note: When customizing QuickForms, you must ensure that the field names remain the same. For more information on customizing QuickForms, refer to the *Interstage Business Process Manager Developer's Guide*.

8.5.2 Creating QuickForms in Local Projects

To create a QuickForm in an existing local project:

1. Select the Start Node, Activity Node, Voting Activity Node, or Compound Activity Node for which you want to create a form, to display the Properties view for those nodes.
2. Select the **Forms** tab.
3. Click **Create New** to open the **Form Properties** dialog.
4. In the **Destination Path** field, replace the default file name of the form with a file name that describes the form and is easily recognized.

   If you change the file name, make sure to specify `.html` as the extension.

   As a default, QuickForms are stored in the `<Interstage BPM Studio Installation Directory>\ibpm\ApplicationCore\ledev\QuickForms` folder. If you want to organize your forms in different folders, you can create subfolders under the `QuickForms` folder. You can select these subfolders using the **Browse** button. However, you must use the `QuickForms` folder or one of its subfolders for form generation.

5. In the **Form Title** field, provide a descriptive title that is to appear on the form.

   The title cannot be longer than 64 characters.

6. To define the look and feel of your form, do the following:

   a) If you already created a form template that you want to use, click **Browse** next to the **Use Template** field and select the form template.

      For more information on form templates, refer to section *Creating Templates for QuickForms* on page 219.

   b) Select the cascading style sheet that you want to use.

      You can select from a set of predefined style sheets that come with Interstage BPM. If you want to use your own style sheets, refer to section *Integrating your own Cascading Style Sheets in QuickForms* on page 220.

7. Specify which of the following components are to be included in your form: the work item header, process panel, and attachment panel.

   Use the **Work Item Header** if you want the form to look the same as the other Interstage BPM Console pages, including the navigation tabs and other buttons that appear. Without this, you can make the form look any way you want to.

   The **Process Panel** displays details about the current state of the work item including who it is assigned to, who started the process instance, when it was assigned, when it is due to be completed, what the choices are, and buttons to accept, decline, or reassign the work item.

   The **Attachment Panel** lists the documents currently attached to the process instance, and provides buttons for adding, removing or editing attachments. The attachment panel depends on the process panel. Therefore, when using the attachment panel you must also use the process panel.

Note: If you are using the Interstage BPM Console to start process instances and respond to work items, you must use the process panel. Otherwise, users will not be given any choices on the form and users will not be able to complete work items. Only deselect the process panel if you provide a custom Interstage BPM client.

8. For each UDA that you want to appear on the form, do the following:

   a) Make sure that it is selected in the **User Defined Attributes** area.

   b) In the **Form Control** column, select a control type for the UDA.

   c) Check the properties of the control and change them if required. For more information, refer to section *Form Controls for QuickForms* on page 217.

   d) In the **Form Control Properties** area, click **Save** to save your changes.

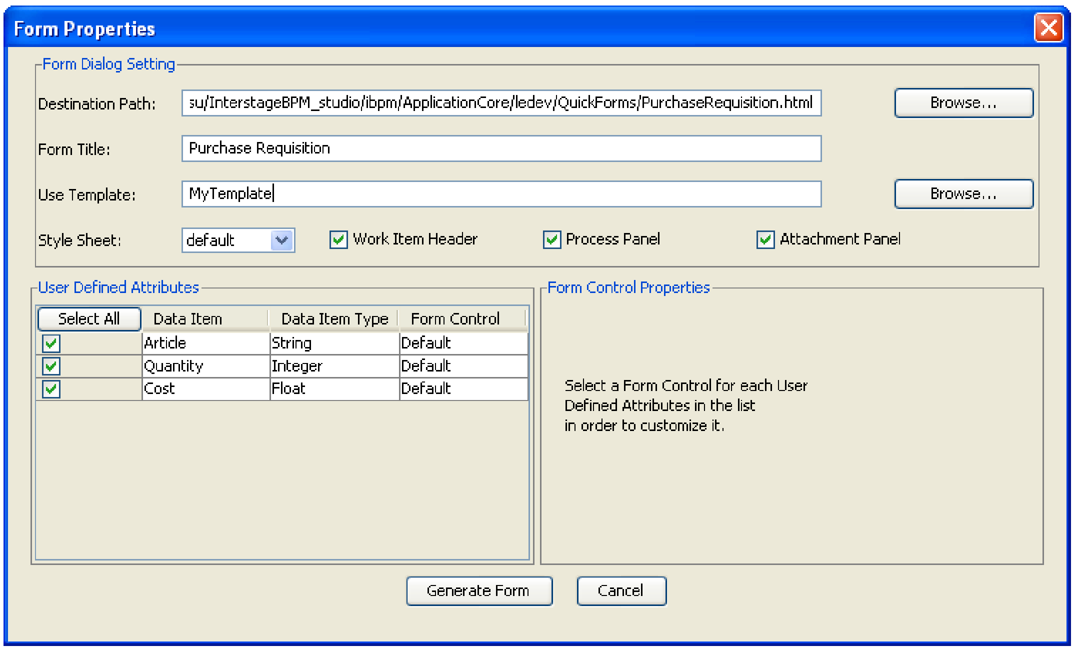
Note: You must click **Save** after making changes to the form control's properties, or they will not take effect.

   e) For text fields, text areas, and read only fields: If you want to use your current settings as the default for that form control type, click **Set Default**.

      The settings are applied to all new fields of that form control type.

   f) For text fields, text areas, and read only fields: If you want to use the default settings that Interstage BPM Studio has defined for that form control type, click **Use Default**.

The default settings are applied to the field that you are currently editing.



Form Properties

Form Dialog Setting

Destination Path: su/InterstageBPM_studio/ibpm/ApplicationCore/ledev/QuickForms/PurchaseRequisition.html

Form Title: Purchase Requisition

Use Template: MyTemplate

Style Sheet: default

✓ Work Item Header ✓ Process Panel ✓ Attachment Panel

User Defined Attributes

Select All	Data Item	Data Item Type	Form Control
✓	Article	String	Default
✓	Quantity	Integer	Default
✓	Cost	Float	Default

Form Control Properties

Select a Form Control for each User Defined Attributes in the list in order to customize it.

Figure 126: Generating a QuickForm

9. Click **Generate Form**.

   The form is generated and stored in the folder that you specified. The **Forms** tab displays the form title and its location.

10. To check the appearance, open the form in your Web Browser. You can customize the layout using any HTML Editor or by modifying the HTML code directly.

Note: When customizing QuickForms, you must ensure that the field names remain the same. For more information on customizing QuickForms, refer to the *Interstage Business Process Manager Developer's Guide*.

Note: When sending the process definition to a machine on which the Interstage BPM Console is installed, make the following adjustments:

1. If you send a QuickForm to the Interstage BPM Console for the **first time**, you need to configure the Interstage BPM Server as follows:
   a. Shutdown the Interstage BPM Server.
   b. Create the following directory:
   `<Interstage BPM Server Installation Directory>/server/instance/default/QuickForms`
   c. Open the following file and insert the part indicated in bold below:
   `<Interstage BPM Server Installation Directory>/server/instance/default/resources/dmscollections/DmsCollections.xml`

```
<DmsCollections>
   <Dms>
      <Path>...</Path>
      <ImplementationClass>...</ImplementationClass>
      <Name>Attachments</Name>
   </Dms>
   <Dms>
      <Path>
        <Server Installation
Directory>/server/instance/default/QuickForms
      </Path>
      <ImplementationClass>
         com.fujitsu.iflow.dmsadapter.impl.FileSystemDmsSessionImpl
      </ImplementationClass>
      <Name>QuickForms</Name>
   </Dms>

</DmsCollections>
```

   d. Restart the Interstage BPM Server.
2. Copy the QuickForm to `<Interstage BPM Server Installation Directory>/server/instance/default/QuickForms`

8.5.3 Form Controls for QuickForms

When creating QuickForms, you define which form controls display the User Defined Attributes (UDAs) on the form. You can select among several form control types like text fields, text areas, lists, and so on. Your actual choices depend on the UDA's data type. For example, you can use a Calendar control only with UDAs of type DATE.

These are the form control types and their properties.

Text Field

Displays a single-line text entry field.

Use the **Character Width** property to specify the width of the field and the **Max Characters** property to specify the maximum number of characters that can be entered in the field.

Password Field

Displays a single-line text entry field with "*" replacing the actual data entered by the user.

Use the **Character Width** property to specify the width of the field and the **Max Characters** property to specify the maximum number of characters that can be entered in the field.

Read Only Field

This value is displayed but cannot be changed on the form.

Use the **Character Width** property to specify the width of the field.

Text Area

Displays a multiple-line text entry field.

Use the **Columns** property to specify the width of the text area, that is the number of characters per line. Use the **Rows** property to specify its length, that is the number of lines.

Combo Box

Displays a drop-down list (single-line display that expands to show more entries). Users are allowed to select only one entry.

Use **List Values** to specify the entries that users can select from. For each entry, you define the name that is displayed to the user and the value that is assigned to the UDA. When defining the value that is assigned to the UDA, make sure that the value matches the UDA's data type.

A sample entry is provided and its value is set to the initial value of the UDA. You can change the sample entry according to your needs.

Say you want to create a combo box that allows users to select a customer level. You want to display descriptive names like `Silver` or `Gold` on the form. However, a numerical value representing the customer level is to be assigned to the UDA. In this case, you could specify the following entries:

Name	Value
Silver	150
Gold	160
Platinum	170

Radio Button

Displays a limited number of options. Users are allowed to select only one.

Use **List Values** to specify the options that users can select from. For each option, you define the name that is displayed to the user and the value that is assigned to the UDA. When defining the value that is assigned to the UDA, make sure that the value matches the UDA's data type.

A sample option is provided and its value is set to the initial value of the UDA. You can change the sample option according to your needs.

Check Box

Displays a limited number of options. Users are allowed to select one or more options.

Use **List Values** to specify the options that users can select from. For each option, you define the name that is displayed to the user and the value that is assigned to the UDA. When defining the value that is assigned to the UDA, make sure that the value matches the UDA's data type.

A sample option is provided and its value is set to the initial value of the UDA. You can change the sample option according to your needs.

List

Displays a multiple-line display with a vertical scroll bar if required.

Use the **Row** property to specify the length of the list, that is the number of lines. Use the **Multiple Selection** property to determine whether users are allowed to select multiple entries from the list.

Use **List Values** to specify the entries that users can select from. For each entry, you define the name that is displayed to the user and the value that is assigned to the UDA. When defining the value that is assigned to the UDA, make sure that the value matches the UDA's data type.

A sample entry is provided and its value is set to the initial value of the UDA. You can change the sample entry according to your needs.

Calendar

Displays a text field with a calendar icon. Clicking the icon displays a pop-up calendar from which users can pick a date that will display in the text field.

Hidden Field

This value is not displayed on the form.

A hidden field is useful when the data that is stored in the UDA is not directly in the form that is convenient for display. A JavaScript could read the UDA's value, convert it to a displayable form, and place that form in a visible form control.

Default

Uses the default form control defined for the UDA's data type. These are the default form controls:

Data Type of UDA	**Default Form Control**
BIGDECIMAL	Text field
BOOLEAN	Radio buttons with the options `true` and `false`
DATE	Calendar
FLOAT	Text field
INTEGER	Text field
LONG	Text field
STRING	Text area

8.5.4 Creating Templates for QuickForms

A form template is an HTML file that contains placeholders for Interstage BPM-specific content. Any standard HTML component can be included in the form template.

Once you have provided a form template, you can create new QuickForms based on that template. This simplifies form creation because you need to implement your particular needs only once in the template and not in every form that you create. Also, using templates helps you to have a consistent look and feel across your QuickForms.

To create a form template for QuickForms:

1. Create a new HTML file that implements the look and feel that you want in your QuickForms.
2. Add the following HTML tags inside the `<body>` tags of your HTML file:

   `{{QuickTemplateStart}}{{QuickTemplateEnd}}`

   You have now created an HTML file that can be used as a form template.
3. When creating a template in a project other than a Workflow Application project: Copy the HTML file that you have just created to the attachments folder used by Interstage BPM Studio:

   `<Interstage BPM Studio Installation Directory>\ibpm\Data\attachments`

   When you create a template for a form stored in a Workflow Application project, the template file is, in analogy to the QuickForm files, stored in the `web` folder of your Workflow Application project.

When creating QuickForms, you can now select your form template. When updating QuickForms, you can use your newly created form template if the form has not been generated using another form template.

Note:
- You can change your form templates at any time. However, these changes do not affect existing QuickForms using that form template.
- A QuickForm can be associated with one form template only. Once you have generated a QuickForm using a form template, you cannot associate another form template with that form nor dissociate the form template from that form.

8.5.5 Integrating your own Cascading Style Sheets in QuickForms

When creating a QuickForm, you specify a cascading style sheet (CSS) that will be used with your form. The style sheet defines the QuickForm's look and feel.

Interstage BPM provides a set of predefined style sheets. You can configure Interstage BPM Studio to use your own style sheets. This style sheet can be any standard-format HTML style sheet.

To configure Interstage BPM Studio to use your own style sheet for QuickForms:

1. Open the QuickForms configuration file that is used by Interstage BPM Studio:

   `<Interstage BPM Studio Installation Directory>\ibpm\ApplicationCore\ledev\Config\QuickFormCSSDef.xml`
2. Add a new `<style>` section and specify the name of your style sheet.

   The name will be displayed in the **Form Properties** dialog of Interstage BPM Studio. The following example shows the addition of a style sheet called `ExampleStyle`.

```
<Styles>
<style>
<name>default</name>
<path>styles/iflow.css</path>
</style>
<style>
<name>ExampleStyle</name>
<path></path>
</style>
```

```
…
</Styles>
```

Note: Interstage BPM Studio does not access the CSS file itself. Therefore, the style sheet's path is not relevant with Interstage BPM Studio.

When creating or updating QuickForms, you can now select your style sheet.

Note: When sending process definitions that reference the style sheet to an Interstage BPM Server, make the following adjustments:

1. Copy the style sheet to the following folder:

   `<Interstage BPM Console Installation Directory>/apps/System/web/styles`

   The style sheet must have a `css` file extension.

2. Add the style sheet that you copied in the previous step to the QuickForms configuration file that is used by the Interstage BPM Server. To do so, add a `<style>` section to the following file:

   `<Interstage BPM Console Installation Directory>/apps/System/web/Config/QuickFormCSSDef.xml`

   The following example shows the addition of a style sheet called `ExampleStyle`. The underlying CSS file is called `example.css`. Its path has to be specified relative to the `<Interstage BPM Console Installation Directory>/apps/System/web`.

   ```
   <Styles>
   <style>
   <name>default</name>
   <path>styles/iflow.css</path>
   </style>
   <style>
   <name>ExampleStyle</name>
   <path>styles/example.css</path>
   </style>
   …
   </Styles>
   ```

   Make sure that you specify the same name as in the QuickForms configuration file of Interstage BPM Studio.

8.6 Managing QuickForms (for compatibility)

8.6.1 Updating QuickForms

You can make changes to an existing QuickForm. For example, you can add additional User Defined Attributes (UDAs), change control types, assign a different cascading style sheet, and so on. The location of the QuickForm and its title cannot be changed.

Note: When updating a QuickForm, Interstage BPM replaces the existing form with a newly generated form. If you have customized the layout of the existing form, all customizations will be lost.

To update a QuickForm:

1. Select the node with which the QuickForm has been associated, to display the Properties view for it.
2. Select the **Forms** tab.
3. Select the form and click **Update**.
4. In the **Form Properties** dialog, make the required changes.

   For more information, refer to section *Creating QuickForms (for compatibility)* on page 212 and to section *Form Controls for QuickForms* on page 217.
5. Click **Update Form**.
6. Confirm that you want to overwrite the existing form.

Note: If you already copied the QuickForm to an Interstage BPM Server, make sure to copy the updated form to the server as well.

8.6.2 Associating QuickForms with Nodes

When you create a QuickForm, it is automatically associated with the node for which you created it. You can reuse existing QuickForms and associate them with additional nodes.

To associate an existing QuickForm with a node:

1. Select the Start Node, Activity Node, Voting Activity Node, or Compound Activity Node with which you want to associate the form, to display the Properties view.
2. Select the **Forms** tab.
3. Click **Browse** and select a QuickForm that you previously created.
4. Click **OK** to associate the form.

   You return to the **Forms** tab. The title of the form and its location is displayed.

8.6.3 Dissociating QuickForms from Nodes

You can dissociate a QuickForm from a node. This removes the association, but does not remove the QuickForm itself.

To dissociate a QuickForm from a node:

1. Select the node to display the Properties view for it.
2. Select the **Forms** tab.
3. Select the form to be dissociated and click **Remove**.

9 UDAs of Type CUSTOM

In Interstage BPM, you can add not only basic types INTEGER and STRING as types of User Defined Attributes, but also UDA of type XML which has complicated data structure. In each UDA of type XML, you need to define XML Schema which defines UDA structure.

You can save XML Schema to a file. Then, you can define UDA of type CUSTOM by using this file, in the same project where XML Schema is defined.

Interstage BPM Studio provides the following functions in order to handle the UDA of type CUSTOM:

- You can extract XML Schema files and WSDL files as `*.xsd` files. You can import these files to Workflow Application project.
- You can import XML Schema files or WSDL files from local file system or remote locations using HTTP protocol (URL).
- You can create/edit XML Schema file in Workflow Application project by using XML Schema Editor.
- You can define CUSTOM UDA type (XML Schema) and use it just like the existing XML UDA type in most views or dialogs.
- You can use application (process definitions and schema files) validation function which provides information about inconsistency related to UDA of type CUSTOM.

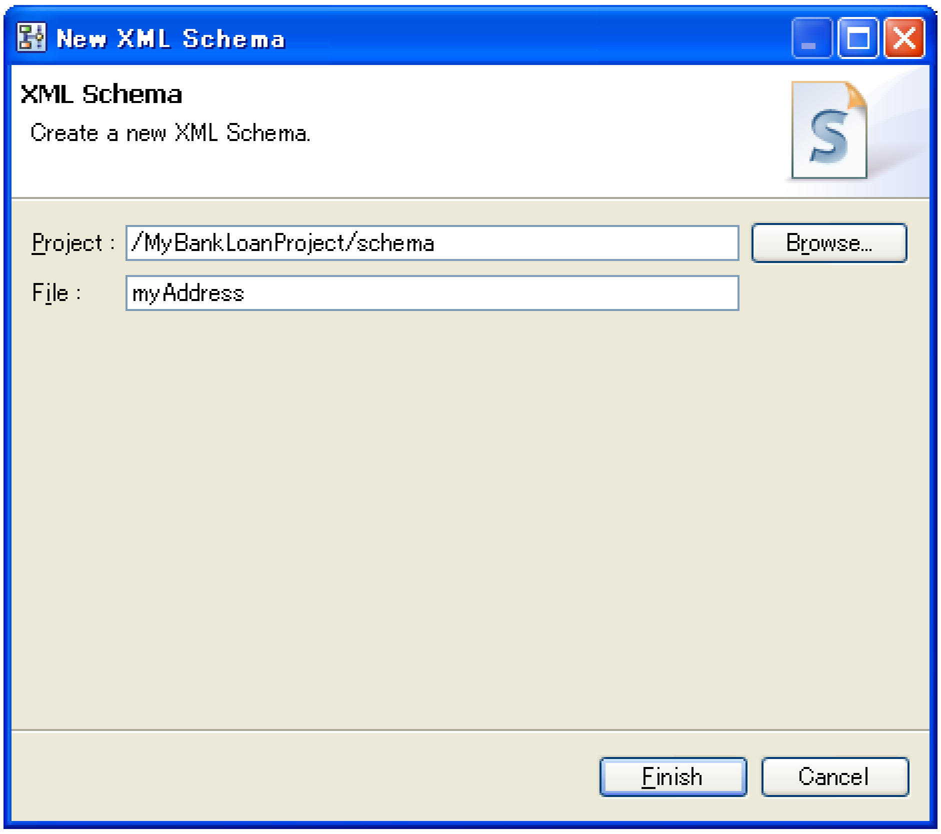
9.1 Creating XML Schema File

You can create XML Schema file using the XML Schema Editor.

To create XML Schema file:

1. In Navigator view, select the Workflow Application project for which you want to create an XML schema file, and select **New > XML Schema** from the **File** menu.

New XML Schema dialog is displayed.



Project : /MyBankLoanProject/schema

File : myAddress

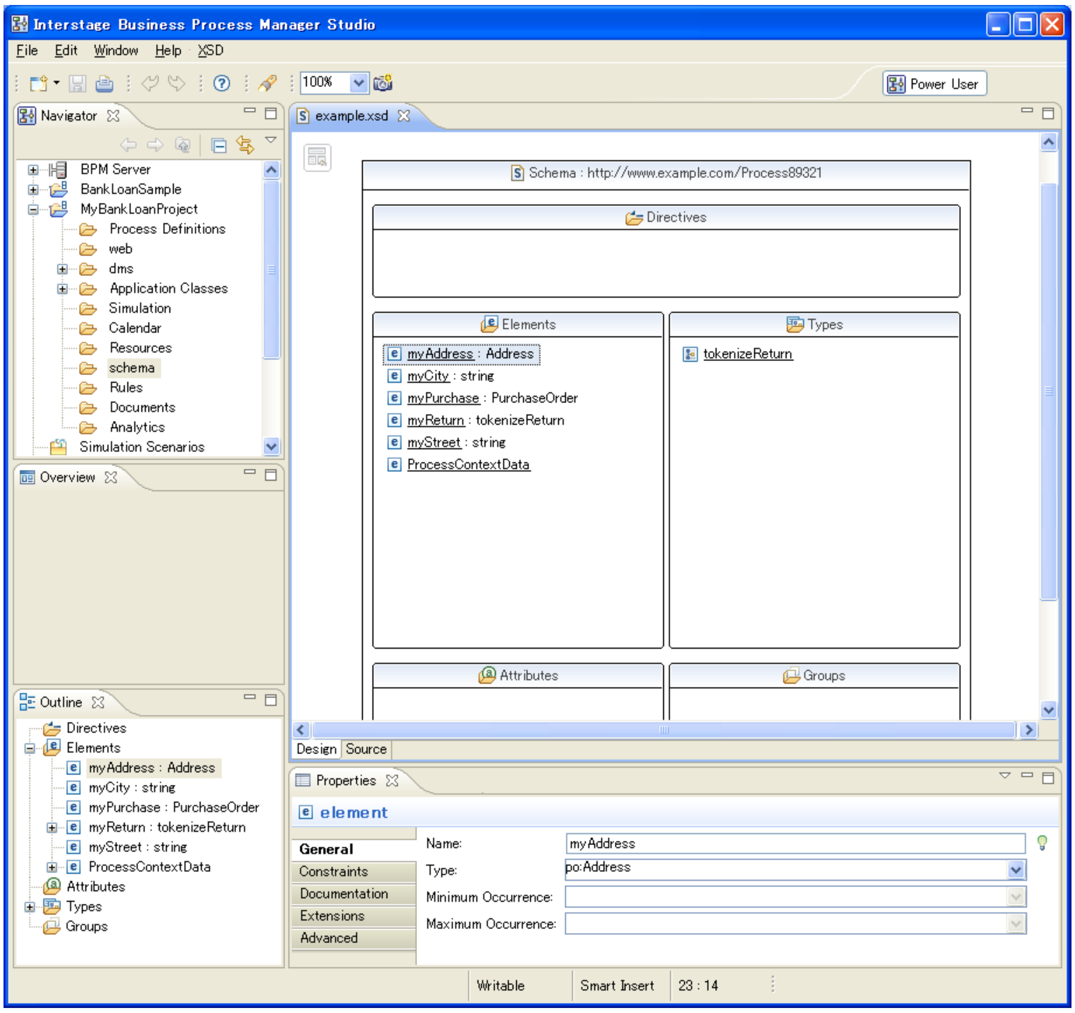
Figure 127: Creating XML Schema File

2. Enter the XML Schema name in **File** field and click **Finish**.
   New XML Schema file (`.xsd` file) is created in **schema** folder.

9.2 XML Schema Editor Views

XML Schema Editor is the tool which you can use to edit an XML Schema file.
In Design view, the visual structure of the XML Schema is displayed.

In Source view, the code structure of the XML Schema is displayed.



Name:	myAddress
Type:	po:Address
Minimum Occurrence:	
Maximum Occurrence:	

Figure 128: XML Schema Editor in Interstage BPM Studio

9.2.1 Overview

The following figure shows the outline of the XML Schema Editor.

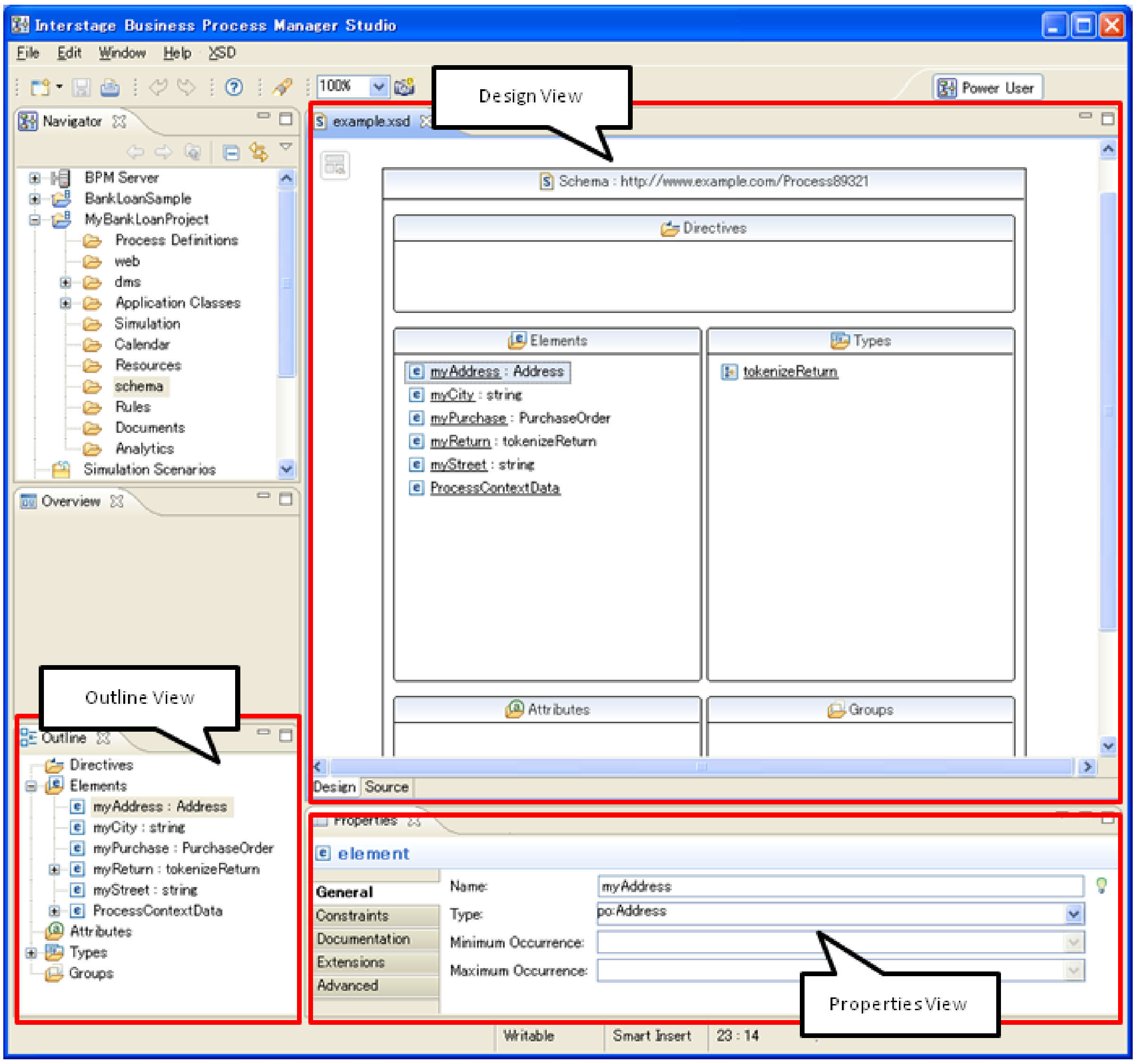


Figure 129: XML Schema Editor

- Design view:
  - This view shows the outline of the XML Schema Editor (Elements, Types, etc.).
  - At the top of the view, the name space where XML Schema is defined, is displayed. **Directives** shows reference which XML Schema uses. **Elements** defines the structure of the document. **Type** shows multiplicity, orders, or rules.
- Source view:
  - You can edit the XML Schema code in this view. The changes are reflected in Design view.
  - You can switch the view using the **Design** and **Source** tabs of the editor.

- Outline view: It highlights component of the XML Schema Editor which is being edited by you.
- Property view: It displays the properties of the components selected in Design view.

9.2.2 Working with XML Schema Editor

For details about editing operations on XML Schema file, refer chapter *Working with XML schemas* in online help of Interstage BPM Studio. For details about other operations, refer relevant topics in *UDAs of Type CUSTOM* on page 223 and other chapters in Interstage BPM Studio User's Guide. You can access online help by clicking **Help Contents** in the **Help** menu of Interstage BPM Studio.

Note:
- You cannot use **include** and **redefine** editor icons.
- You cannot edit XML Schema file preferences.
- In **attribute ref** view and **element ref** view, do not to use the **Add** button in the **Constraints** tab. If you click this **Add** button, syntax error is displayed in Source view.

Note: When you double-click an XML Schema file in **schema** folder in **Navigator** view, that XML file is opened in XML Schema Editor. In this case, the user session in XML Schema Editor remains active and focus is not moved to the **Navigator** view even if you click that view. In this scenario, some menu items (for example, creating new process definition) might be disabled. You can change the focus correctly by clicking the window other than the **Navigator** view.

9.3 Using UDAs of Type CUSTOM

This section provides you information about how to create an XML Schema and define it as a UDA of type CUSTOM in a process definition.

1. Create an XML Schema by any of the two methods described below:
    1. Create an XML Schema using the XML Schema Editor. Refer *Creating XML Schema File* on page 223 for details. For details about working with XML Schema Editor, refer online help by clicking **Help Contents** in the **Help** menu of Interstage BPM Studio.
    2. Import XML Schema from an XSD or WSDL file.

       The imported file (for example, `purchaseorder.xsd`) should be as given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.example.com/PO1"
      xmlns:po="http://www.example.com/PO1">
 <xsd:complexType name="PurchaseOrder">
  <xsd:sequence>
   <xsd:element name="customerName" type="xsd:string"/>
   <xsd:element name="dueDate"      type="xsd:date"/>
   <xsd:element name="shipTo"       type="po:Address"/>
   <xsd:element name="billTo"       type="po:Address"/>
   <xsd:element name="totalCost"    type="xsd:decimal"/>
   <xsd:element name="items"  type="po:LineItem" minOccurs="0"
     maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="Address">
  <xsd:sequence>
   <xsd:element name="street" type="xsd:string"/>
   <xsd:element name="city"   type="xsd:string"/>
   <xsd:element name="state"  type="xsd:string"/>
   <xsd:element name="zip"    type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="LineItem">
  <xsd:sequence>
   <xsd:element name="productName" type="xsd:string"/>
   <xsd:element name="quantity"    type="xsd:string"/>
   <xsd:element name="price"       type="xsd:decimal"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

When the XSD file is imported, `purchaseorder.xsd` will be created in the **schema** folder in **Navigator** view. When you double-click `purchaseorder.xsd`, XML Schema Editor is displayed and shows `purchaseorder.xsd`. The following figure shows the XML Schema:

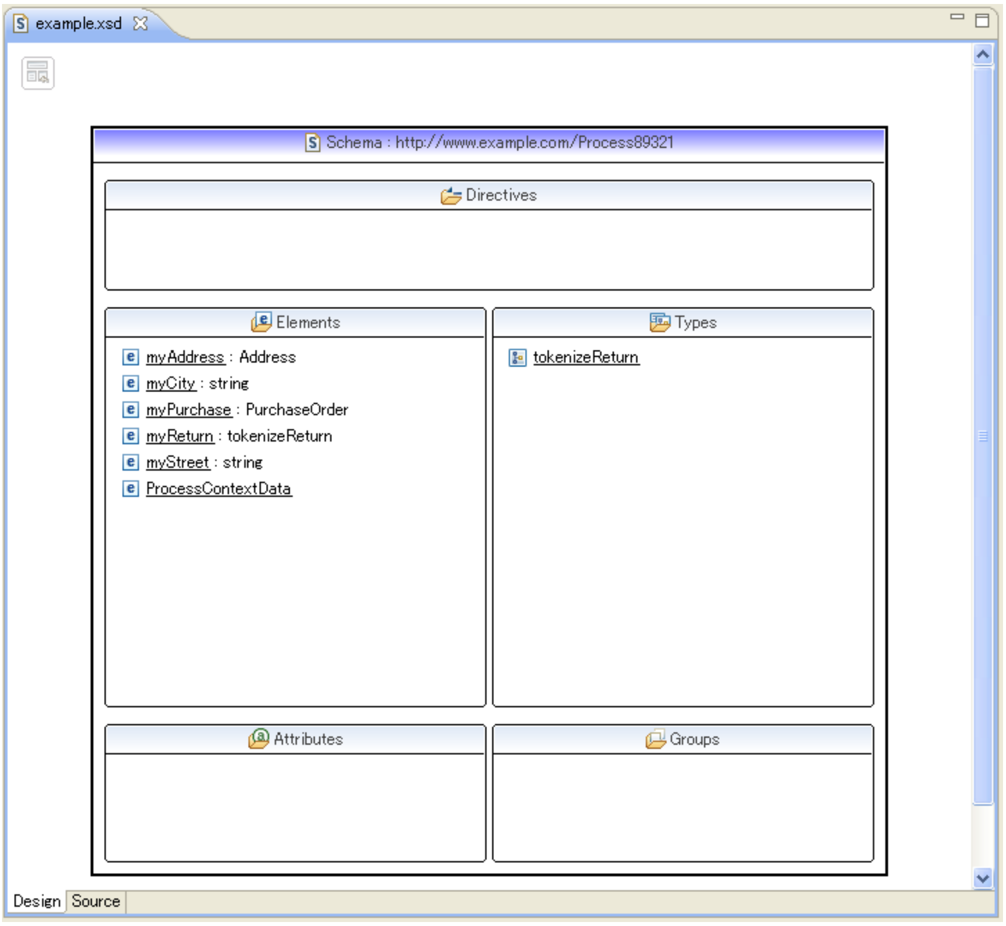
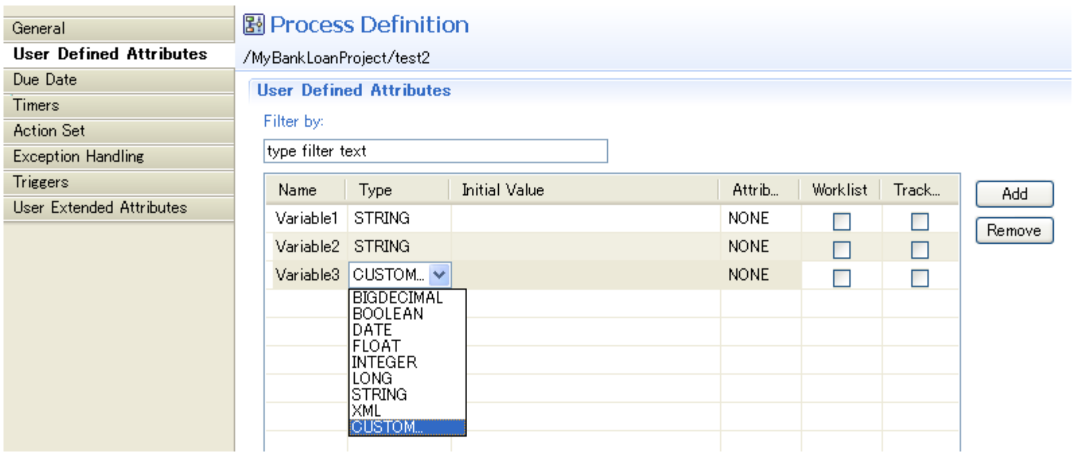


Figure 130: XML Schema

Note:

- You can use only **element** and **type** of XML Schema for a UDA of type CUSTOM. You cannot use any other XML Schema components (example, **group**) for a UDA of type CUSTOM.
- You must set the `elementFormDefault` attribute to **qualified** at the schema element in XML Schema which you would use as the UDA of type CUSTOM.
- If the `elementFormDefault` attribute is set to **qualified**, each of the element values of a UDA of type CUSTOM needs an appropriate namespace.

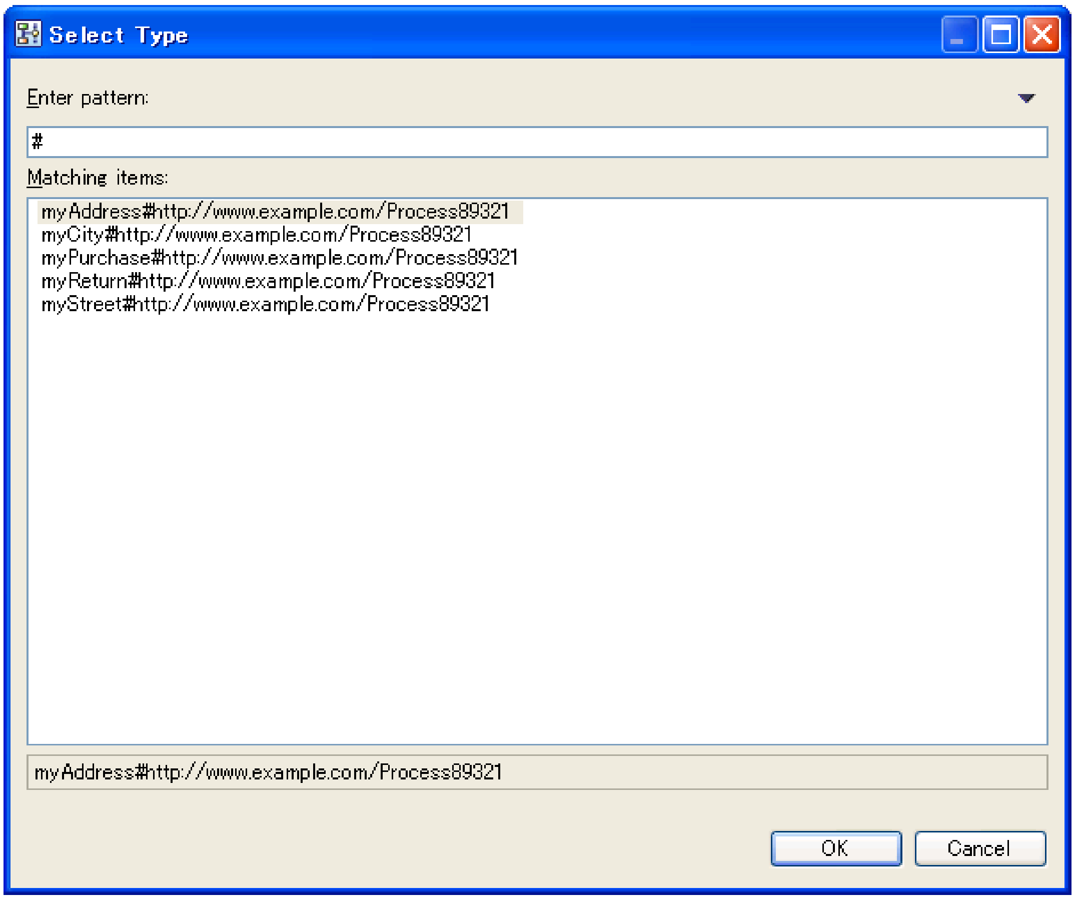
2. Specify the XML Schema to UDA as given below:
   1. Open the process definition.
   2. Click **User Defined Attributes** tab in **Properties** view.
   3. Click the **Add** button to add UDA and select **CUSTOM** from the drop-down list in **Type** column.



Name	Type	Initial Value	Attrib...	Worklist	Track...
Variable1	STRING		NONE	☐	☐
Variable2	STRING		NONE	☐	☐
Variable3	CUSTOM...		NONE	☐	☐

Figure 131: Selecting Type CUSTOM

The **Select Type** dialog is displayed.



Enter pattern:

```
#
```

Matching items:

```
myAddress#http://www.example.com/Process89321
myCity#http://www.example.com/Process89321
myPurchase#http://www.example.com/Process89321
myReturn#http://www.example.com/Process89321
myStreet#http://www.example.com/Process89321
```

```
myAddress#http://www.example.com/Process89321
```

Figure 132: Select Type

4. Select Type or Element in **Select Type** dialogue.
5. Select `Address@http://www.example.com/P01`. For details, refer *Specifying User Defined Attributes of Type CUSTOM* on page 157.

Note: In case you add an attribute element which has `use` attribute set to `required`, to a complex global element, it is not used to define UDAs of type CUSTOM.

9.4 Managing XML Schema File

This section provides information about managing XML Schema files.

9.4.1 Updating XML Schema File

You can update an existing XML Schema file.

To update an XML Schema file:

1. In Navigator view, double-click the XML Schema file in **schema** folder or right-click and select **Open** from the Pop-up menu.

2. Update the XML Schema file.
3. Click **Save** to save the changes.

9.4.2 Renaming XML Schema File

You can rename XML Schema file considering references to it.

To rename an XML Schema file:

1. In Navigator view, right-click the XML Schema file in **schema** folder and select **Rename** from the Pop-up menu.
2. Enter the new name in the **New Name** field.
3. Select the **Update references** checkbox.
4. Click **OK**.

Note:
- If you define target XML Schema file as a UDA of type CUSTOM and click **Details**, the name of its process definition is showed.
- If you select **Update references** and click **OK** and then click **Details**, XML Schema file names which have dependency relation with the target XML Schema, are displayed.
- If you select **Update references** checkbox and rename an XML Schema file, other XML Schema files which are importing it are updated automatically. If you do not select the **Update references** checkbox, you need to update other XML Schema files by yourself.

9.4.3 Deleting XML Schema File

You can delete XML Schema file considering references to it.

To delete an XML Schema file:

1. In Navigator view, right-click the XML Schema file in **schema** folder and select **Delete** from the Pop-up menu.
2. Select the **Delete the dependencies** checkbox.
3. Click **OK**.

Note:
- If you define target XML Schema file as a UDA of type CUSTOM and click **Details**, the name of its process definition is showed. However, if you delete the XML Schema file, UDA is not updated.
- If you select **Delete the dependencies** and click **OK** and then click **Details**, XML Schema file names which have dependency relation with the target XML Schema, are displayed.
- If you select **Delete the dependencies** checkbox and delete an XML Schema file, other XML Schema files which are importing it are deleted automatically. If you do not select the **Delete the dependencies** checkbox, you need to delete other XML Schema files by yourself.

9.4.4 Importing XML Schema File

You can import XML Schema from XSD file or WSDL file, from local file system or remote locations using HTTP protocol (URL).

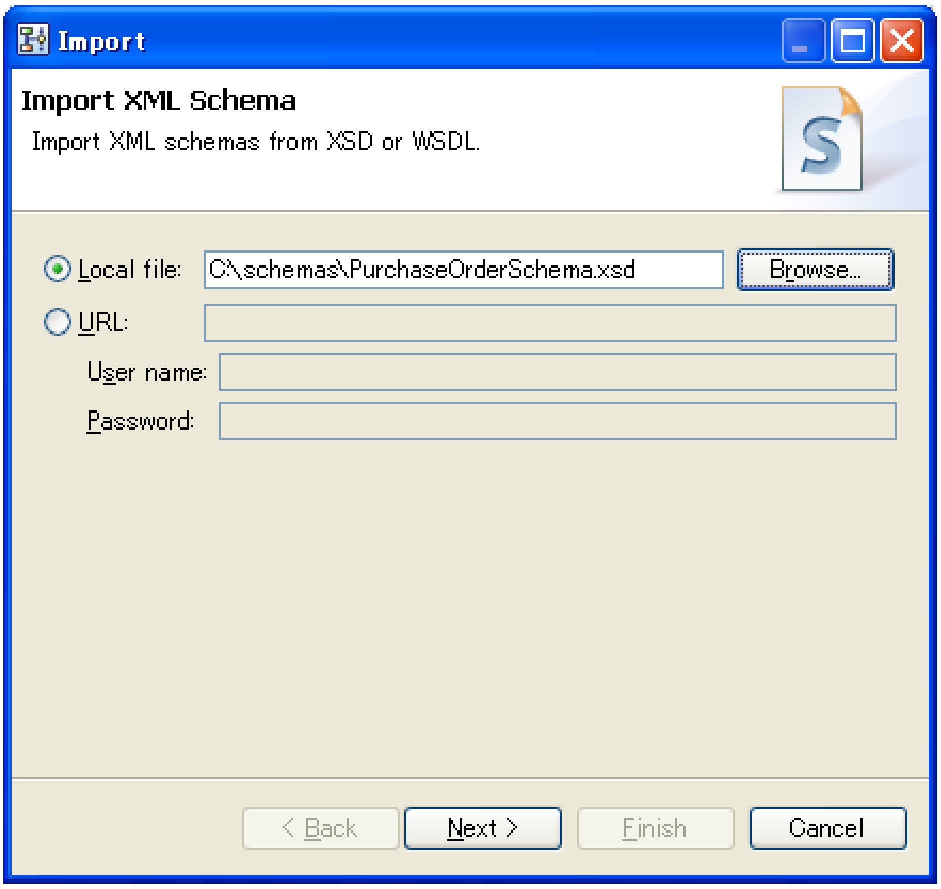
To import an XML Schema file:

1. In Navigator view, right-click the XML Schema file in **schema** folder and select **Import** from the Pop-up menu.

   Import XML Schema dialog is displayed.

2. Select if you want to import the XML Schema file from a **Local file** or from a remote location (**URL**).

   1. If you select **Local file**, enter the file name or click **Browse** and select the XSD or WSDL file.



Import

Import XML Schema

Import XML schemas from XSD or WSDL.

Local file: C:\schemas\PurchaseOrderSchema.xsd

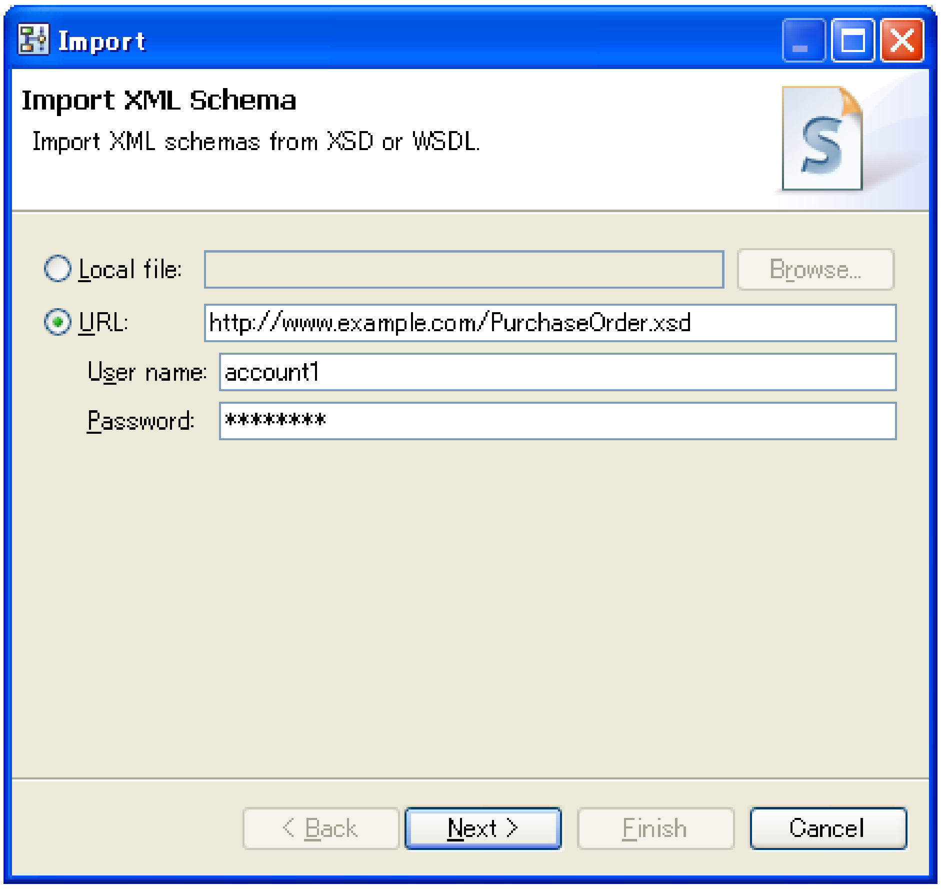
URL:

User name:

Password:

Figure 133: Import from Local File

2. If you select **URL**, enter the URL in the **URL** field and if you need to be identified for this action, enter **User name** and **Password**.



( ) Local file:
(●) URL: http://www.example.com/PurchaseOrder.xsd
User name: account1
Password: ********

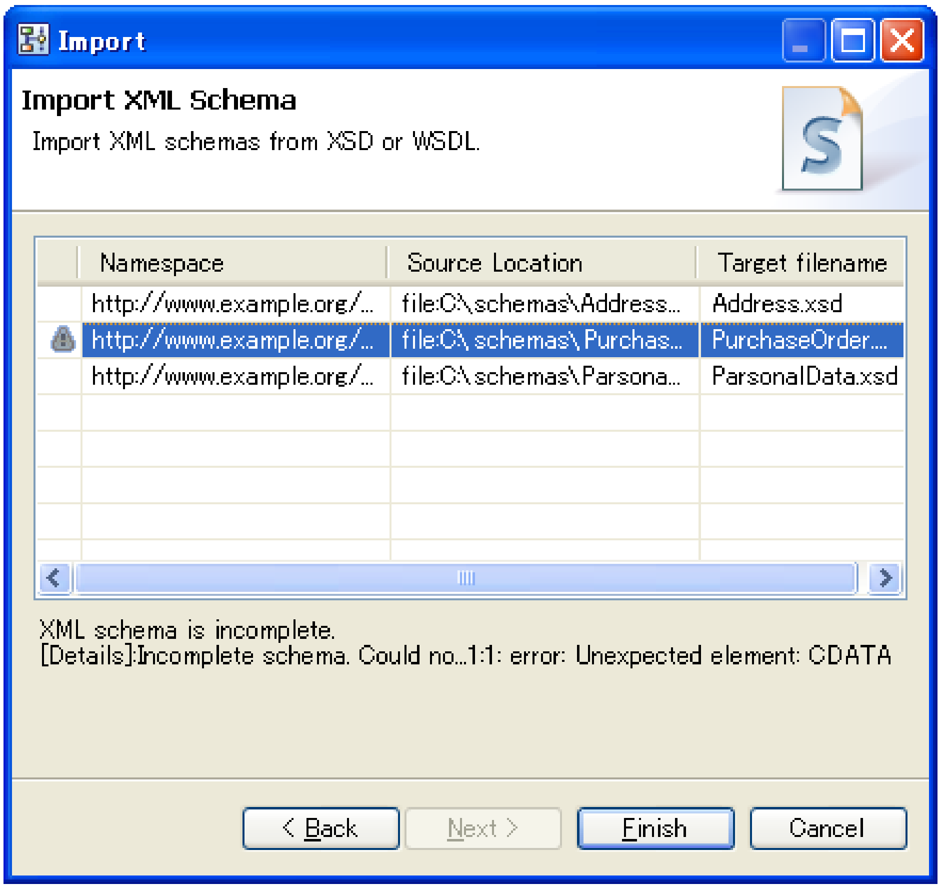
Figure 134: Import from URL

Note: You need not enter **User name** and **Password** after entering the **URL** if you do not wish to be identified for this action.

3. Click **Next**.

   The list of local files or XSD files, which URL's target contains, is displayed. If the XSD file you selected includes/imports another file then related files are imported simultaneously.

Note that if there are no import/include files, an icon is showed at the left of table. When you click the icon, information about the Schema that does not exist, is displayed at the bottom of the dialog.



	Namespace	Source Location	Target filename
	http://www.example.org/...	file:C:\schemas\Address...	Address.xsd
⚠	http://www.example.org/...	file:C:\ schemas\ Purchas...	PurchaseOrder....
	http://www.example.org/...	file:C:\ schemas\ Parsona...	ParsonalData.xsd

XML schema is incomplete.
[Details]:Incomplete schema. Could no...1:1: error: Unexpected element: CDATA

Figure 135: XML Schema Error

4. Click **Finish**. The file name is added in **schema** folder in Navigator view.
5. (Optional) You can change the name of the file directly in the **Target filename** column.

9.4.5 Exporting XML Schema File

You can export an XML Schema file to a file system from Interstage BPM Studio. You can import the file to another file system.

Note: You can export a Workflow Application with all its components to a file. After you export the Workflow Application project to a file, you can deploy this file to Interstage BPM Server. For details, refer *Exporting Workflow Application Projects* on page 100.

To export an XML Schema file:

1. In Navigator view, right-click the XML Schema file in **schema** folder and select **Export** from the Pop-up menu.
2. Select an appropriate location on **Export XML Schema** prompt.

3. Click **Save**.

9.5 Validating XML Schema File

To validate a UDA of type CUSTOM:

1. In Navigator view, select the Workflow Application which you want to validate.
2. Right-click the Workflow Application and select **Validate** from the Pop-up menu.

   The following items are ensured:

   - Elements or Types of XML Shchema are right.
   - Verifying whether the XPath expression is validated against XML Schema.

Note:
- In case of UDA of type XML, XML data and XPath are not validated against XML Schema.
- If an error is detected during validation, error and warning are displayed in Problems view. In Navigator view, error icon is displayed on the **schema** folder and the XML Schema file in which the error has been detected.

10 Modeling Subprocesses, Remote Subprocesses and Chained-Processes

Interstage BPM 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 or on a remote workflow server.

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, remote subprocesses and chained-processes.

10.1 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 *Connecting Parent and Subprocess Definitions* on page 236.

   When the process definitions are connected, you can easily navigate from the parent to the subprocess definition. For instructions, refer to section *Navigating to Subprocess Definitions* on page 237.
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. XML and Custom UDAs can be mapped even if the types are different.

6. Define data mappings for the UDAs that need to be passed back and forth.

   For instructions, refer to section *Defining Data Mappings for Subprocesses* on page 238.

Note: Be careful when designing process definitions that have recursive subprocesses. Check all process definitions involved and make sure that there are no infinite recursions.

10.1.1 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. Make sure 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**.

10.1.2 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 from a parent process definition to a subprocess 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 openend in a separate window, for example:

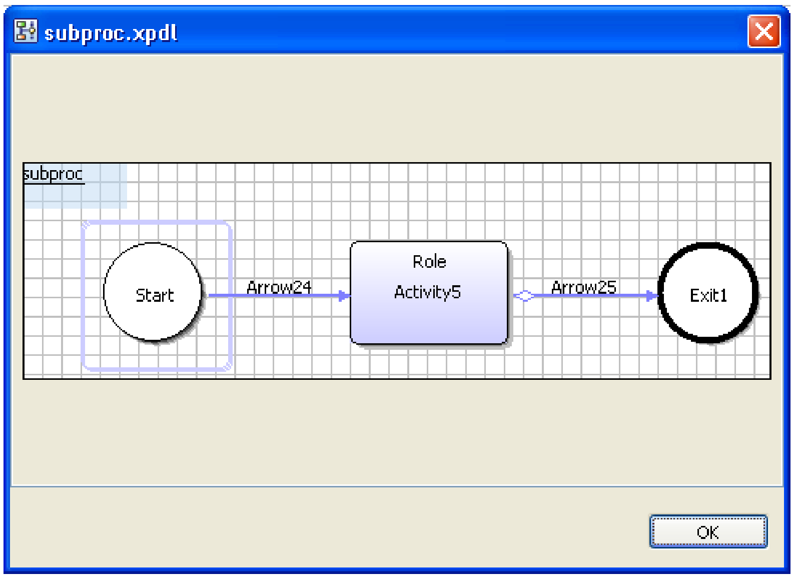


Figure 136: Subprocess Definition

10.1.3 Defining Data Mappings for Subprocesses

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.

Note: Data mapping can be defined between XML UDA and Custom UDA. However, data passing will fail if they have different schemas.

- You have connected the parent process definition and the subprocess definition.

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.

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 or CUSTOM 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 or CUSTOM. 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 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.

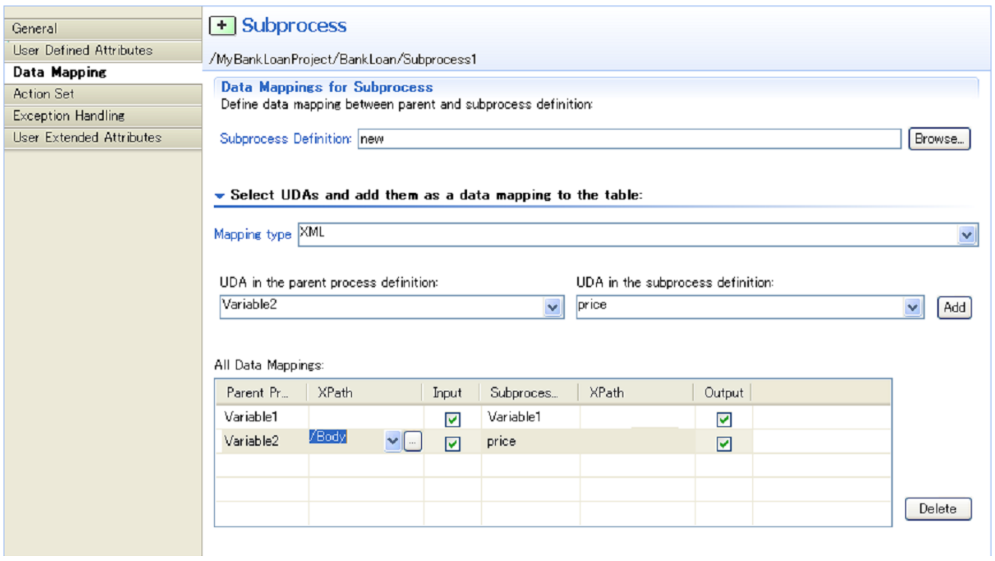
Note: If you specify XPath until non-leaf nodes as an XPath expression, the parent process definition's XML element structure specified with XPath expression must be same as that of the subprocess definition.

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 or CUSTOM 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 or type XML, 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 **Output**.



Subprocess

/MyBankLoanProject/BankLoan/Subprocess1

Data Mappings for Subprocess

Define data mapping between parent and subprocess definition:

Subprocess Definition: new

Select UDAs and add them as a data mapping to the table:

Mapping type: XML

UDA in the parent process definition: Variable2

UDA in the subprocess definition: price

All Data Mappings:

Parent Pr...	XPath	Input	Subproces...	XPath	Output
Variable1		✓	Variable1		✓
Variable2	/Body	✓	price		✓

Figure 137: Defining Data Mappings

10.2 Modeling Remote Subprocesses

Remote subprocesses are subprocesses that run on a remote workflow server. A remote workflow server might be, for example, another Interstage BPM Server.

Interstage BPM supports two open protocols for communication between workflow servers: Simple Workflow Access Protocol (SWAP) and Asynchronous Service Access Protocol (ASAP). These protocols pass XML messages over HTTP between workflow servers.

You are recommended to use ASAP when the processes run on integrated Interstage BPM Servers. Use SWAP with CollaborationRing only. For details about CollaborationRing integration, contact your local Fujitsu Support Organization.

A remote subprocess is represented by a Remote Subprocess Node in the parent process definition. The interaction between parent and remote subprocess comprises the following steps:

1. When a Remote Subprocess Node is reached, the local workflow server sends a `StartProcess` request to the remote workflow server, carrying with it the User Defined Attribute (UDA) values that the remote subprocess instance will work on.
2. The remote workflow server must receive this request, and start a process instance.
3. When the remote process instance completes, its workflow server sends a `ProcessCompleted` message back, carrying with it the results of the subprocess instance.
4. The local workflow server must receive this message, incorporate the results, and complete the Remote Subprocess Node allowing the process instance to continue to the next node.

For a successful interaction, the following must be configured with the parent process definition:

- The URI of the remote process definition
- The names of the UDAs that will be passed back and forth
- Return values of the subprocess definition.

The following instructions assume that the process definitions to be connected will run on Interstage BPM Servers. For details about CollaborationRing integration, contact your local Fujitsu Support Organization.

To model a remote 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 Remote Subprocess Node to the parent process definition.
4. For each result that the subprocess might return, add an outgoing arrow to the Remote Subprocess Node.

   A Remote Subprocess Node may have one or more outgoing arrows. Only one arrow is chosen when the parent process resumes. The arrow that is chosen is the one that matches the result of the remote subprocess.
5. Make sure that the names of the outgoing arrows match exactly the result values that are returned by the remote subprocess.

   The result values correspond to the names of the Exit Nodes in the remote subprocess definition.

Note: The names of the Exit Nodes in the subprocess definition and the names of the outgoing arrows must be identical, also regarding uppercase/lowercase.

6. Connect the parent to the remote subprocess definition.

   For instructions, refer to section *Connecting Parent and Remote Subprocess Definition* on page 241.
7. In both parent and subprocess definitions, specify the 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. XML and Custom UDA can be mapped even if the types are different.

8. Define data mappings for the UDAs that need to be passed back and forth.

   For instructions, refer to section *Defining Data Mappings for Remote Subprocesses* on page 242.

10.2.1 Connecting Parent and Remote Subprocess Definition

In order to connect parent and remote subprocess definition, you need to know the URL of the remote subprocess definition.

The following procedure tells you how to find the URL if you are integrating remote subprocess definitions modeled with Interstage BPM. If you are integrating remote subprocess definitions modeled with other products, refer to the documentation of that product on how to find the URL.

To connect process definitions:

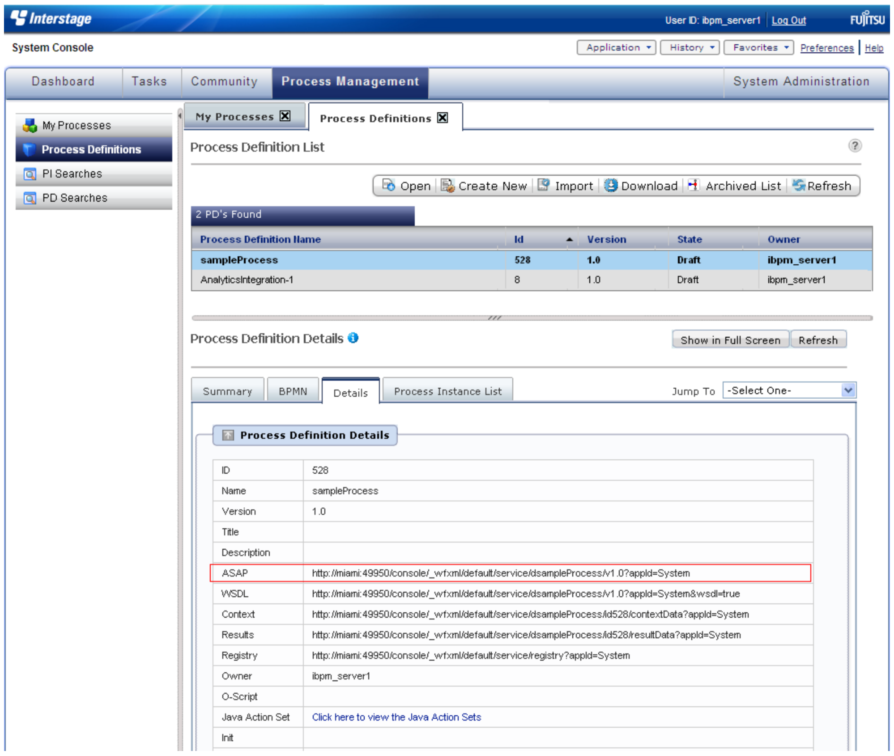
1. Make sure that you know the hostname of the Interstage BPM Server on which the remote subprocess will run.

   Access the Interstage BPM Console on the Server where the definition of the remote subprocess is located and log in to the Console. Refer to the *Interstage Business Process Manager User's Guide* for details.

2. In the Console, select the remote subprocess definition from the Process Definitions list.

   a) Open the remote subprocess definition.

b) In the **Process Definition Details** area, click **Details** tab.



2 PD's Found

Process Definition Name	Id	Version	State	Owner
sampleProcess	528	1.0	Draft	ibpm_server1
AnalyticsIntegration-1	8	1.0	Draft	ibpm_server1

Process Definition Details

ID	528
Name	sampleProcess
Version	1.0
Title	
Description	
ASAP	http://miami:49950/console/_wfxml/default/service/dsampleProcess/v1.0?appId=System
WSDL	http://miami:49950/console/_wfxml/default/service/dsampleProcess/v1.0?appId=System&wsdl=true
Context	http://miami:49950/console/_wfxml/default/service/dsampleProcess/id528/contextData?appId=System
Results	http://miami:49950/console/_wfxml/default/service/dsampleProcess/id528/resultData?appId=System
Registry	http://miami:49950/console/_wfxml/default/service/registry?appId=System
Owner	ibpm_server1
O-Script	
Java Action Set	Click here to view the Java Action Sets
Init	

Figure 138: Finding out the URL - part I

c) The **Details** tab is opened. On this page, you see the URL of the selected process definition in the ASAP field.

d) Copy the URL in the **ASAP** field in order to avoid typos.

3. Open the parent process definition in the Interstage BPM Studio.
4. Make sure that you have added a Remote Subprocess Node to the parent process definition.
5. Select the Remote Subprocess Node to display the Properties view for the node.
6. Select the **Data Mapping** tab.
7. Paste the URL you copied in Step 2d) to **Factory Address of Subprocees Definition** field.
8. Select the communication protocol to be used by the workflow servers on which the parent process and the subprocess will run.

   You are recommended to use ASAP when the processes run on integrated Interstage BPM Servers. Use SWAP with CollaborationRing only. For details about CollaborationRing integration, contact your local Fujitsu Support Organization.

10.2.2 Defining Data Mappings for Remote Subprocesses

Prerequisites:

- You have specified User Defined Attributes (UDAs) that will be passed back and forth in the parent process definition and in the remote subprocess definition.

- When running the process and subprocess on Interstage BPM Servers: The UDAs you want to pass have the same data type in the parent process definition and remote subprocess definition.
- You know the identifiers of the UDAs that are used in the remote subprocess definition.
- You have connected the parent process definition and the remote subprocess definition.

When data is to flow between parent and remote subprocess, you map the UDA of the parent process definition to the corresponding UDA of the remote subprocess definition. Also, you specify in which directions values are to be passed between parent and remote subprocess definition.

To map data between parent and remote subprocess definition:

1. Open the parent process definition.
2. Select the Remote Subprocess Node to display the Properties view for the node.
3. Select the **Data Mapping** tab.
4. To map a UDA:
   a) From the **Mapping Type** drop-down list, select the data type of the UDA you want to map.
   b) Select the UDA in the parent process definition (**UDA in the local process definition**).
   c) Type the identifier (ID) of the corresponding UDA in the remote subprocess definition (**UDA in the remote process definition**).
   d) Click **Add**.
5. If the UDA type is XML or CUSTOM 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 or Custom. The XPath expressions related to the selected UDA are displayed in the **XPath** drop-down list.
6. 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 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.

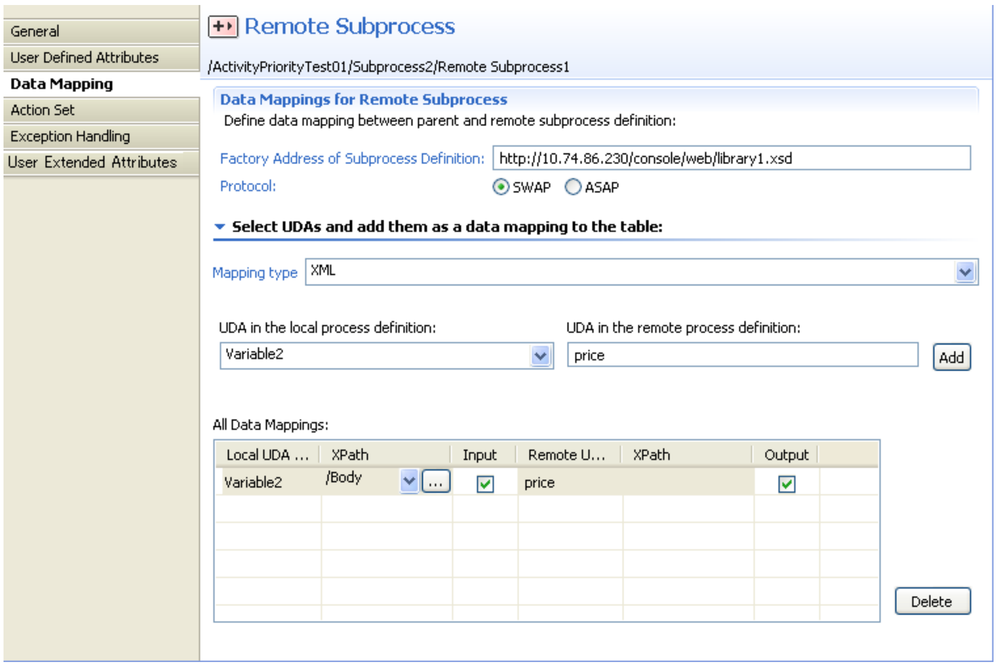
Note: The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note: The XPath expressions related to the selected XML or CUSTOM 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 of type XML, 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.

7. In the **All Data Mappings** area, specify the data flow between process definitions:
   a) When the data value passes from the parent to the remote subprocess, click **Input**.

b) When the data value passes from the remote subprocess to the parent process, click **Output**.



Remote Subprocess

/ActivityPriorityTest01/Subprocess2/Remote Subprocess1

Data Mappings for Remote Subprocess

Define data mapping between parent and remote subprocess definition:

Factory Address of Subprocess Definition: http://10.74.86.230/console/web/library1.xsd

Protocol: [✓ SWAP] ASAP

Select UDAs and add them as a data mapping to the table:

Mapping type: XML

UDA in the local process definition: Variable2

UDA in the remote process definition: price

All Data Mappings:

Local UDA ...	XPath	Input	Remote U...	XPath	Output
Variable2	/Body	✓	price		✓

Figure 139: Defining the Data Mapping

10.3 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 *Connecting Parent and Chained-Process Definitions* on page 245.

   When the process definitions are connected, you can easily navigate from the parent to the chained-process definition. For instructions, refer to section *Navigating to Chained-Process Definitions* on page 245.

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. XML and Custom UDA can be mapped even if the types are different.

6. Define data mappings for the UDAs that need to be passed to the chained-process.

   For instructions, refer to section *Defining Data Mappings for Chained-Processes* on page 245.

10.3.1 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. Make sure 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**.

10.3.2 Navigating to Chained-Process Definitions

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.

You can navigate from a parent process definition to a 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 openend in a separate window.

10.3.3 Defining Data Mappings 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 or CUSTOM 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 or Custom. 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 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.

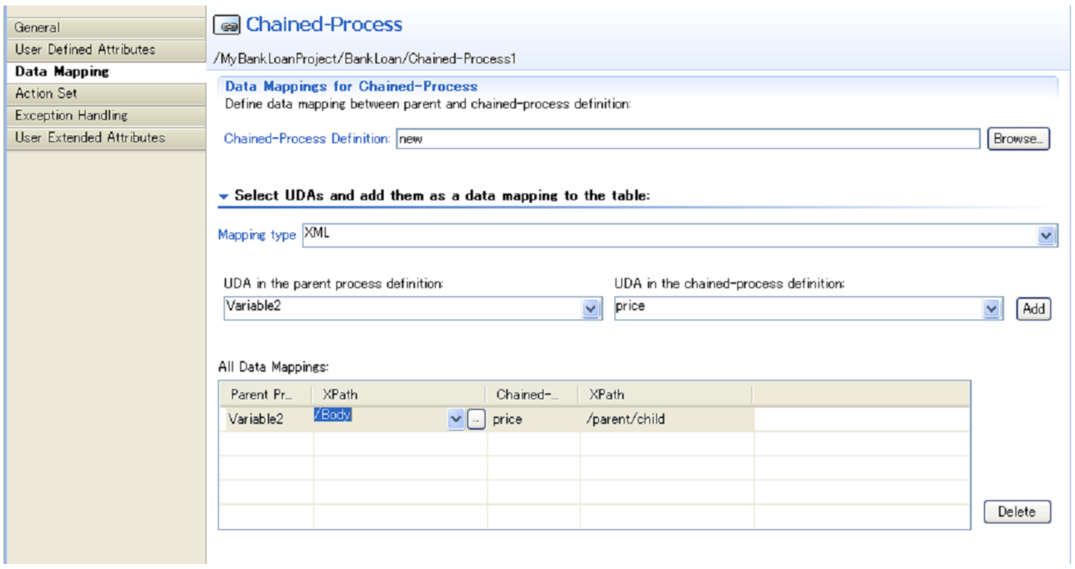
Note: The edited XPath expression is displayed as the latest XPath in the **XPath** drop-down list.

Note: The XPath expressions related to the selected XML or CUSTOM 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 XML 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.



/MyBankLoanProject/BankLoan/Chained-Process1

Data Mappings for Chained-Process

Define data mapping between parent and chained-process definition:

Chained-Process Definition: new

Select UDAs and add them as a data mapping to the table:

Mapping type: XML

UDA in the parent process definition: Variable2

UDA in the chained-process definition: price

All Data Mappings:

Parent Pr...	XPath	Chained-...	XPath
Variable2	/Body	price	/parent/child

Figure 140: Defining Data Mappings

11 Decision Tables

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 Tables avoid the need to learn, develop, and support a complex rules engine infrastructure. Rules engines are very generic and building a usable business application with them can take a long time. Also, applications are often limited because they must use the rules engine's API or access provided by the rules engines.

To add Decision Tables, see *Creating a New Decision Table* on page 250 .

11.1 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 *Creating Workflow Application Projects* on page 85 for instructions.
2. Add Rules Sets to the **Rules** folder. Refer to the section *Creating a New Rule Set* on page 249.
3. Add Decision Table Files to the **Rules Set** folder. Refer to the section*Creating a New Decision Table* on page 250.
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 *Adding or Editing Conditions* on page 253 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 *Adding or Editing Results* on page 255 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 *Adding or Editing Decisions* on page 257 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 *Validating Decision Tables for Errors and Warnings* on page 259 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*Validating Decision Table Rules* on page 260 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 *Assigning Decision Tables to Java Action* on page 262 for instructions. You will map conditions using the **Action Set** tab from the **Activity Node Properties** dialog.

11.2 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.

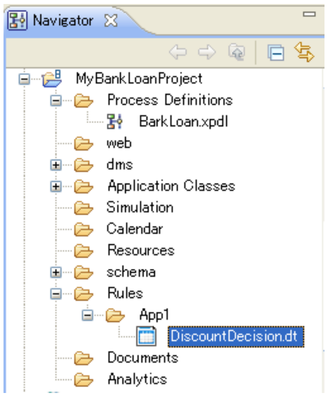


Figure 141: Decision Table Menu

Perform the following steps to create a Decision Table File

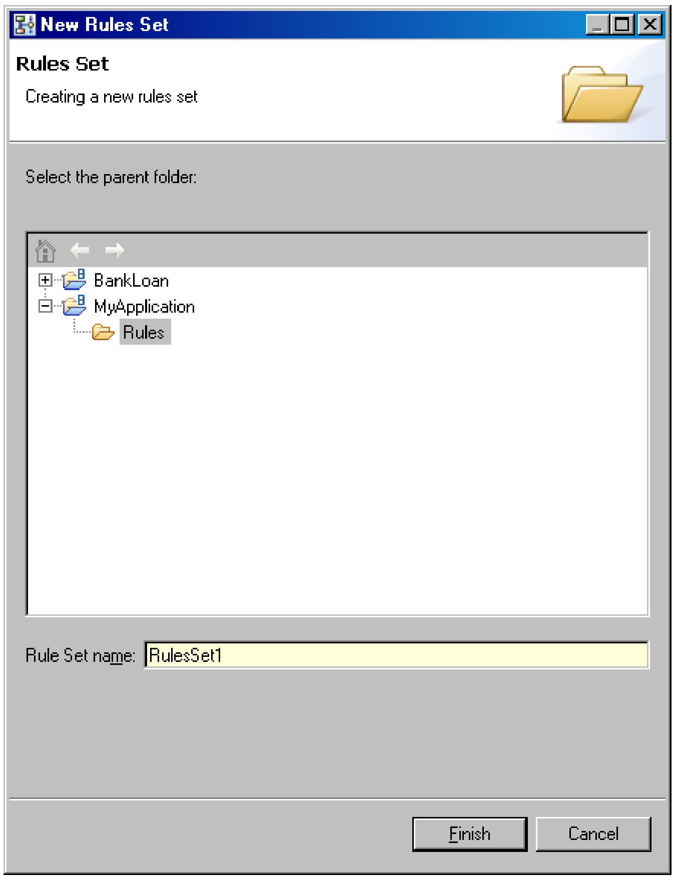
- Create a new application. Refer to *Creating Workflow Application Projects* on page 85
- Create a **Rules Set** folder. Refer to *Creating a New Rule Set* on page 249
- Create a Decision Table inside the **Rules Set** folder. Refer to *Creating a New Decision Table* on page 250

11.2.1 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.



Rule Set name: RulesSet1

Figure 142: New Rules Set Wizard

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.

To add a Decision Table to the **Rules Set** folder, see *Creating a New Decision Table* on page 250.

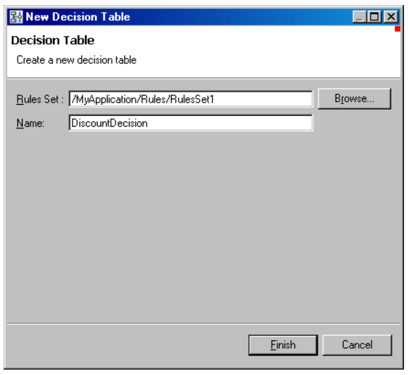
11.2.2 Creating a New Decision Table

Pre-requisites: To create a Decision Table File, you should have already 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.



/MyApplication/Rules/RulesSet1

DiscountDecision

Figure 143: New Decision Wizard

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.

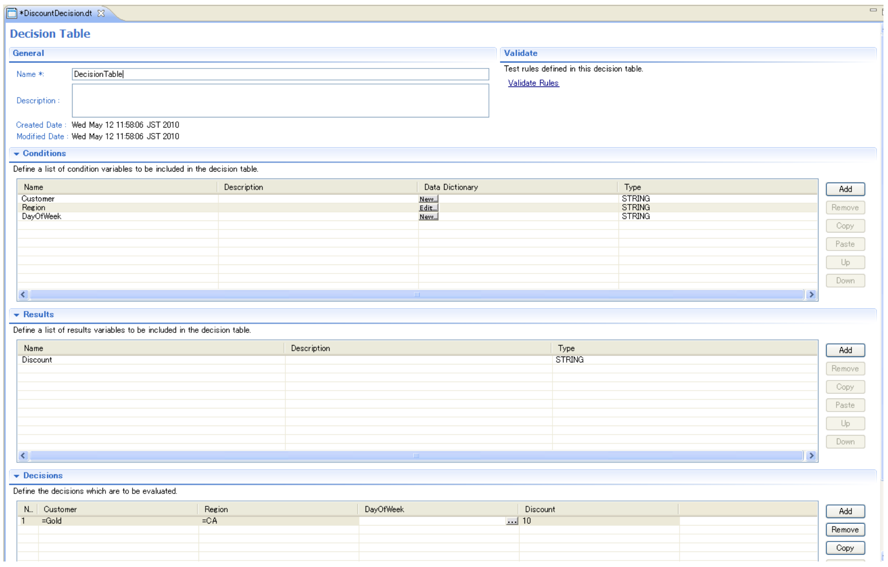
Note: Even if you create a new Decision Table, it might not be complete due to some errors. These errors are displayed in the **Problems** view.

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 but recommended).

Note: The decision table should be created with .dt extension.

Note: 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`.



Name *: DecisionTable

Description :

Created Date : Wed May 12 11:58:06 JST 2010

Modified Date : Wed May 12 11:58:06 JST 2010

Test rules defined in this decision table.

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
Customer		New...	STRING
Region		Edit...	STRING
DayOfWeek		New...	STRING

Define a list of results variables to be included in the decision table.

Name	Description	Type
Discount		STRING

Define the decisions which are to be evaluated.

N...	Customer	Region	DayOfWeek	Discount
1	=Gold	=CA		10

Figure 144: Creating a Decision Table

4. Click **Save** in the toolbar.

   Your new Decision Table is saved. To know more about adding and editing Decision Tables, refer to *Editing a Decision Table* on page 253.

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* on page 252 and *Importing Decision Table Files* on page 252 and *Creating Workflow Application Projects* on page 85 sections for more information.

Exporting Decision Table Files

You can export saved Decision Tables to an external file to back them up or archive them. Right click on the Decision Table file in the Navigator view of Studio, and select **Export**. You can also go to **File** and select **Export**. Save the file to the local file system.

Importing Decision Table Files

Before you can use these instructions effectively, you must have a Decision Tables file (`*.dt`) from an Interstage BPM installation containing Decision Tables that you want to import. You will also need to know the location of this file.

To import Decision Tables from another Interstage BPM installation:

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. Select the file and click **Open**.

3. The Decision Table file that you imported is displayed in the corresponding **Rules Set** folder, in the Navigation view.

11.3 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.

- *Adding or Editing Conditions* on page 253
- *Adding or Editing Results* on page 255
- *Adding or Editing Decisions* on page 257

11.3.1 Adding or Editing Conditions

Pre-requisites: To use the instructions in this section you must have already created a Decision Table. See *Creating a New Decision Table* on page 250 for information on creating a Decision Table file.

A condition is an individual data variable defined in the Conditions list.

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.

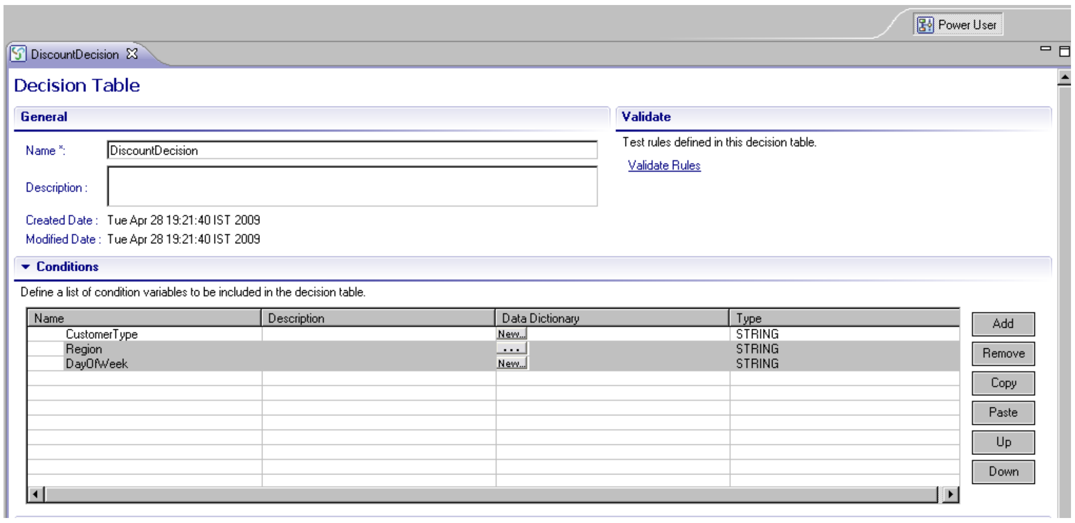
   The Decision Table Editor is displayed.

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* on page 254 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, DayOfWeek of data type `STRING`.



Decision Table

General

Name *: DiscountDecision

Description :

Created Date : Tue Apr 28 19:21:40 IST 2009

Modified Date : Tue Apr 28 19:21:40 IST 2009

Validate

Test rules defined in this decision table.

Conditions

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
CustomerType		New...	STRING
Region		...	STRING
DayOfWeek		New...	STRING

Figure 145: Adding a Condition

8. To remove a condition, select it in the **Conditions** list and click **Remove** .

   The condition is removed from the Decision Table Conditions List.
9. To copy the condition, select it in the **Conditions** list and click **Copy**.

   The condition is copied.
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 `Cntrl` keys and selecting multiple rows.

You can now add results to your Decision Table. See *Adding or Editing Results* on page 255 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, and the system will automatically use the appropriate values. This makes the system more flexible.

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.

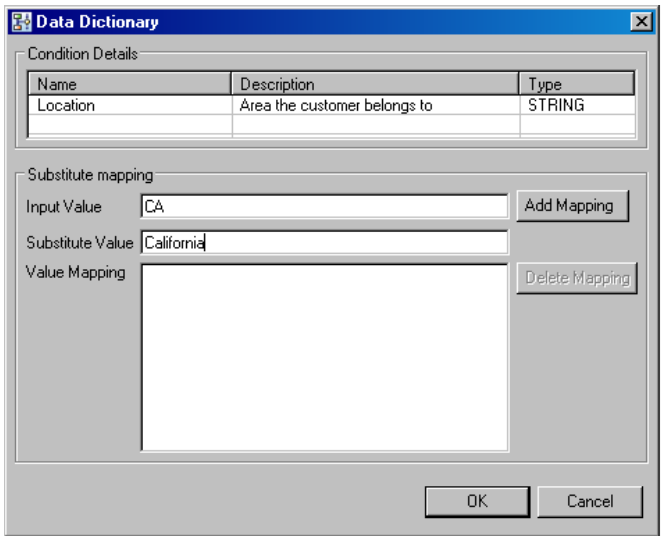
2. Enter the UDA value in the **Input Value** field.
3. Enter the value that you want to substitute for your UDA Value in the **Substitute value** field.
4. Click **Add Mapping**.

   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.



Name	Description	Type
Location	Area the customer belongs to	STRING

Input Value: CA
Substitute Value: California

Figure 146: Substitute Mapping CA to California

11.3.2 Adding or Editing Results

Pre-requisites: To use the instructions in this section you must have created a Decision Table. See *Creating a New Decision Table* on page 250 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.

A result is a pre-defined action. Use the **Result** section to add and store results. After a result is mapped to a condition parameter in the **Decision** section, if the condition specified in the Decision expression is met, then the result action is executed.

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.

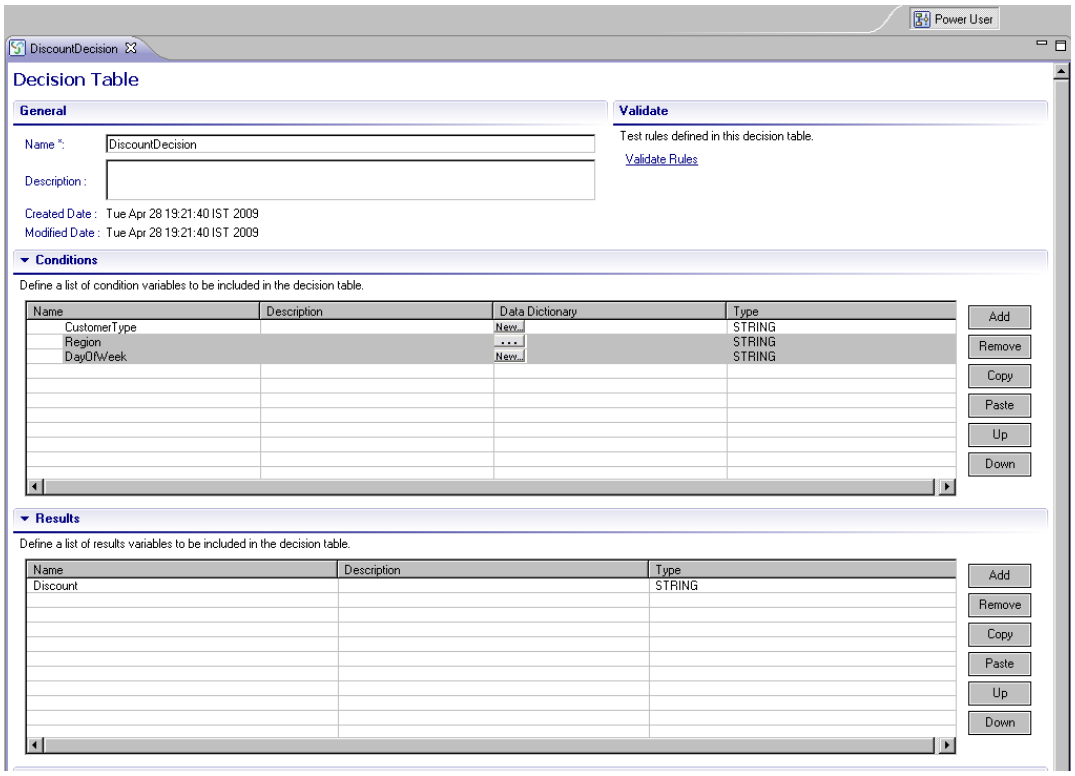
   The **Decision Table Editor** is displayed.

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`.



Decision Table

General

Name *: DiscountDecision

Description :

Created Date : Tue Apr 28 19:21:40 IST 2009

Modified Date : Tue Apr 28 19:21:40 IST 2009

Validate

Test rules defined in this decision table.

Conditions

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
CustomerType		New...	STRING
Region		...	STRING
DayOfWeek		New...	STRING

Results

Define a list of results variables to be included in the decision table.

Name	Description	Type
Discount		STRING

Figure 147: Adding a Result

6. To remove a result, select it in the **Results** list and click **Remove** .

   The result is removed from the **Results** list.

7. To copy the result, select it in the **Results** list and click **Copy**.

   The result gets copied.

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 `Cntrl` keys and selecting multiple rows.

You can now add decisions to your Decision Table. See *Adding or Editing Decisions* on page 257 for more information.

11.3.3 Adding or Editing Decisions

Pre-requisites: Before using the instructions in this section, you must have added conditions and results to your decision table. See *Adding or Editing Conditions* on page 253 and *Adding or Editing Results* on page 255.

To add or edit a decision in 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.

   The **Decision Table Editor** is displayed.

2. To add a new decision to the Decision Table, in the **Decision** section of the **Decision Table Editor**, click **Add**.

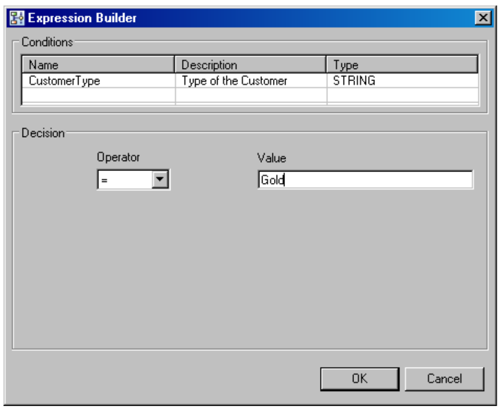
   A new row will be added with incremental serial number in the **No.** column of the Decisions table. If this is the first decision you are adding to the Decision Table the numeral 1, is displayed in the **No.** column.

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.

Note: The various comparison operators that are supported are listed below.

5. Select the operation that you want to apply for the conditon from the **Operator** drop-down.



Name	Description	Type
CustomerType	Type of the Customer	STRING

Operator: =
Value: Gold

Figure 148: Expression Builder

The Expression Buider 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.

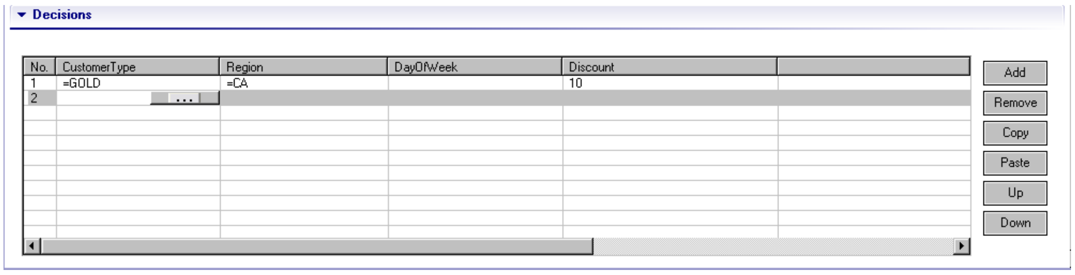
Note: For the operators `=,!=,<,>,<=,>=` , the format is operator followed by the value. For example, `=xyz`. For operators `in,like,not like, between` the format is operator(value1,value2...). For example, `in(a,b,c)`.

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.



No.	CustomerType	Region	DayOfWeek	Discount	
1	=GOLD	=CA		10	
2					

Figure 149: Adding a Decision

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.

At run time, Interstage BPM evaluates all of the defined decisions in the order shown on the Decision Table Details page. When defining decisions, make sure 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 *Validating Decision Table Rules* on page 260.

To know about mapping UDAs, conditions and results to a Process Definition using JavaAction, see *Assigning Decision Tables to Java Action* on page 262.

11.4 Validating Decision Tables for Errors and Warnings

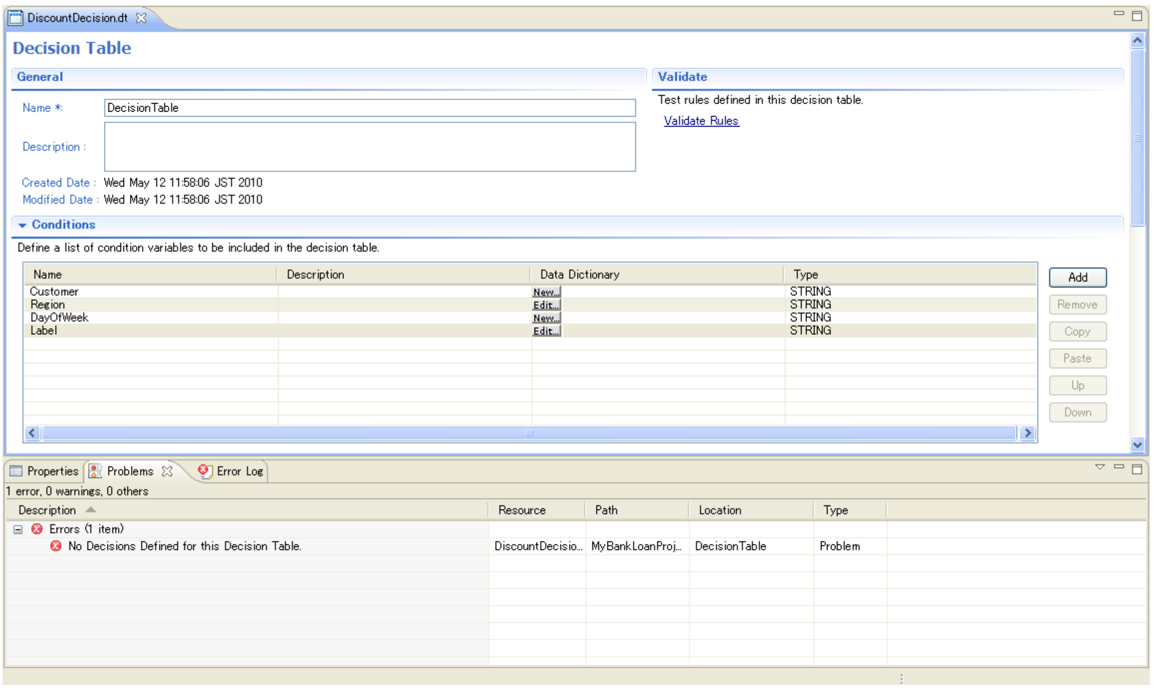
Decision Tables need to be validated for any errors or warnings. Decision Tables do not test successfully if errors exist in them. Decision Table files can be validated by using any of the following methods:

- **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 **Problems View** tab below the Decision Table editor.
- **Using the Validate Rules link**: Click this link to validate the Decision Table. Refer to *Validating Decision Table Rules* on page 260 for more information about **Validate Rules** link. The **Test Decision Tables** dialog 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 **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 **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 *Problems View* on page 66 for more information about **Problems View**. The figure below displays the **Problems View** and the errors and warnings in the Decision Table.



Name *: DecisionTable

Description :

Created Date : Wed May 12 11:58:06 JST 2010
Modified Date : Wed May 12 11:58:06 JST 2010

Test rules defined in this decision table.

Validate Rules

Define a list of condition variables to be included in the decision table.

Name	Description	Data Dictionary	Type
Customer		New...	STRING
Region		Edit...	STRING
DayOfWeek		New...	STRING
Label		Edit...	STRING

1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
No Decisions Defined for this Decision Table.	DiscountDecisio...	MyBankLoanProj...	DecisionTable	Problem

Figure 150: Validating Decision Table File

11.5 Validating Decision Table Rules

Pre-requisites: You must have a complete Decision Table to test before using these instructions. At a minimum, you must map conditions for your Decision Table before testing it.

Interstage BPM provides a means by which you can validate Decision Tables that you have created. Upon clicking the **Validate Rules** link, the Decision Table is validated for errors and warnings, if any, in the Decision Table. If errors exist in the Decision Table then the **Test Decision Tables** dialog does not open. If only warnings exist in the Decision Table then the **Test Decision Tables** dialog opens. After validating the Decision Table, the errors and warnings in the Decision Table are displayed in **Problems View** tab. Refer to *Validating Decision Tables for Errors and Warnings* on page 259 for validation methods of the Decision Tables and *Problems View* on page 66 for **Problems View**.

Using the **Test Decision Tables** dialog, you can test a Decision Table and validate the result.

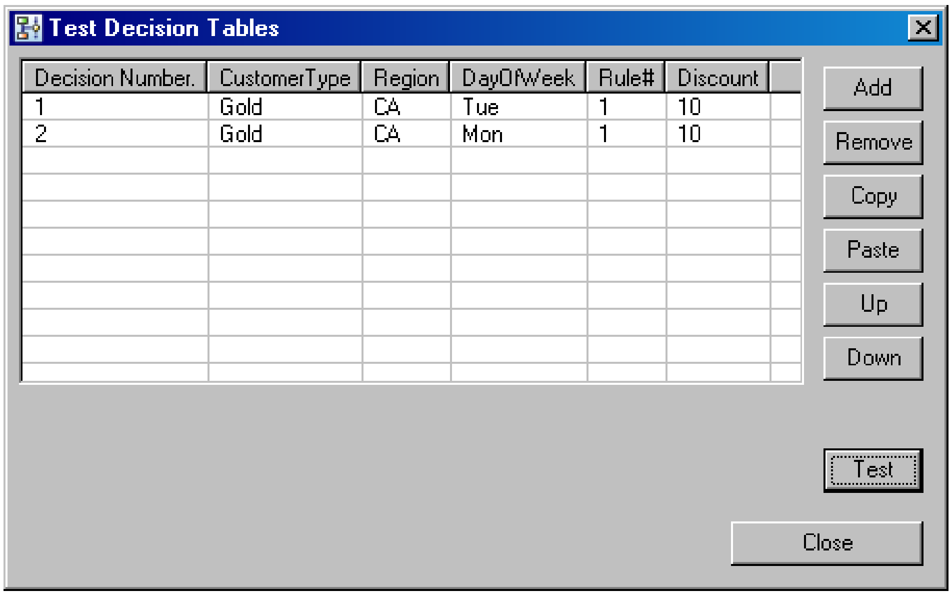
To test a Decision Table:

1. Make sure 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 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**.

   Test results are generated on the Decision Table Test page. Complete audit information is provided. With this information, you can easily see which decision is matched along with its result. The Decision Table is also provided on this page for quick reference. The test data will be saved after each test.

   The figure below displays the results of a test on the DiscountDecision Decision Table.



Decision Number.	CustomerType	Region	DayOfWeek	Rule#	Discount
1	Gold	CA	Tue	1	10
2	Gold	CA	Mon	1	10

Figure 151: Results of Test on DiscountDecisionDecision Table

11.6 Using the Decision Table Action

When you install Interstage BPM, a pre-built Java Action is included to facilitate the use of Decision Tables. This section includes instructions for selecting process definitions that will work with a Decision Table, creating a Decision Table Action to contain your Decision Table, adding the Decision Table Action, and using the process that contains the Decision Table Action.

11.6.1 Selecting a Process Definition to Contain Your Decision Table

A Decision Table is useful only if it is run as a component of an Interstage BPM 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

See*Assigning Decision Tables to Java Action* on page 262 for information on assigning Decision Table to a Java Action.

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.

11.6.2 Assigning Decision Tables to Java Action

Prerequisites:

- You have defined your conditions, results, and decisions in a Decision Table. See *Decision Tables* on page 248 section for more information on creating Decision Tables.
- The process definition contains User Defined Attributes (UDAs).

You can integrate Decision Tables into your process definition using the Decision Tables Java Action.

To integrate a Decision Table:

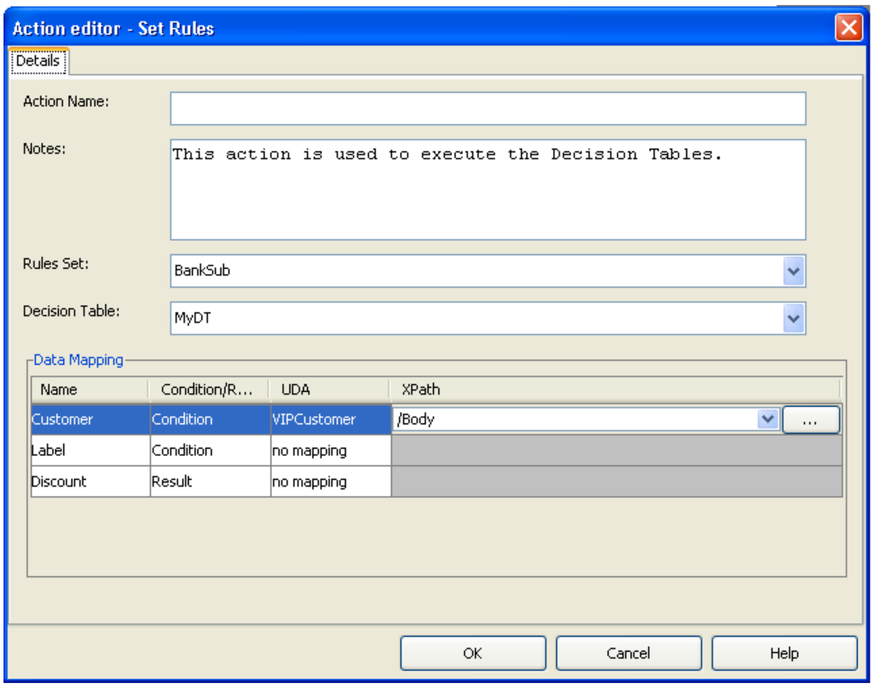
1. Open the process definition that will contain your Decision Table by double clicking the `.XPDL` from the **Process Definitions** folder.

   The **Process Definition Editor** is displayed.
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.



Action Name:

Notes: This action is used to execute the Decision Tables.

Rules Set: BankSub

Decision Table: MyDT

Data Mapping

Name	Condition/R...	UDA	XPath
Customer	Condition	VIPCustomer	/Body
Label	Condition	no mapping	
Discount	Result	no mapping	

Figure 152: Integrating Decision Tables

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.

Note: 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.

Note: 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 not 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 or CUSTOM 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.

Note: The XPath expressions related to the selected XML or CUSTOM 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 of type XML, 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.

Note: **XPath Editor** only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

Note: If a Custom UDA is selected, XPath expressions specifying only leaf nodes (such as text or attribute nodes) will be shown in the **XPath** drop-down list.

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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12 Advanced Process Modeling

This chapter explains how to use Java Actions and triggers. It also provides information about how to define JavaScript expressions.

12.1 Using Java Actions

Using Java Actions, you can customize process execution and integrate with external systems.

Interstage BPM Studio provides several types of built-in Java Actions: Notification Actions, Server Actions, Database Actions, Rules Actions, XML Actions, and Integration Actions. It also provides a mechanism to integrate methods of your own Java classes as Generic Actions. In addition, Interstage BPM Studio provides 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.

12.1.1 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.
- **Prologue Actions**. 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.
- **Epilogue Actions**. 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.
- **Role Actions**. 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.
- **Error Actions**. 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.
- **Compensation Actions**. A Compensation Action can be defined for another Java Action that accesses a system outside of Interstage BPM, 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**

on* 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.

12.1.2 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. For example, a Web Service Node can only have Web Service Actions assigned to it and they can be assigned as Epilogue Actions only.
- **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, Interstage BPM 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 *Dealing With Errors in Java Actions* on page 271.

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 contents 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.

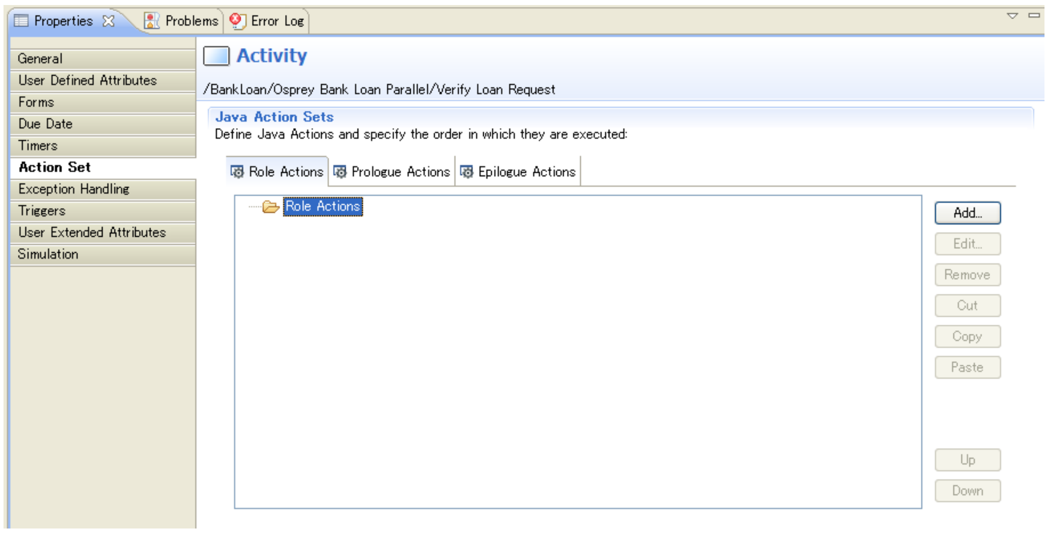


Figure 153: Action Set Tab for Activity Nodes

The following figure shows the **Action Set** tab for a process definition.

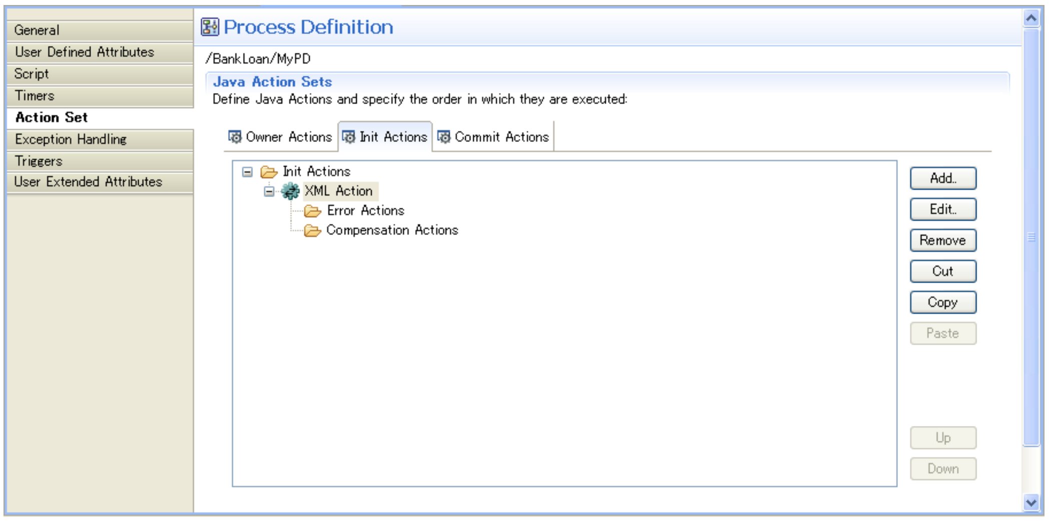


Figure 154: Action Set Tab for Process Definitions

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 *Defining Due Dates* on page 169 and *Defining Timers* on page 171.

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.

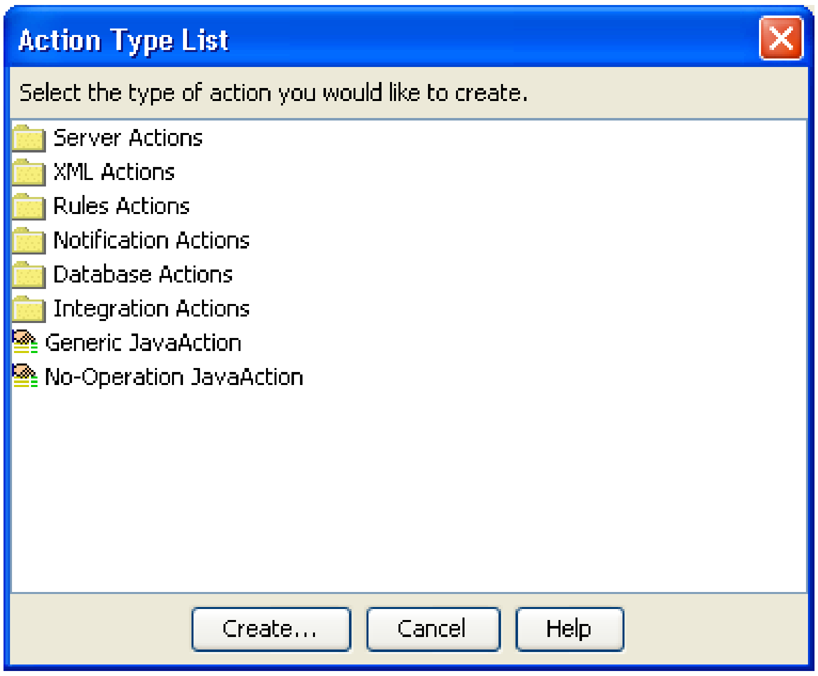


Figure 155: Action Type List

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**.

   A dialog is displayed where you fill in the details for the selected Java Action. You find detailed explanations of the particular Java Actions in the subsequent sections of this chapter, where all available Java Actions are described.

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.

   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 *Cutting, Copying and Pasting Java Actions* on page 269).

12.1.3 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.

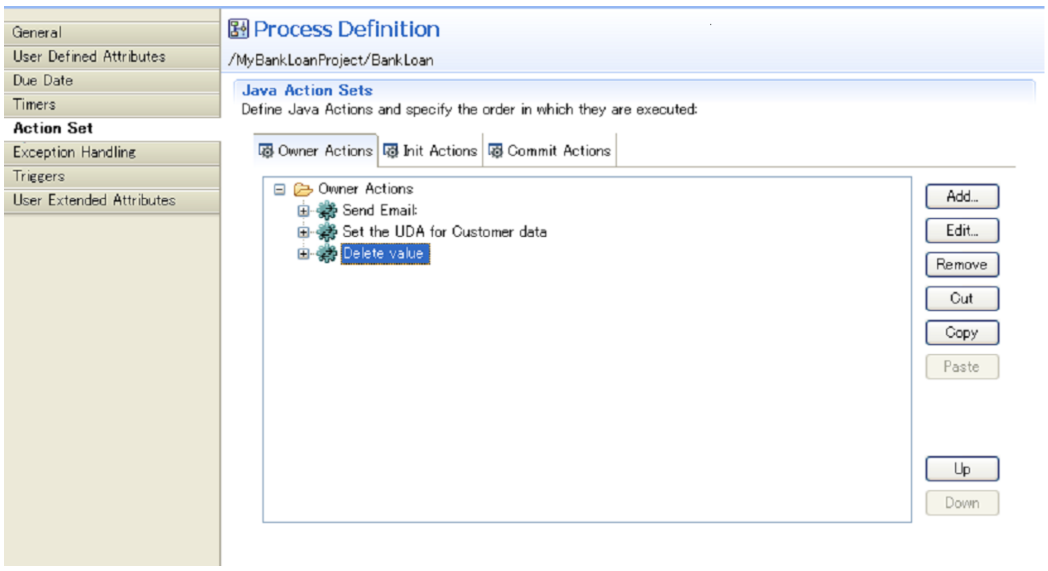


Figure 156: Displaying the Owner Action Set

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. You find detailed explanations of the particular Java Actions in the subsequent sections of this chapter.

4. Click **OK** to close the dialogs.

12.1.4 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 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 *Dealing With Errors in Java Actions* on page 271). All other settings of the copied Error Action (e.g. name, description, WSDL of a web service, etc.) 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 *Dealing With Errors in Java Actions* on page 271). All other settings of the copied Error Action (e.g. name, description, WSDL of a web service, etc.) 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.

12.1.5 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**.

12.1.6 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 Interstage BPM 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 Ations**: 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. Make sure that the **Commit Transaction after completion** check box is selected for that node.

12.1.7 Dealing With Errors in Java Actions

When an error occurs during the execution of a Java Action, an exception is thrown. Interstage BPM 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.

Interstage BPM 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 Interstage BPM 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.

  Refer *Error Handling Sample* on page 276 for more information.

- **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 Remote Subprocess Node or 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:

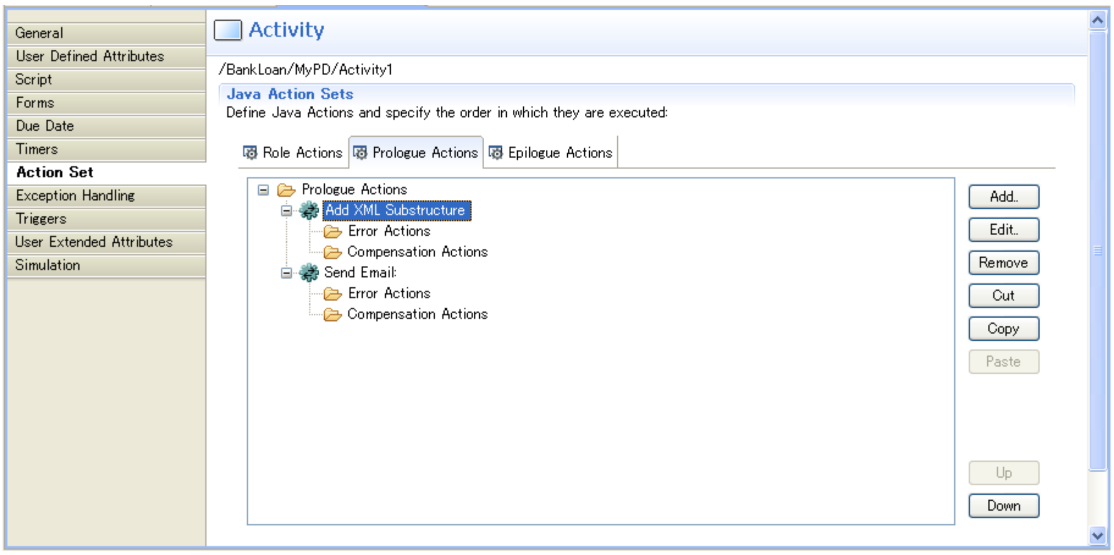


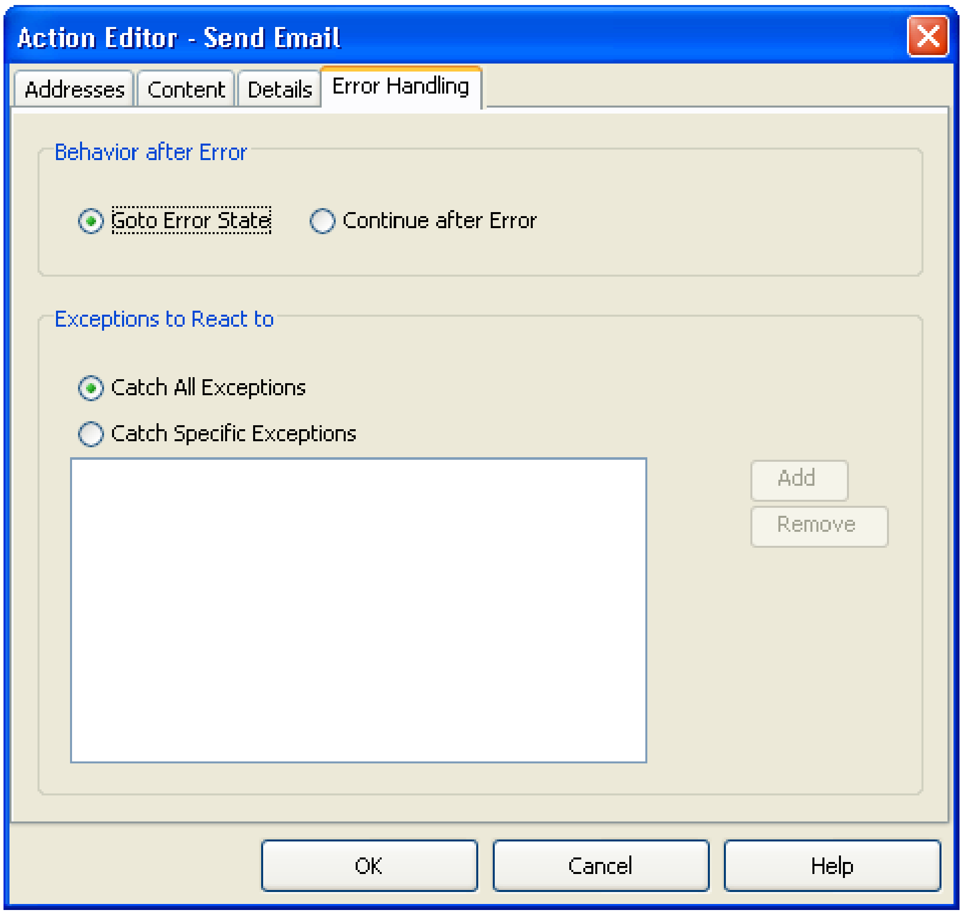
Figure 157: Properties view with Prologue Actions

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. You find detailed explanations of the particular Java Actions in the subsequent sections of this chapter, where all available Java Actions are described.
6. When defining an Error Action for a Remote Subprocess Node or 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 for a Remote Subprocess Node or a regular Java Action. On process definition level and when defining Compensation Actions, this tab is not available.



Behavior after Error: [✓ Goto Error State] Continue after Error

Exceptions to React to: [✓ Catch All Exceptions] Catch Specific Exceptions

Figure 158: Displaying the Error Handling Tab

- **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 Add XML Substructure Prologue Action:

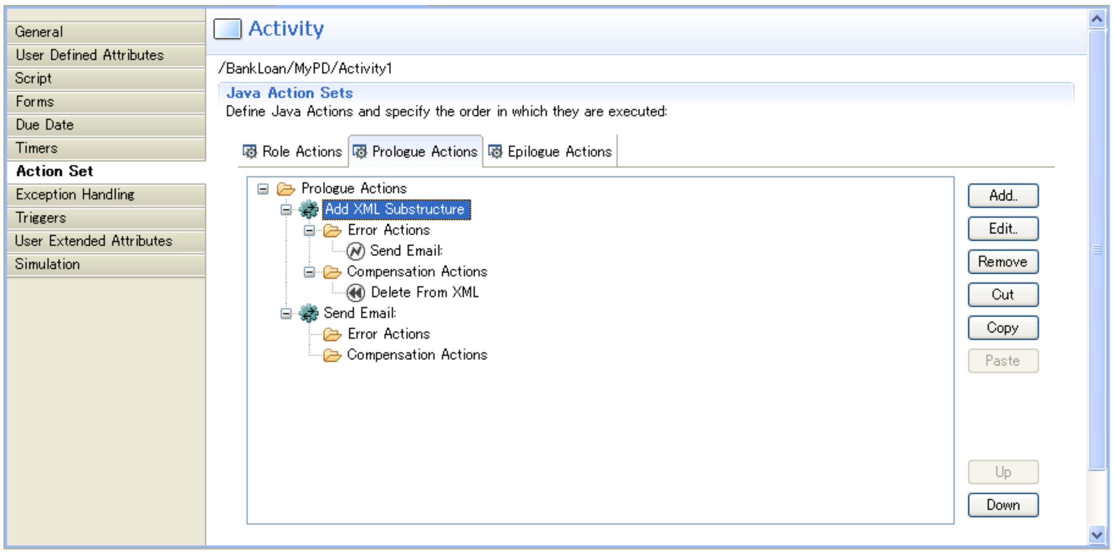


Figure 159: Adding Error and Compensation Actions

12.1.8 Error Handling Sample

The following examples demonstrate the execution of regular Java Actions in conjunction with Error Actions and Compensation Actions.

Using Error Actions

This example shows how to use regular Java Actions with Error Actions.

You are planning to execute the following Prologue Actions:

1. Sending an email message
2. Transferring a file to a remote location using FTP
3. Inserting a row to an external database

For these Prologue Actions, you create the following Error Actions for an Activity Node:

Prologue Actions (PA)	**Error Actions (EA) and Settings**
Send email (PA1)	No-Operation Java Action (EA1): **Behavior after Error** setting: **Continue after Error** **Exceptions to React to** setting: **Catch Specific Exceptions**
Transfer file to remote location using FTP (PA2)	Notification Actions -> SendEmail (EA2): **Behavior after Error** setting: **Goto Error State** **Exceptions to React to** setting: **Catch Specific Exceptions**

Prologue Actions (PA)	Error Actions (EA) and Settings
Insert row in external database (PA3)	Error Action for connection errors, for example writing a log file (EA3.1): **Behavior after Error** setting: **Goto Error State** **Exceptions to React to** setting: **Catch Specific Exceptions** Error Action for duplication errors (database entry is already available), for example sending a notification email (EA3.2) **Behavior after Error** setting: **Continue after Error** **Exceptions to React to** setting: **Catch Specific Exceptions**

During execution, Java Actions may or may not throw an exception. If they throw an exception, the following actions can be executed:

- **Send email (PA1) fails to execute**: A No-Operation Action has been defined for this error case. If PA1 throws one of the specified exceptions, the No-Operation Java Action is executed. Nothing happens because this action specifies no operation. After that, the process instance continues to be executed.
- **Transfer file to remote location (PA2) fails to execute**: A Notification Action has been defined for this error case. If PA2 throws one of the specified exceptions, the Notification Action is executed, for example a notification email is sent. After that, all internal processes are rolled back, and the process instance goes to error state.
- **Insert row in external database (PA3) fails to execute**: Two Java Actions have been defined for this error case. If PA3 throws a connection error exception, Error Action EA3.1 will be executed. EA3.1 reports the error, all internal processes are rolled back, and the process instance goes to error state. If PA3 throws an exception because the same database entry is already available, EA3.2 catches the exception and sends a notification email, and the process instance continues to be executed.

Note: If PA1 or PA2 or PA3 throws an exception which is not defined in the **Exceptions to React to** setting of corresponding Error Actions, no Error Actions are executed, all internal processes are rolled back, and the process instance goes to error state.

The following figure shows the execution of regular Java Actions and Error Actions:

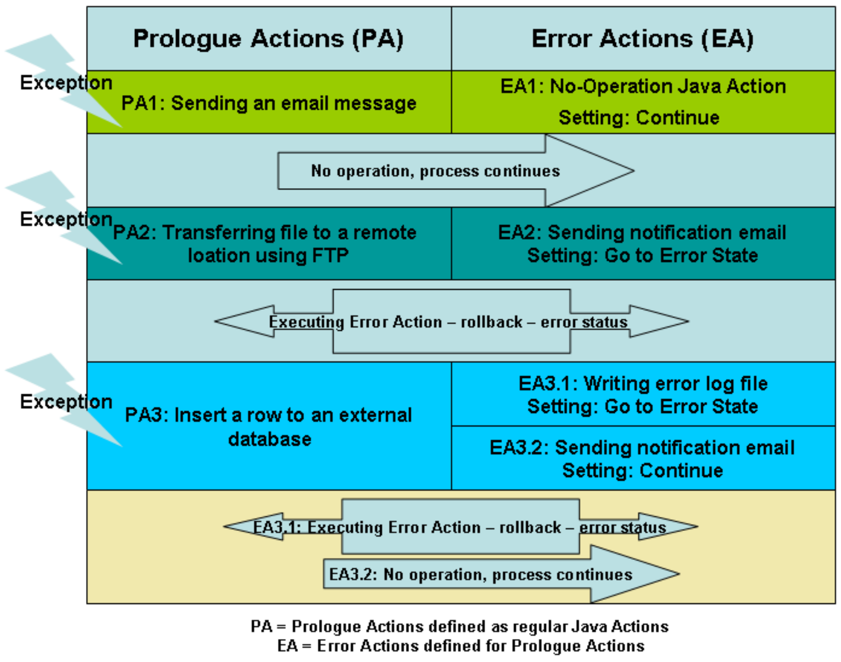

Prologue Actions (PA)
Error Actions (EA)
Exception
PA1: Sending an email message
EA1: No-Operation Java Action
Setting: Continue
No operation, process continues
Exception
PA2: Transferring file to a remote loation using FTP
EA2: Sending notification email
Setting: Go to Error State
Executing Error Action – rollback – error status
Exception
PA3: Insert a row to an external database
EA3.1: Writing error log file
Setting: Go to Error State
EA3.2: Sending notification email
Setting: Continue
EA3.1: Executing Error Action – rollback – error status
EA3.2: No operation, process continues
PA = Prologue Actions defined as regular Java Actions
EA = Error Actions defined for Prologue Actions

Figure 160: Executing Java Actions and Error Actions

Using Compensation Actions

This example shows how Compensation Actions function with regular Java Actions.

Compensation Actions are not executed in the normal path of execution. They are executed only when the actions before them were successful, but the transaction was rolled back. Then, in a separate transaction, Compensation Actions are executed in reverse order.

You can define multiple Compensation Actions for each Java Action in an Action Set.

Note: You cannot define a Compensation Action for 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.

In the following example, you are planning to execute the same Prologue Actions as described in the example above with an addition two Java Actions:

1. Sending an email message
2. Some other action
3. Transferring a file to a remote location using FTP
4. Some other action
5. Insert a row to an external database

You have created Compensation Actions for some of the Prologue Actions. You can indicate when exactly the Compensation Action is supposed to run by placing it in the list of actions. The following table shows which Compensation Actions have been specified for the Prologue Actions:

Prologue Actions (PA)	Compensation Actions (CA)
Sending an email message (PA1)	Sending another email message, for example, to apologize for any changes in the original email (CA1)
Some other action (PA2)	No Compensation Action defined (CA2)
Transferring a file to a remote location using FTP (PA3)	Removing the file from the FTP server (CA3)
Some other action (PA4)	No Compensation Action defined (CA4)
Insert a row to an external database (PA5)	Delete the row from the external database (CA5)

During execution, Java Actions may or may not throw an exception. At runtime, the following actions can be executed:

- **Normal execution:** In normal execution, the Java Actions specified in Step 1 would be run. The Compensation Actions specified in Step 2 are skipped and ignored.
- **Execution with errors:** As an example, PA4 throws an exception. First, a new transaction is used to execute the defined Compensation Actions. All Compensation Actions defined for the Java Actions that have been successfully executed before the exception was thrown, are invoked in reverse order. In the example, the Compensation Action for FTP transfer (CA3) is executed, and then the Compensation Action for sending an email (CA1) is executed in that order. CA2 cannot be executed because no Compensating Action has been defined for the second Prologue Action (PA2). Finally, the entire transaction is rolled back. This only affects, however, changes in the Interstage BPM Application Server context.

Note: Only successful actions can be compensated. If, for example, PA5 fails, then again only CA3 and CA1 are executed.

The following figure summarizes the execution of regular Java Actions, taking Compensation Actions into account:

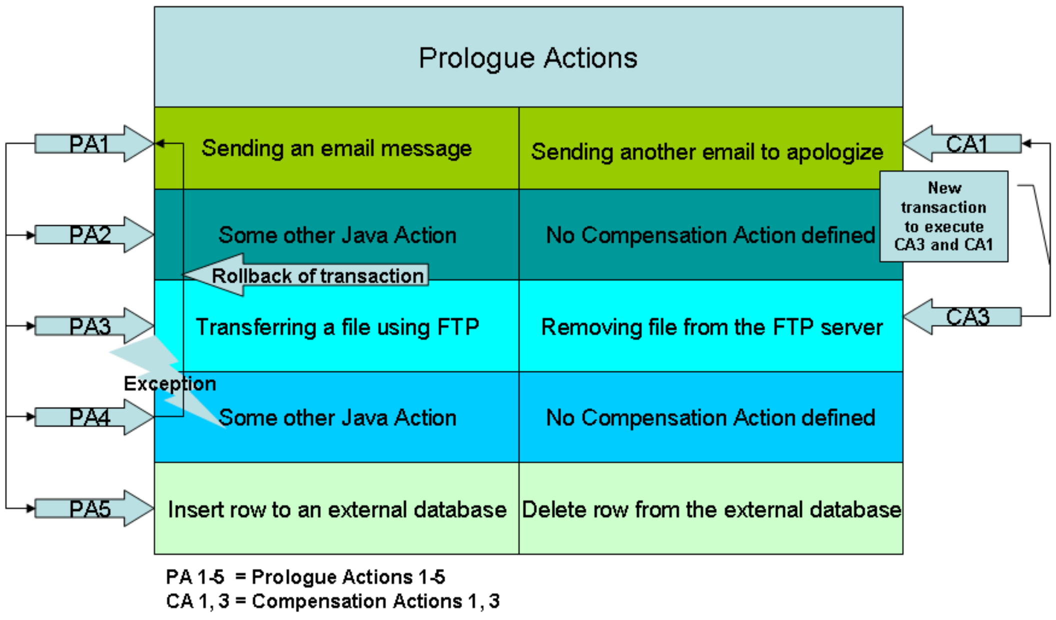

Prologue Actions
PA1
Sending an email message
Sending another email to apologize
CA1
New transaction to execute CA3 and CA1
PA2
Some other Java Action
No Compensation Action defined
Rollback of transaction
PA3
Transferring a file using FTP
Removing file from the FTP server
CA3
Exception
PA4
Some other Java Action
No Compensation Action defined
PA5
Insert row to an external database
Delete row from the external database
PA 1-5 = Prologue Actions 1-5
CA 1, 3 = Compensation Actions 1, 3

Figure 161: Executing regular Java Actions and Compensation Actions

Using Error and Compensation Actions

This example shows how Error and Compensation Actions are executed if a Java Action throws an exception.

In the following example, you have created one Java Action Set (for example a Prologue Action Set) that consists of two Java Actions:

- **Java Action 1**: Java Action 1 (JA1) is used to update an external database. One Error Action Set and one Compensation Action Set are assigned to this Java Action. The Error Action Set contains one Error Action and the Compensation Action Set two Compensation Actions.

- **Java Action 2**: Java Action 2 (JA2) is used to call a Web service. This Java Action contains an Error Action Set with three Error Actions. For this Java Action, no Compensation Actions are available.

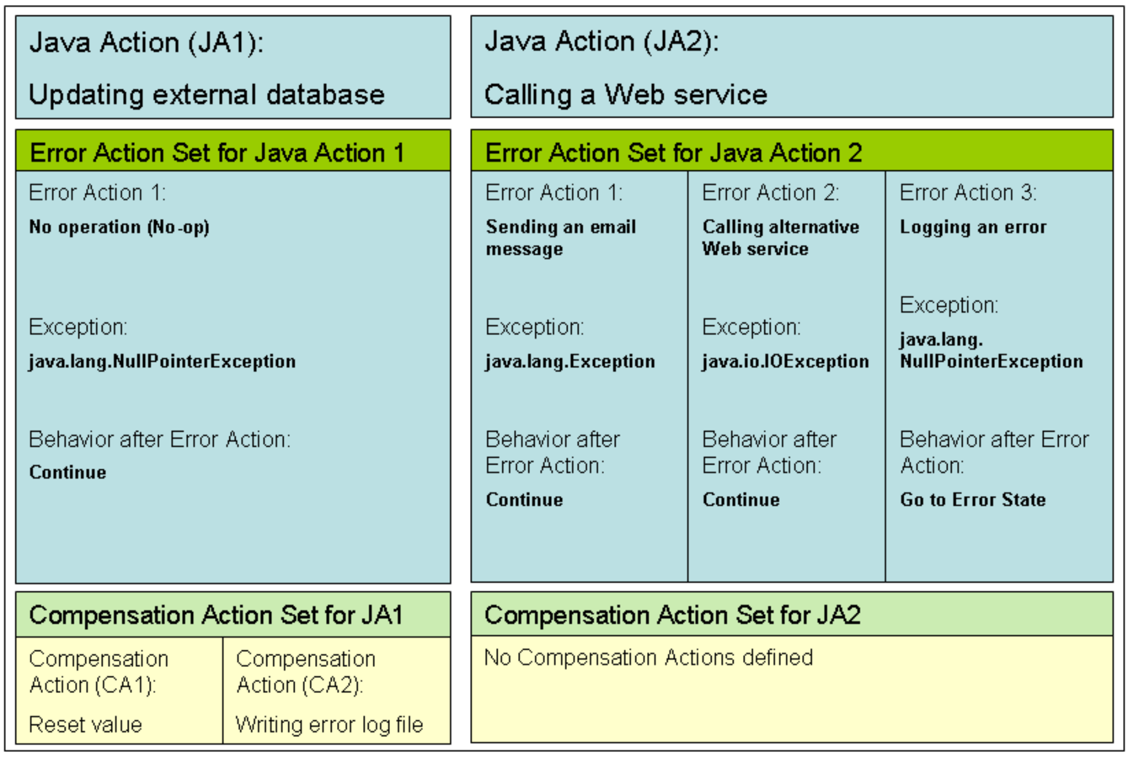

Java Action (JA1):
Updating external database
Java Action (JA2):
Calling a Web service
Error Action Set for Java Action 1
Error Action 1:
No operation (No-op)
Exception:
java.lang.NullPointerException
Behavior after Error Action:
Continue
Error Action Set for Java Action 2
Error Action 1:
Sending an email message
Exception:
java.lang.Exception
Behavior after Error Action:
Continue
Error Action 2:
Calling alternative Web service
Exception:
java.io.IOException
Behavior after Error Action:
Continue
Error Action 3:
Logging an error
Exception:
java.lang.NullPointerException
Behavior after Error Action:
Go to Error State
Compensation Action Set for JA1
Compensation Action (CA1):
Reset value
Compensation Action (CA2):
Writing error log file
Compensation Action Set for JA2
No Compensation Actions defined

Figure 162: Executing Java Actions in Error Cases

At runtime, the following actions can be executed:

- **Action 1:** In normal execution, the two Java Actions (JA1 and JA2) are executed one after another. No errors occur during execution.
- **Action 2:** JA1 (Updating external database) throws an exception, for example a `FileNotFoundException`. Since no Error Action has been defined for this exception, the process instance goes to error state.
- **Action 3:** JA1 (Updating external database) throws an exception. In the example, a `NullPointerException` occurs. An Error Action has been defined for this exception, so the `No-op` Error Action included with the Error Action Set will be executed. The exception gets caught, and the process instance continues.
- **Action 4:** JA1 (Updating external database) is successfully executed, and JA2 (Calling a Web service) throws an `IOException`. The Error Actions included with the `Calling alternative Web service` and `Sending email` Error Actions will be executed because the exception thrown is a subclass of `Exception`. As in the previous action, the exception gets caught, and the process instance continues.
- **Action 5:** JA1 (Updating external database) is successfully executed, and JA2 (Calling a Web service) throws a `NullPointerException`. This exception is a subclass of `Exception`, and the `Logging an error` and `Sending an email message` Error Actions will be executed. However, the exception does not get caught because the **Goto Error State** setting overrides the **Continue**

after Error setting. Therefore, the two Error Actions are executed first. After that, the two Compensation Actions of JA1 are executed. Finally, the process instance goes to error state.

12.2 Using Server Actions

Server Actions enable you to interact with the Interstage BPM 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

12.2.1 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.

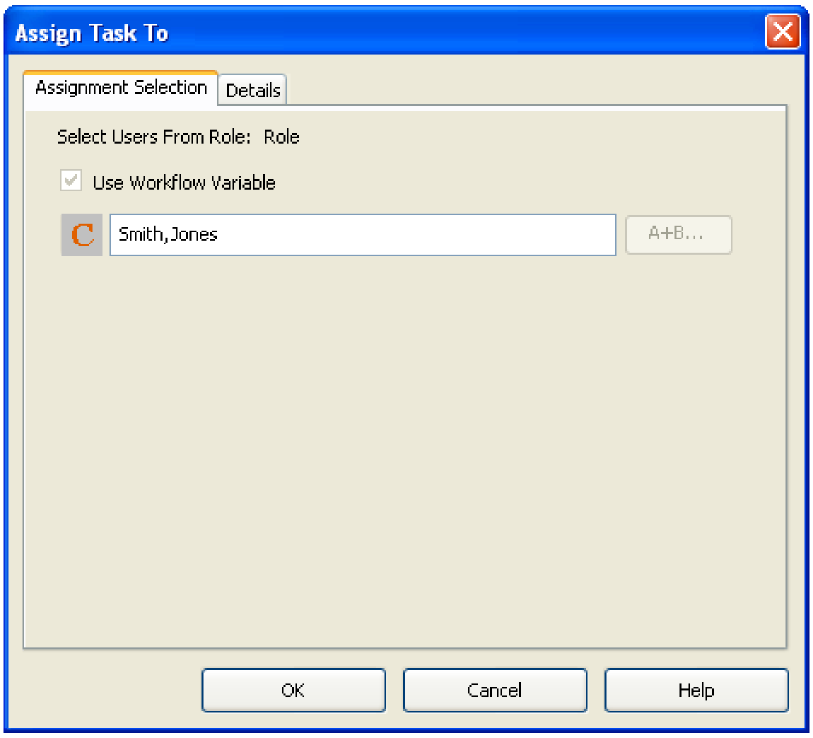
To assign an activity to a user:

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 *Defining JavaScript Expressions* on page 371.

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.



Select Users From Role: Role

Use Workflow Variable ✓

Smith,Jones

Figure 163: Assigning an Activity to Users

> **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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

> **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.

12.2.2 Assigning Task to Performer of Completed Activity

While designing a Process Definition, we can assign activities to a group in the **General** tab of **Properties** view of the Process Definition (refer *Configuring Work Item Generation* on page 150 for more information about assigning activities to a group) or by using Java Actions. If you want to assign activities 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.

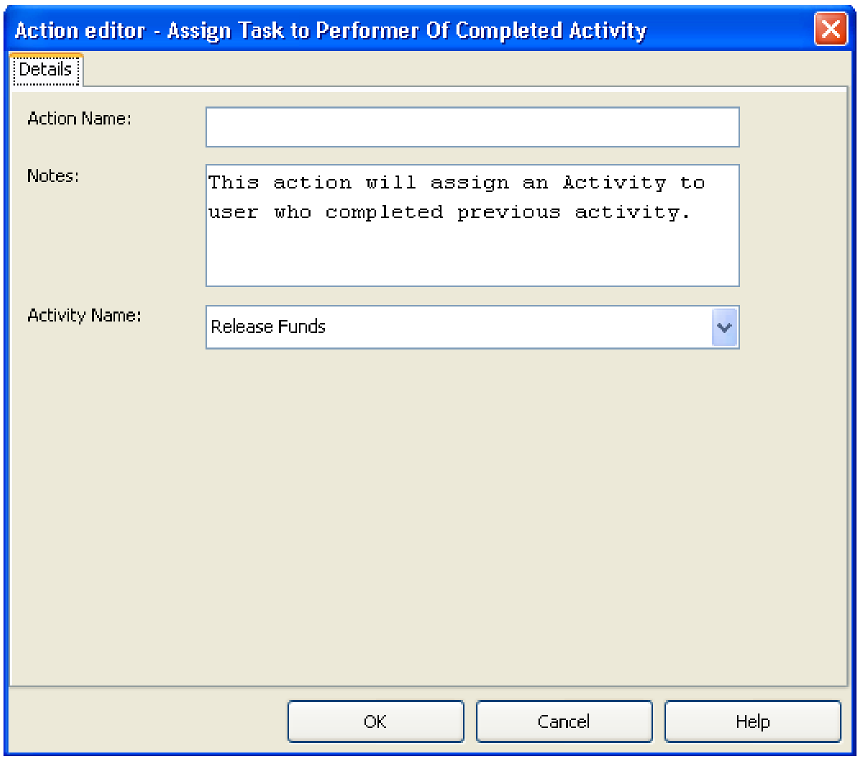
   Java Action Sets area is displayed in **Action Set** tab. By default, **Role Actions** tab is selected in **Java Action Sets**.

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**.

Action editor - Assign Task To Performer Of Completed Activity dialog is displayed as shown in the figure below.


Action editor - Assign Task to Performer Of Completed Activity
Details
OK
Cancel
Help

Action Name:

Notes: This action will assign an Activity to user who completed previous activity.

Activity Name: Release Funds

Figure 164: Assigning task to performer of completed activity

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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.3 Setting Assignee from Relationship

While designing a Process Definition, we can assign activities to a group in the **General** tab of **Properties** view of the Process Definition (refer *Configuring Work Item Generation* on page 150 for more information about assigning activities to a group) or by setting Java Actions. If you want to assign activities 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.

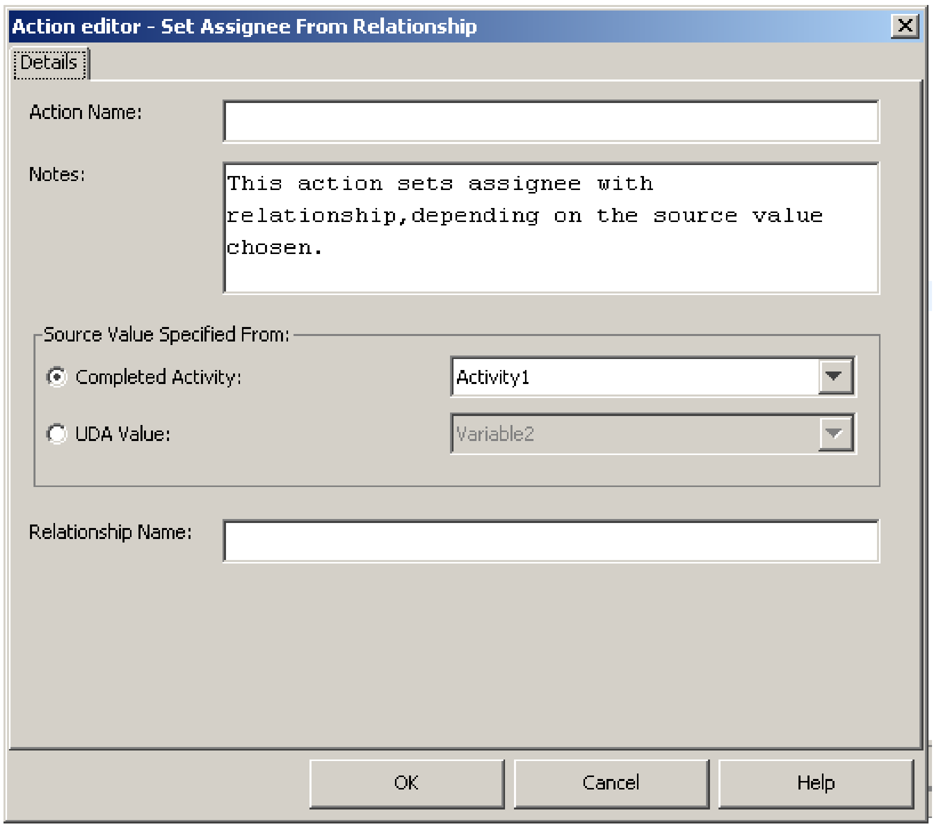
   The **Java Action Sets** area is displayed in **Action Set** tab. By default, **Role Actions** tab is selected in **Java Action Sets**.

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**.

Action editor - Set Assignee From Relationship dialog is displayed as shown in the figure below.



Action Name:

Notes: This action sets assignee with relationship,depending on the source value chosen.

Source Value Specified From:

[✓ Completed Activity:] Activity1

UDA Value: Variable2

Relationship Name:

Figure 165: Setting assignee from relationship

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: Refer the *Interstage Business Process Manager Developer's Guide* for Relationship Name.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.4 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 *Setting Activity Level Priority* on page 137.

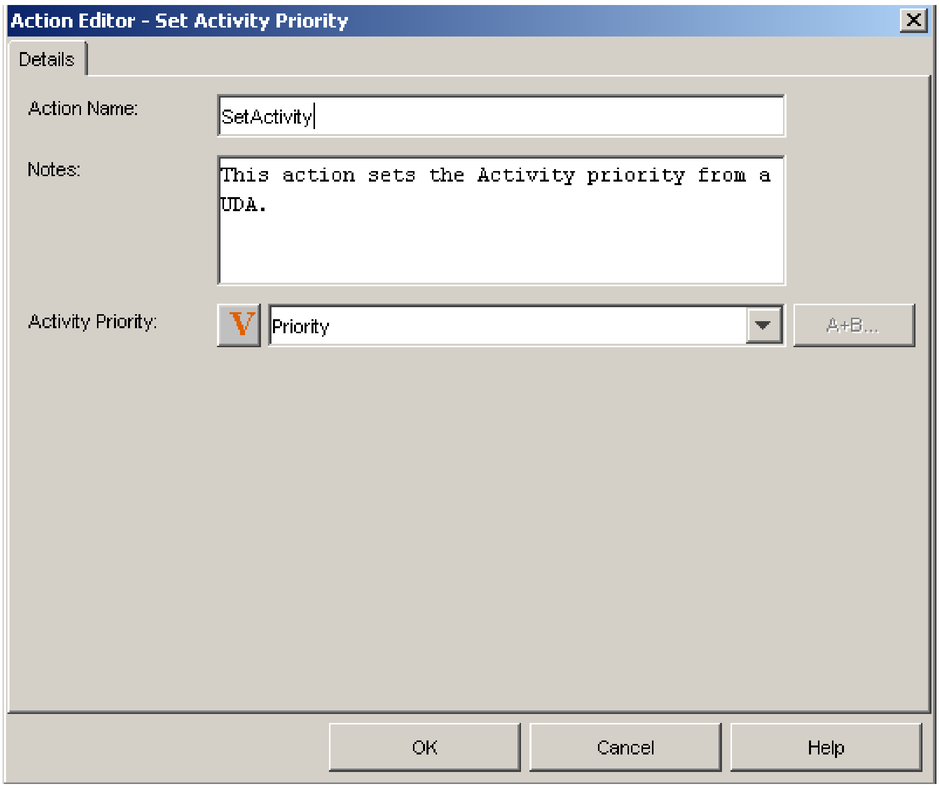
To set the activity priority using Java Action:

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* on page 371.

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`.



Action Name: SetActivity

Notes: This action sets the Activity priority from a UDA.

Activity Priority: V Priority

Figure 166: Setting Activity 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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.5 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:

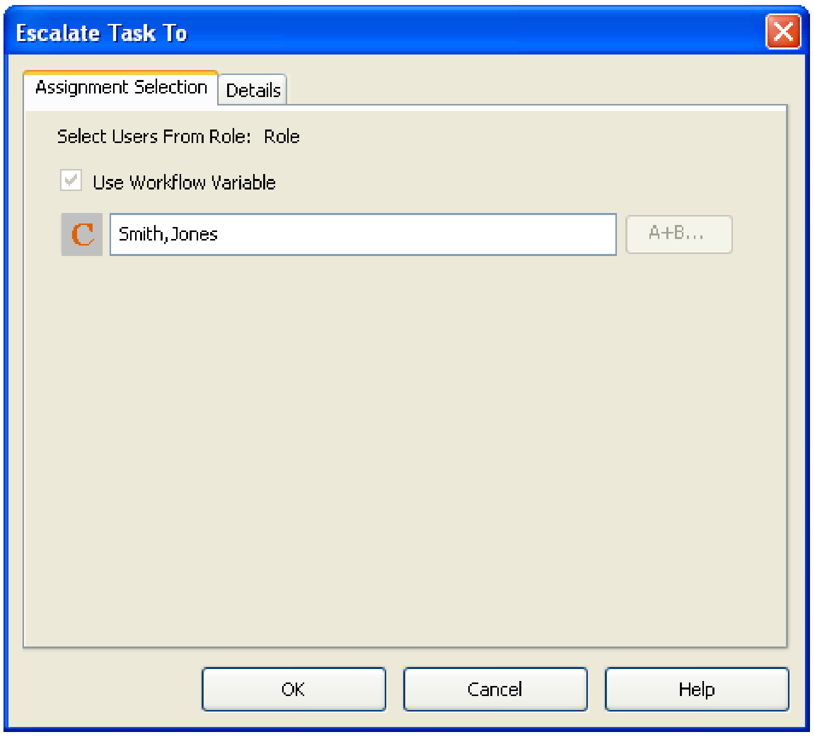
1. Define a due date or timer for an Activity Node or Voting Activity Node or Compound Activity Node.

   Refer to sections *Defining Due Dates* on page 169 or *Defining Timers* on page 171 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 *Defining JavaScript Expressions* on page 371.

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.



Select Users From Role: Role

[✓] Use Workflow Variable

Smith,Jones

Figure 167: Escalating an Activity

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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

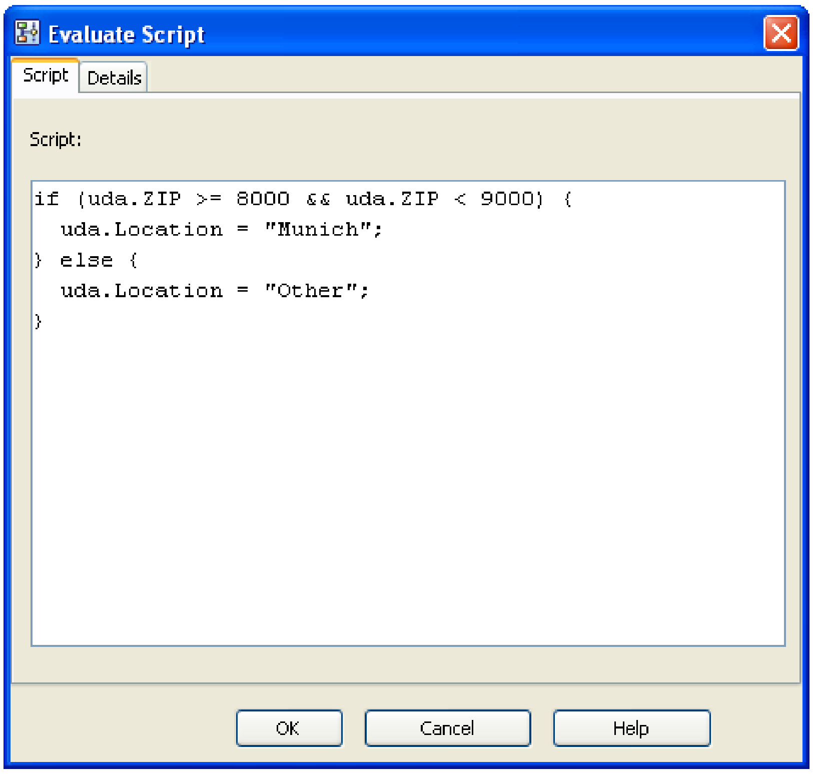
12.2.6 Evaluating a JavaScript

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 *Supported JavaScript Functions* on page 404.

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.



```
if (uda.ZIP >= 8000 && uda.ZIP < 9000) {
  uda.Location = "Munich";
} else {
  uda.Location = "Other";
}
```

Figure 168: Evaluating a JavaScript

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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

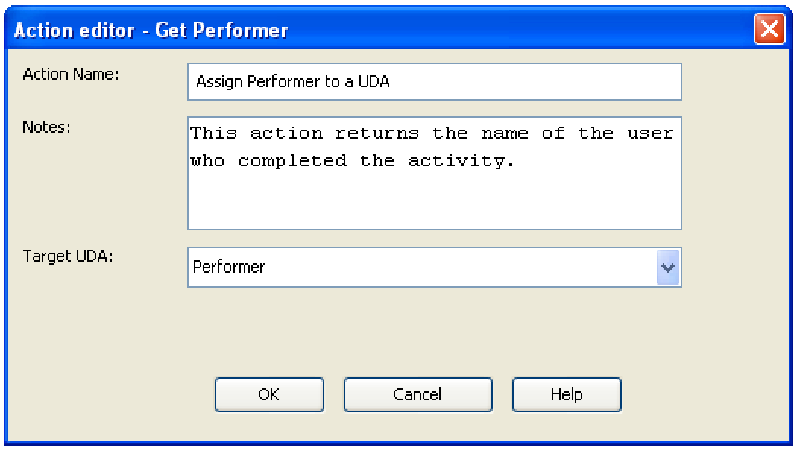
12.2.7 Getting the Performer of an Activity

Prerequisite: The process definition has a User Defined Attribute (UDA) to which the performer can be assigned.

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.

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.



Action Name: Assign Performer to a UDA

Notes: This action returns the name of the user who completed the activity.

Target UDA: Performer

Figure 169: Assigning the Performer's Name to a UDA

6. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

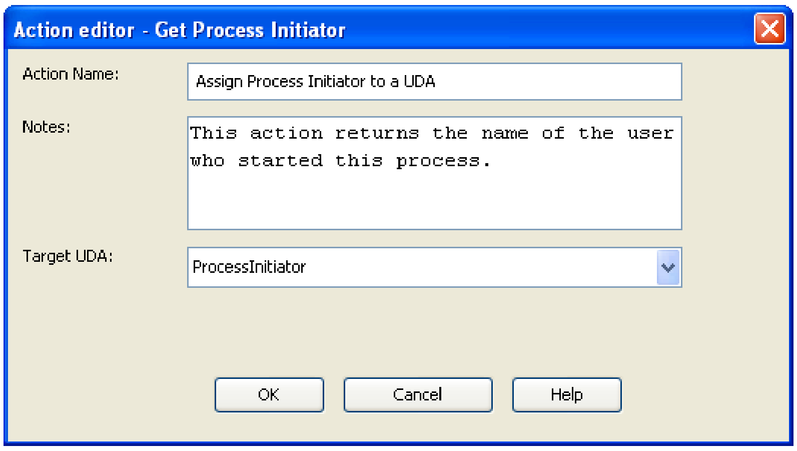
12.2.8 Getting the Process Initiator

Prerequisite: The process definition has a User Defined Attribute (UDA) to which the process initiator can be assigned.

Using the `Get Process Initiator` Java Action, you can assign the name of the user who starts the process instance to a UDA.

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.



Action Name: Assign Process Initiator to a UDA

Notes: This action returns the name of the user who started this process.

Target UDA: ProcessInitiator

Figure 170: Assigning the Initiator 's Name to a UDA

6. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.9 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.

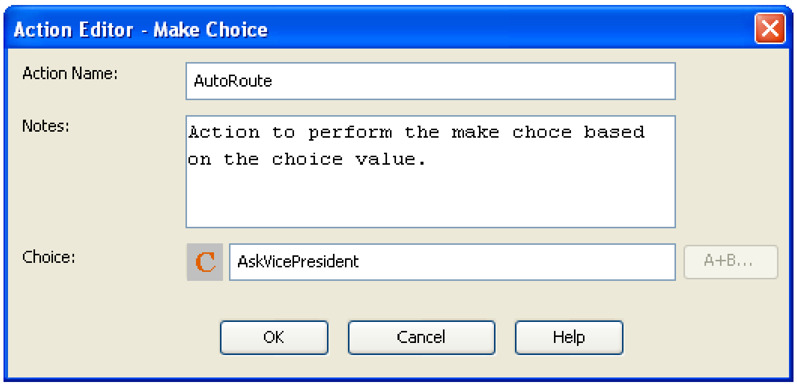
To make an automatic choice:

1. Define a timer for the Activity Node or Voting Activity Node or Compound Activity Node.

   Refer to section *Defining Timers* on page 171 for instructions.
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 *Defining JavaScript Expressions* on page 371.



Action Name: AutoRoute

Notes: Action to perform the make choce based on the choice value.

Choice: AskVicePresident

Figure 171: Making an Automatic Choice

6. Click **OK**.

12.2.10 Setting the Process Instance Name

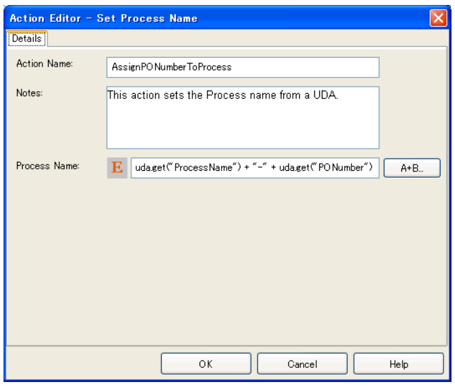
By default, a process instance has the same name as the process definition from which it is created. However, its name can be changed while the process instance is running using the `Set Process Instance Name` Java Action.

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 *Defining JavaScript Expressions* on page 371.

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.



Action Name: AssignPONumberToProcess

Notes: This action sets the Process name from a UDA.

Process Name: uda.get("ProcessName") + "-" + uda.get("PONumber")

Figure 172: Setting the Process Instance Name

6. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.11 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 Process Instance Priority` Java Action.

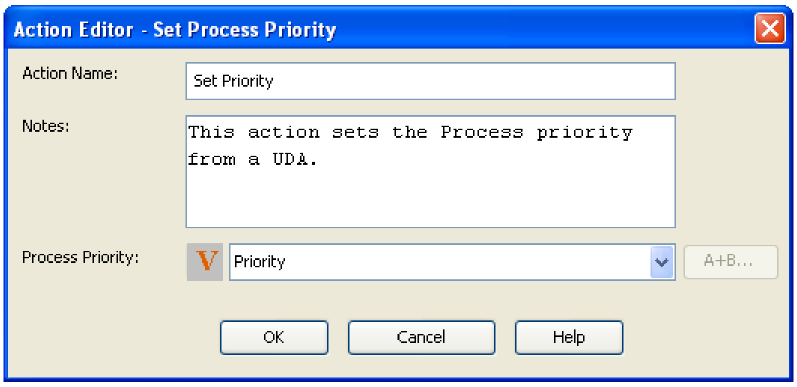
To set the process instance priority:

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 *Defining JavaScript Expressions* on page 371.

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 process instance priority is set to the value of the UDA `Priority`.



Action Name: Set Priority

Notes: This action sets the Process priority from a UDA.

Process Priority: Priority

Figure 173: Setting Process Instance 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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.12 Setting the Process Instance Description

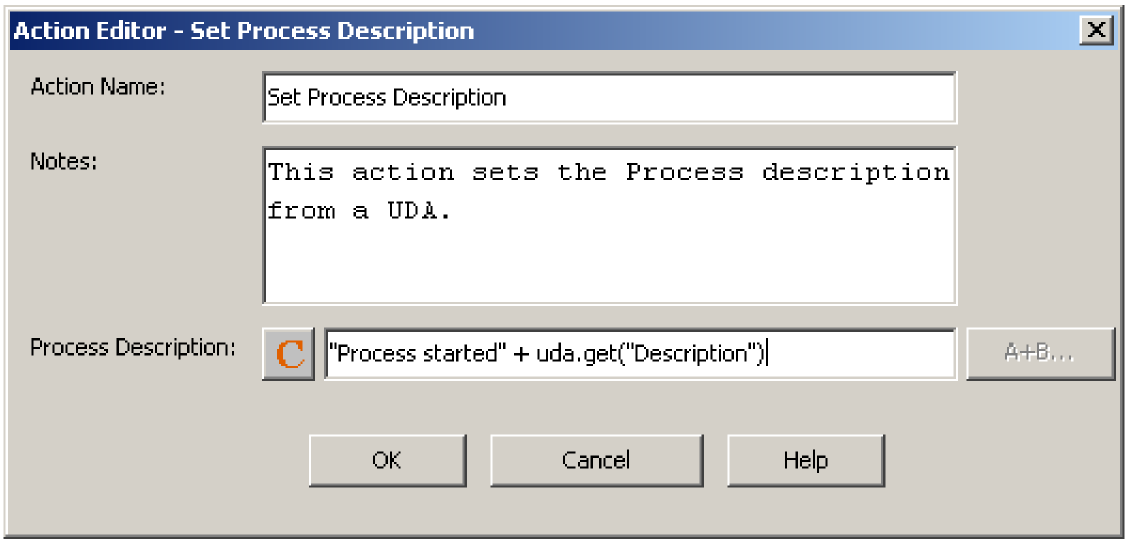
By default, a process instance is given the same description as the process definition from which it is created. However, its description can be changed while the process instance is running using the `Set Process Instance Description` Java Action.

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 *Defining JavaScript Expressions* on page 371.

In the following example, the process instance description is composed of the string `Process started` and the value of the UDA `Description`.



Action Name: Set Process Description

Notes: This action sets the Process description from a UDA.

Process Description: "Process started" + uda.get("Description")

Figure 174: Setting the Process Instance Description

6. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.2.13 Assigning Values to User Defined Attributes

Prerequisite: You have added User Defined Attributes (UDAs) to the process definition.

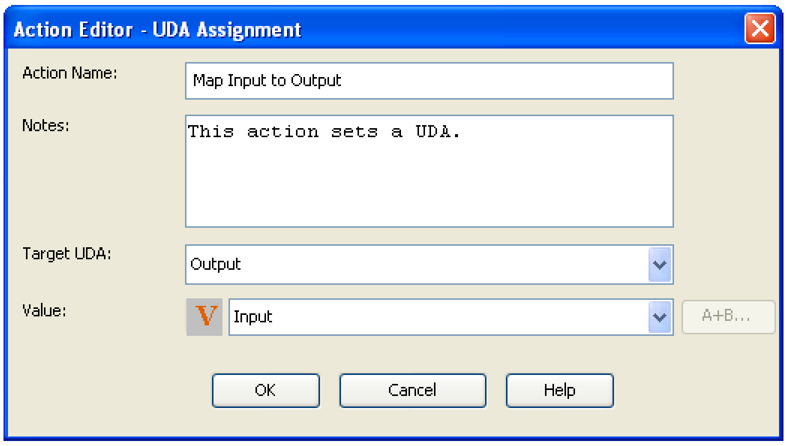
You can set the value of a UDA using the `UDA Assignment` Java Action. You can set it, for example, to the value of another UDA.

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 *Defining JavaScript Expressions* on page 371.

In the following example, the value of the UDA `Input` is assigned to the UDA `Output`.


Action Editor - UDA Assignment

Action Name: Map Input to Output

Notes: This action sets a UDA.

Target UDA: Output

Value: Input

Figure 175: Setting a UDA

7. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.3 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 or Custom. In addition, you can delete XML substructures, text elements or attribute values from a UDA of type XML or Custom, assign an XML string to a UDA of type XML or Custom and extract values from XML data.

12.3.1 Adding a Substructure in XML

Prerequisites:

The process definition has a UDA of type XML or Custom to which a new substructure can be added.

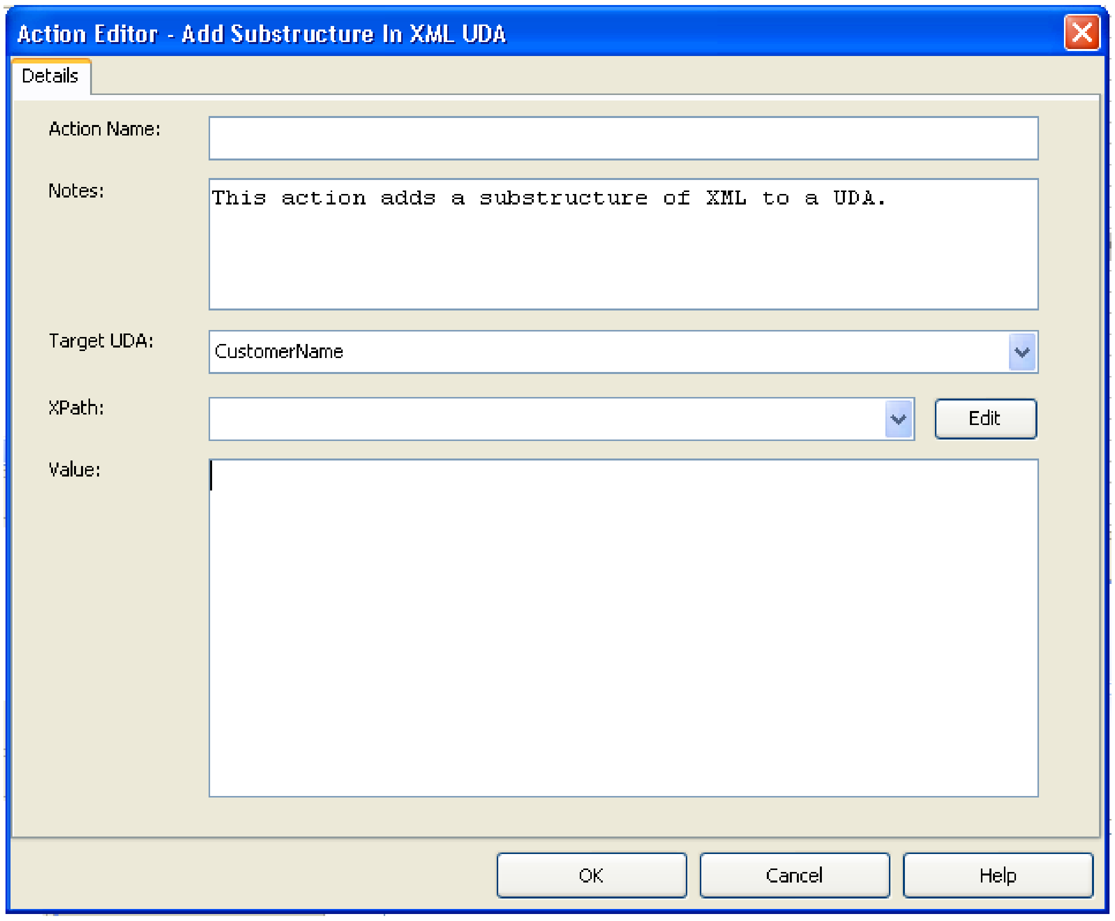
When specifying a UDA of type XML or Custom, you might want to add an XML substructure to it. This can be achieved using the `Add Substructure in XML` Java Action.

Note: You can assign this Java Action to process definitions and all nodes.

To add a substructure to a UDA of type XML or Custom:

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 or Custom you have defined for the process definition or node (`CustomerName` UDA, in the example), and provides a default description:



Action Name:

Notes: This action adds a substructure of XML to a UDA.

Target UDA: CustomerName

XPath:

Value:

Figure 176: Displaying the Action Editor dialog

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 or Custom.

   All the XPaths related to the target UDA (type XML or Custom) 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.

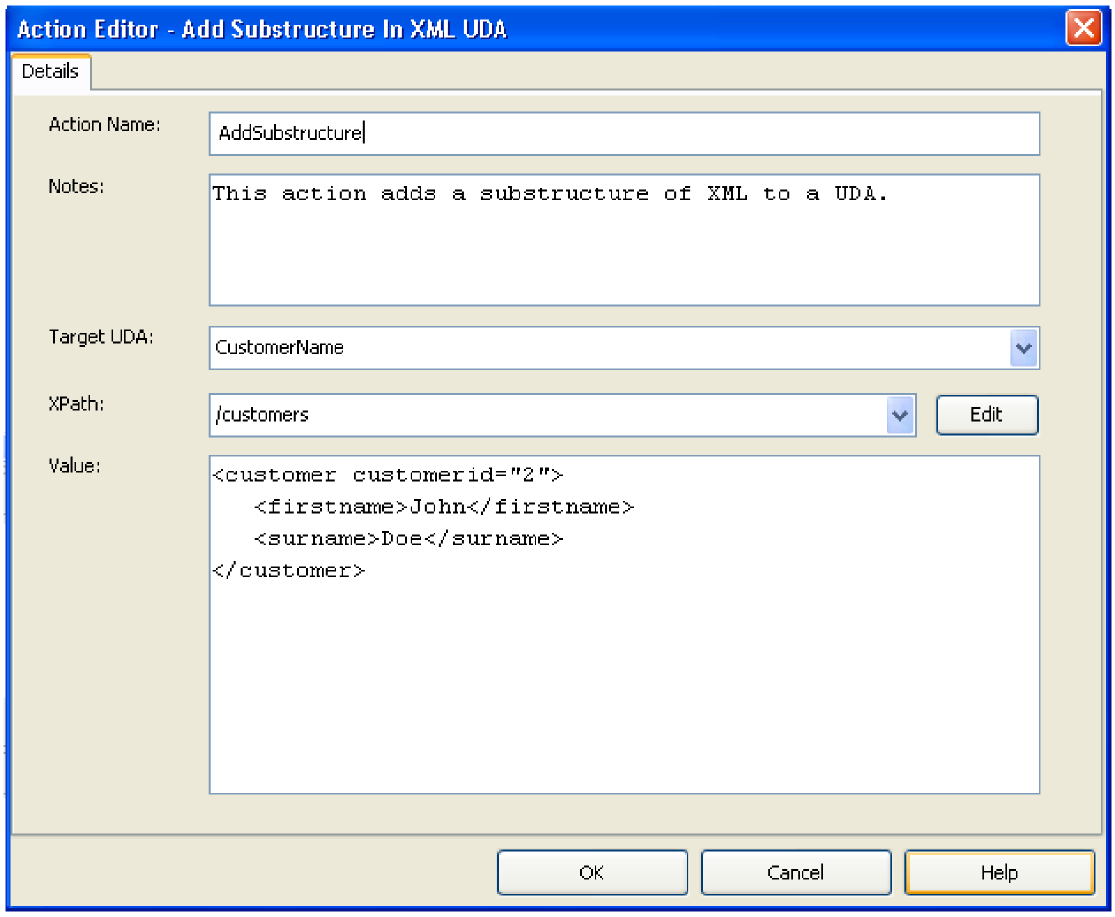
Note: The XPath expressions related to the selected XML or Custom UDA are displayed in the XPath drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected XML UDA, the XPath list that can be 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.

Note: If a Custom UDA is selected, XPath expressions specifying only non-leaf nodes will be shown in the **XPath** drop-down list.

8. In the **Value** field, specify the value of the XML substructure.

   When entering a UDA value, use the same tags as for specifying a Web Service call or assigning an XML string to a UDA. For more information, refer to sections *Assigning an XML String to a User Defined Attribute* on page 304 and *Web Services: Special Tags for SOAP Request Messages* on page 342. The following example shows the Action Editor dialog how the `customer` substructure is added to the `CustomerName` UDA:



Action Name: AddSubstructure

Notes: This action adds a substructure of XML to a UDA.

Target UDA: CustomerName

XPath: /customers

Value:

```
<customer customerid="2">
   <firstname>John</firstname>
   <surname>Doe</surname>
</customer>
```

Figure 177: Adding a Substructure in XML UDA

9. Click **OK**.

   When closing the dialog, Interstage BPM 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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.3.2 Extracting User Defined Attribute Values from XML Data

Prerequisites:

- The process definition has a User Defined Attribute (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 are familiar with XPath expressions and know how to write them.

You can extract data from an XML string coming in to Interstage BPM 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 Interstage BPM.

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 *Specifying User Defined Attributes* on page 151.
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 or CUSTOM.

      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 or CUSTOM.

   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.

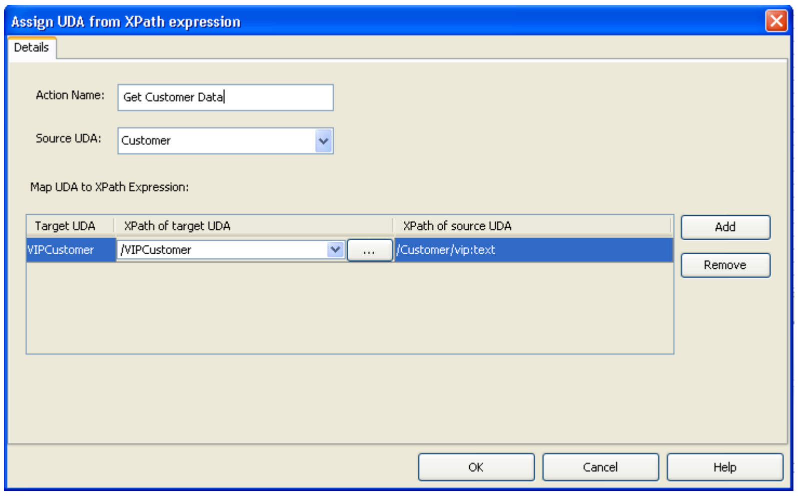
g) If you want to remove a mapping, select it and click **Remove**.

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 or CUSTOM 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 XML 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 Interstage BPM 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.


Assign UDA from XPath expression

Details

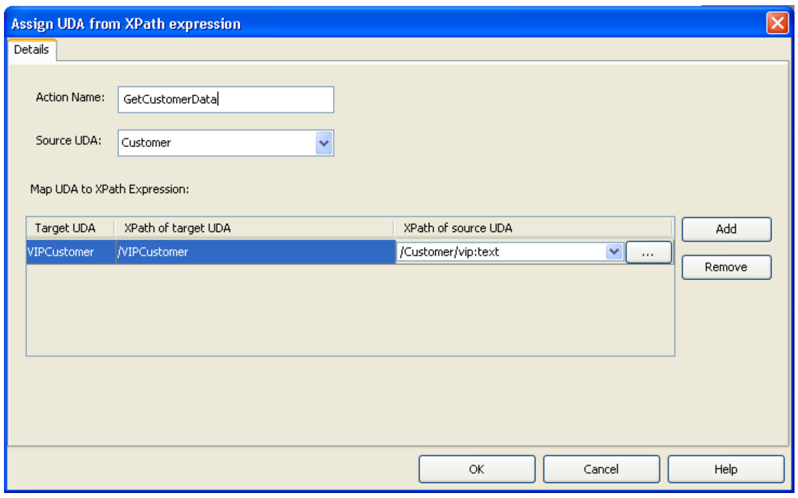
Action Name: Get Customer Data

Source UDA: Customer

Map UDA to XPath Expression:

Target UDA	XPath of target UDA	XPath of source UDA
VIPCustomer	/VIPCustomer	/Customer/vip:text

Figure 178: Extracting Data from an XML String - XPath of target UDA


Assign UDA from XPath expression

Details

Action Name: GetCustomerData

Source UDA: Customer

Map UDA to XPath Expression:

Target UDA	XPath of target UDA	XPath of source UDA
VIPCustomer	/VIPCustomer	/Customer/vip:text

Figure 179: Extracting Data from an XML String - XPath of source UDA

7. Click **OK**.

Note:	It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.3.3 Assigning an XML String to a User Defined Attribute

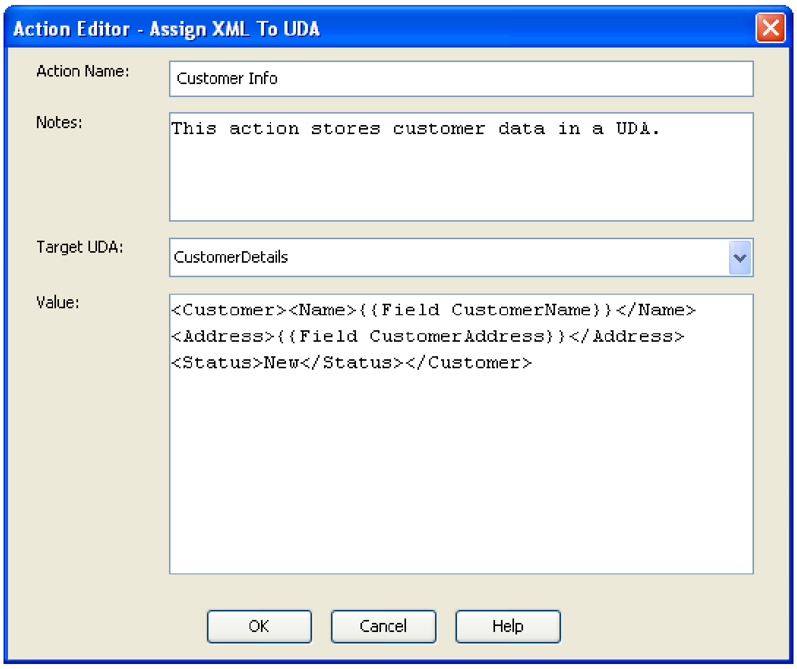
Prerequisite: The process definition has a User Defined Attribute to which the XML string can be assigned.

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 Interstage Business Process Manager.

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 or CUSTOM.
6. In the **Value** field, type the XML string to be assigned.



Action Name: Customer Info

Notes: This action stores customer data in a UDA.

Target UDA: CustomerDetails

Value:
```
<Customer><Name>{(Field CustomerName)}</Name>
<Address>{(Field CustomerAddress)}</Address>
<Status>New</Status></Customer>
```

Figure 180: Assigning an XML String to a UDA

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>}}`
- `{{Xml <UDAName>}}`
- `{{Js <JavaScriptExpression>}}`
- `{{JsXml <JavaScriptExpression>}}`

These are the same tags that can be used when specifying the input parameters for a Web Service call. For more information and for examples, refer to section *Web Services: Special Tags for SOAP Request Messages* on page 342.

7. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.3.4 Deleting Substructures, Text and Attribute Values from a UDA

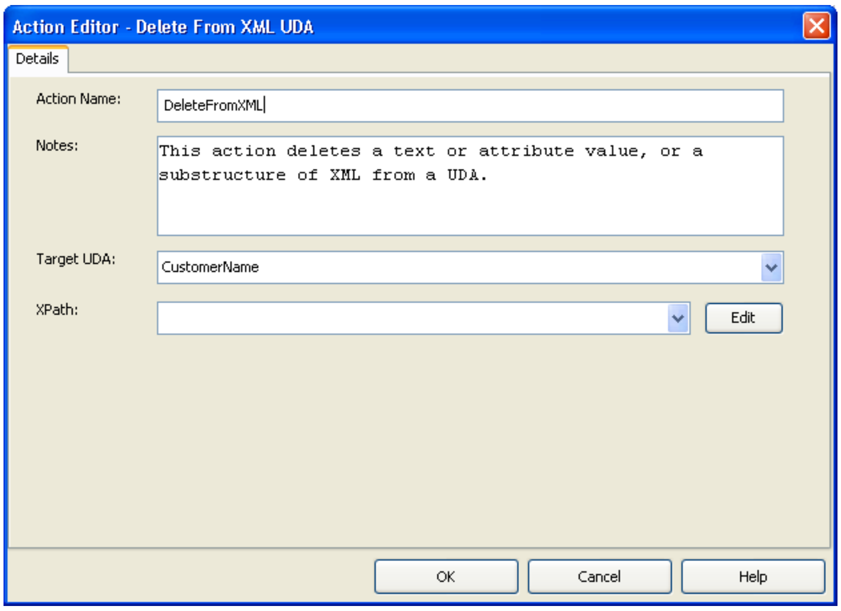
Prerequisite: The process definition has a User Defined Attribute of type XML or CUSTOM from which a substructure, text element or attribute value can be removed.

You might want to remove substructures, text, or attribute values from a UDA of type XML or CUSTOM. 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 or CUSTOM you have defined for the process definition or node (`CustomerName` UDA, in the example), and provides a default description:



Action Name: DeleteFromXML

Notes: This action deletes a text or attribute value, or a substructure of XML from a UDA.

Target UDA: CustomerName

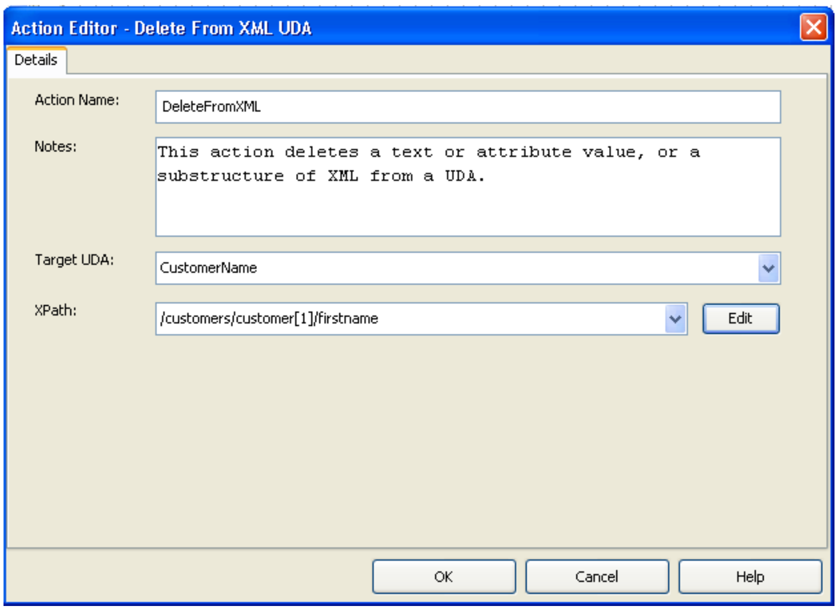
XPath:

Figure 181: Displaying the Action Editor

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 or CUSTOM.
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 *Adding a Substructure in XML* on page 298, for the complete substructure):



Action Name: DeleteFromXML

Notes: This action deletes a text or attribute value, or a substructure of XML from a UDA.

Target UDA: CustomerName

XPath: /customers/customer[1]/firstname

Figure 182: Deleting a substructure

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 or CUSTOM 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 XML 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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.3.5 Setting a Substructure in XML

Prerequisites:

The process definition has a UDA of type XML or CUSTOM whose structure you want to change by replacing an existing substructure.

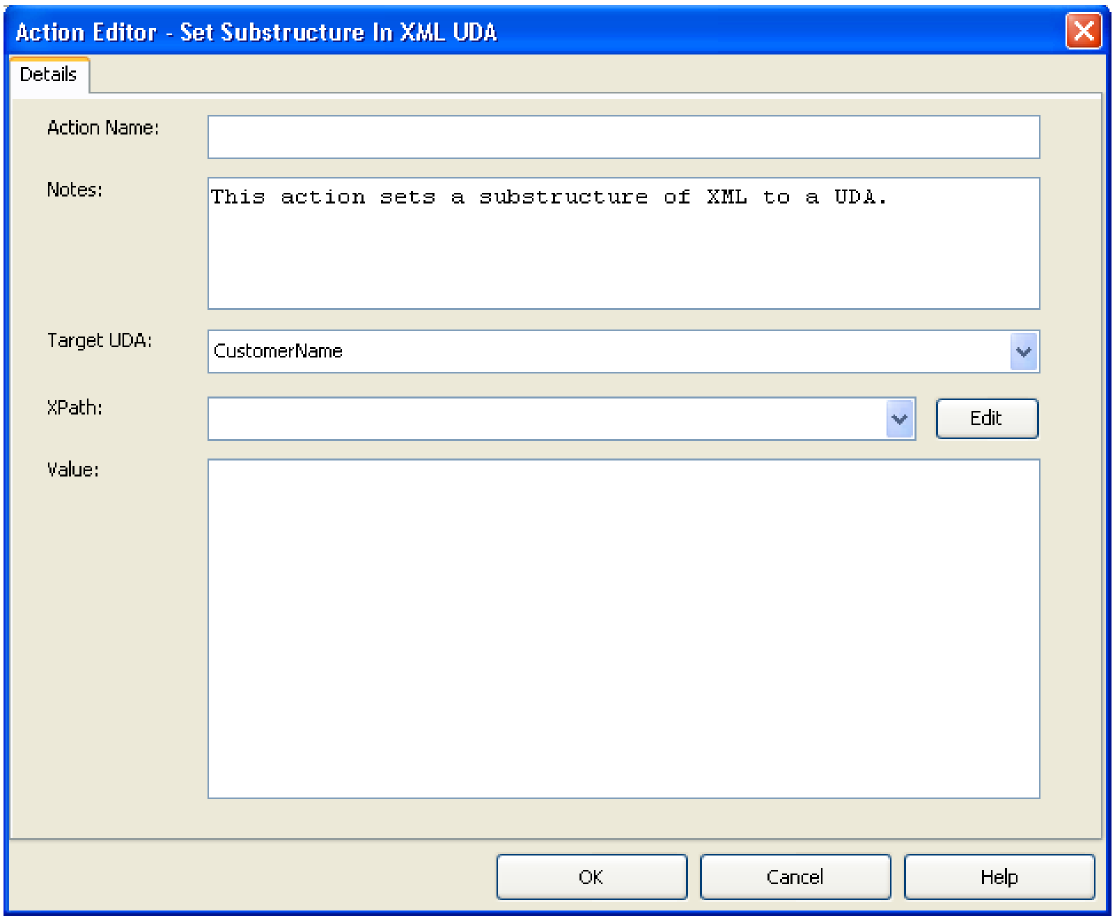
You might want to set a new XML substructure to a UDA of type XML or CUSTOM. This can be achieved using the `Set Substructure in XML` Java Action. You can assign this Java Action to process definitions and all nodes.

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 or CUSTOM:

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:



Notes: This action sets a substructure of XML to a UDA.

Target UDA: CustomerName

Figure 183: Displaying the Action Editor dialog

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.

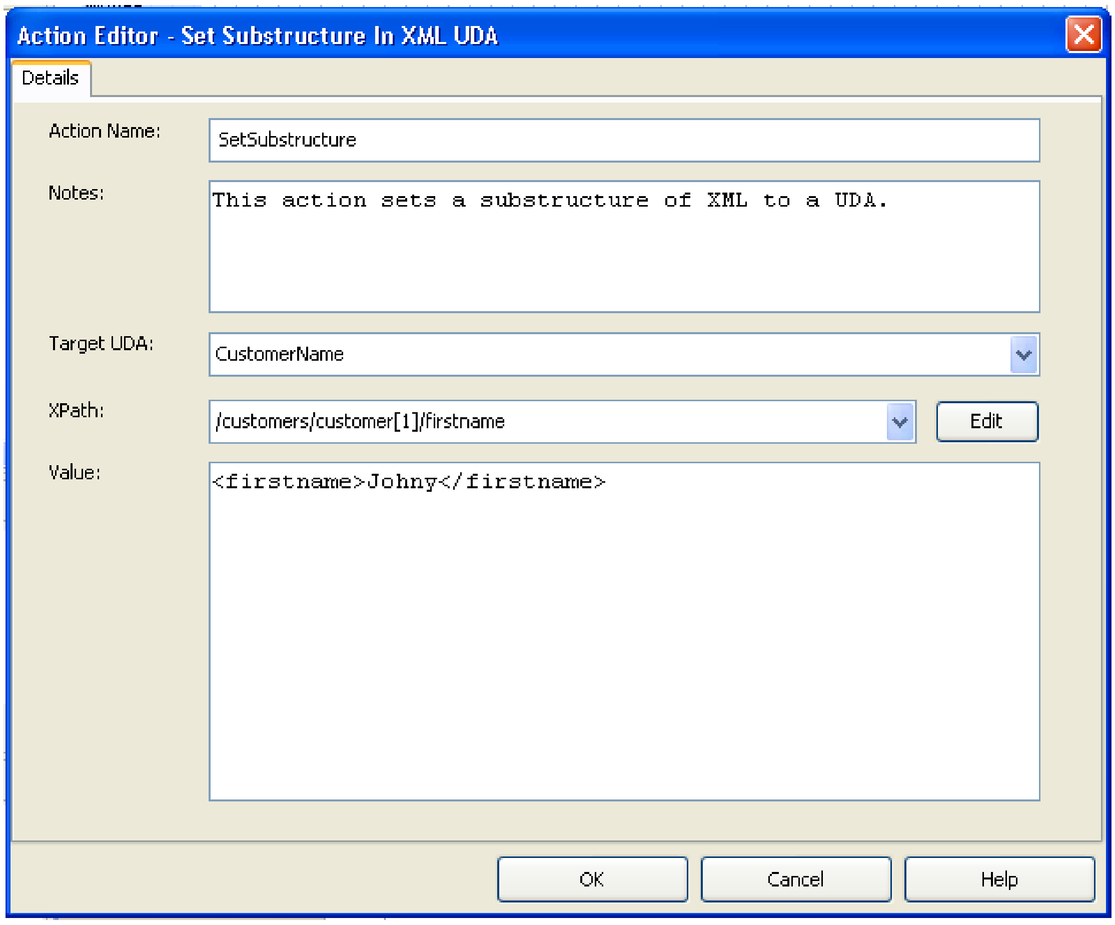
Note: The XPath expressions related to the selected XML or CUSTOM UDA are displayed in the XPath drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected XML UDA, the XPath list that can be 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.

Note: If a CUSTOM UDA is selected, XPath expressions specifying only non-leaf nodes will be displayed in the XPath drop-down list.

8. In the **Value** field, specify the value of the text or attribute.

   When entering a UDA value, use the same tags as for specifying a Web Service call or assigning an XML string to a UDA. For more information, refer to sections *Assigning an XML String to a User Defined Attribute* on page 304 and *Web Services: Special Tags for SOAP Request Messages* on page 342. The following example shows the Action Editor dialog used for replacing the customer substructure you have added before (refer to section *Adding a Substructure in XML* on page 298 for more details):



Action Name: SetSubstructure

Notes: This action sets a substructure of XML to a UDA.

Target UDA: CustomerName

XPath: /customers/customer[1]/firstname

Value: `<firstname>Johny</firstname>`

Figure 184: Setting a Substructure in XML

9. Click **OK** to close the Action Editor dialog.

   When closing the dialog, Interstage BPM 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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.3.6 Setting Text or Attribute Values in XML

Prerequisites:

The process definition has a UDA of type XML or CUSTOM for which you want to set a new text or attribute value.

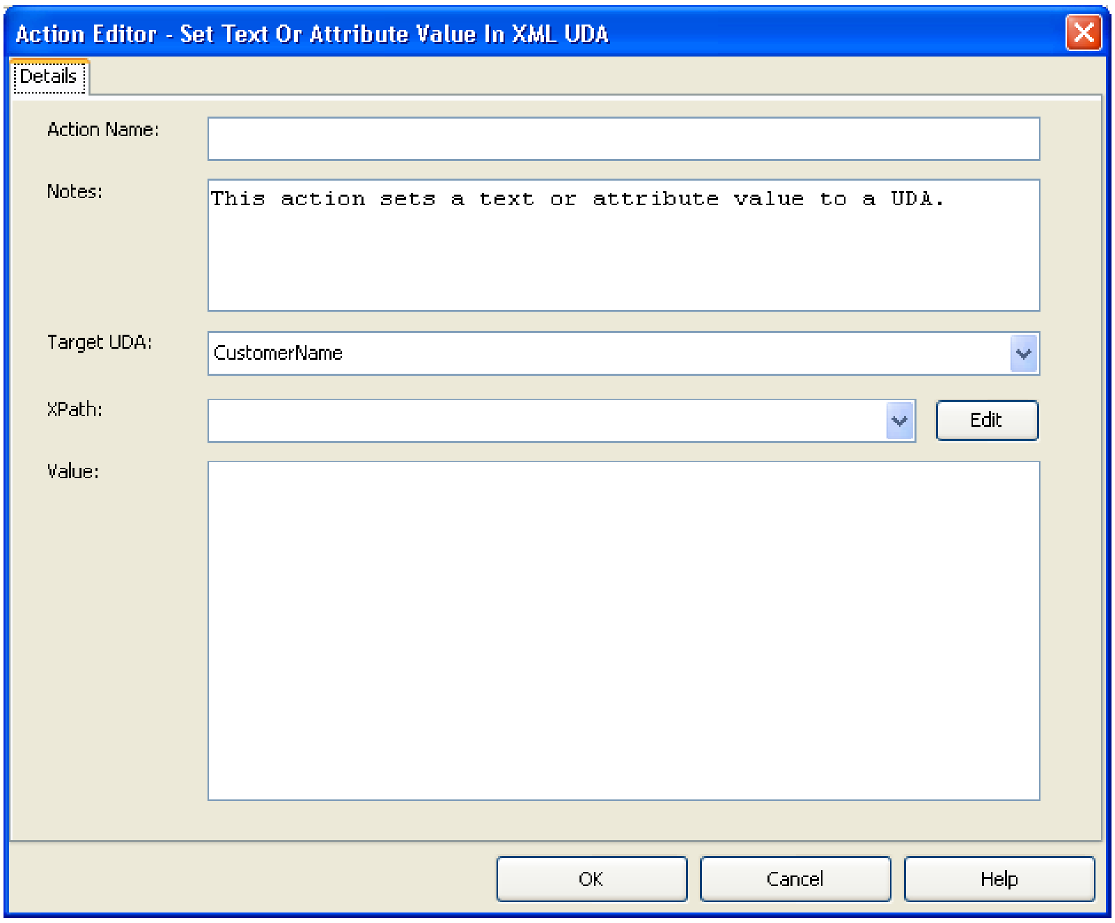
When specifying a UDA of type XML or CUSTOM, you might want to set a new text or attribute value to the UDA. This can be achieved using the `Set Text or Attribute Value in XML` Java Action.

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:



Notes: This action sets a text or attribute value to a UDA.

Target UDA: CustomerName

Figure 185: Displaying the Action Editor dialog

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 or CUSTOM.

   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.

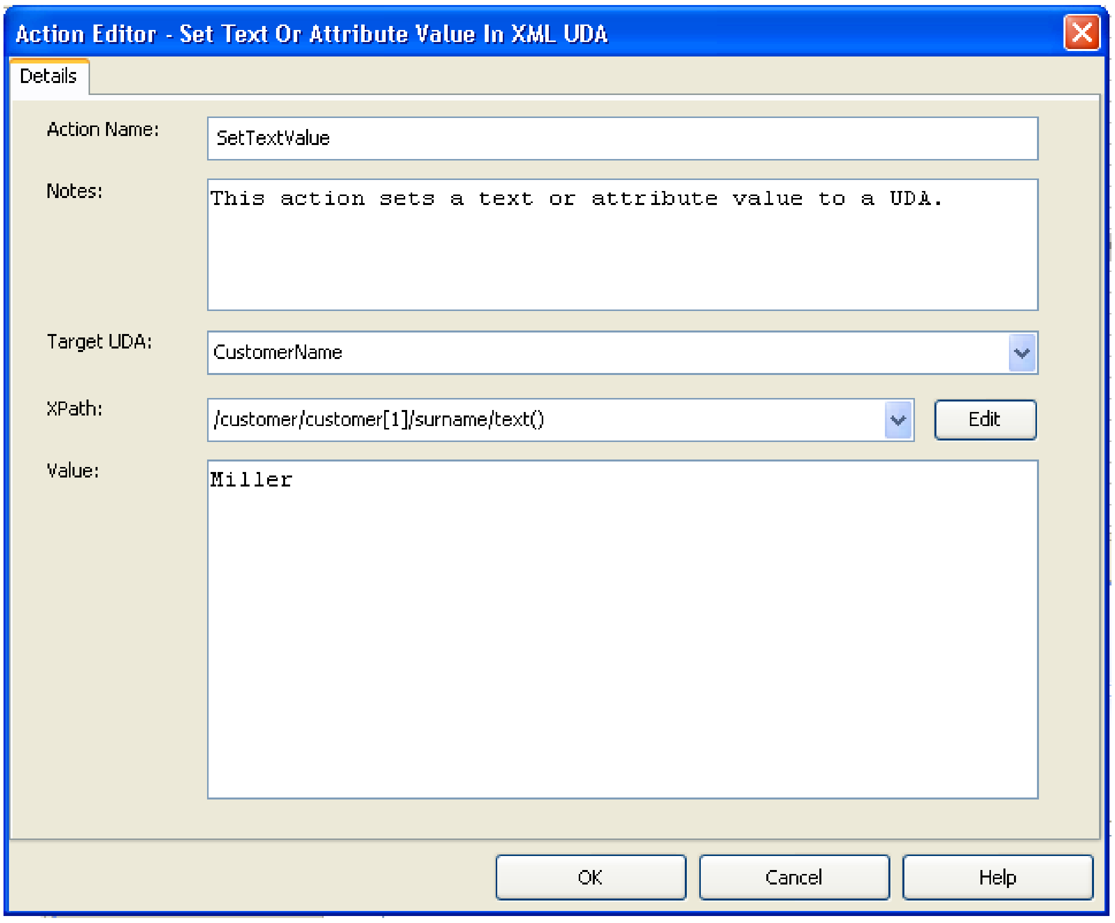
Note: The XPath expressions related to the selected XML or CUSTOM UDA are displayed in the XPath drop-down list. When XML Schema is defined in selected UDA, the XPath list that can be used with this XML Schema is displayed. When Initial value is defined in selected XML UDA, the XPath list that can be 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.

Note: If a CUSTOM UDA is selected, XPath expressions specifying only leaf nodes (e.g. text and attribute nodes) will be displayed in the XPath drop-down list.

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 specifying a Web Service call or assigning an XML string to a UDA. For more information, refer to sections *Assigning an XML String to a User Defined Attribute* on page 304 and *Web Services: Special Tags for SOAP Request Messages* on page 342. The following example shows the Action Editor dialog for changing the last name of the customer from "Doe" (refer to section *Adding a Substructure in XML* on page 298 for the complete substructure) to "Miller". Therefore, use the following XML value and XPath expression:



Action Name: SetTextValue

Notes: This action sets a text or attribute value to a UDA.

Target UDA: CustomerName

XPath: /customer/customer[1]/surname/text()

Value: Miller

Figure 186: Setting Text or Attribute Values in XML

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 Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.4 Using Rules Actions

A Rules Action is a special type of Java Action that acts as an interface to a rules engine. This interface allows you to use advanced process logic in your process definitions at every point in the process definition where you could attach an Action Set.

Interstage BPM supports the following rules engines:

- ILOG JRules
- Blaze Advisor

Before you can create a Rules Action, you must have a rules file that has been created with one of these rules engines. Typically, these rules files will need to integrate Interstage BPM functionality like retrieving or setting process instance owners, User Defined Attribute values, and so on. For more information on creating rules files that use Interstage BPM functionality, refer to the *Interstage Business Process Manager Developer's Guide*.

Rules Actions also allow you to integrate Decision Tables into your process definitions. A Decision Table is a compact way of representing process logic. It consists of a number of business rules where each rule specifies conditions and consequences. For more information, refer to the *Decision Tables* on page 248

12.4.1 Integrating Blaze Rules

Prerequisite: You have a server configuration file that has been created with the Blaze Advisor rules engine.

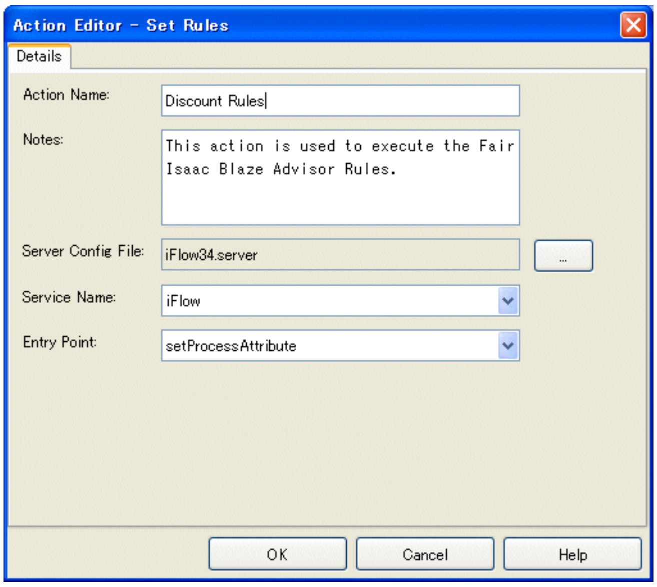
To integrate Blaze rules:

1. Import the server configuration file (`*.server`) that you want to use in your Java Action to the following folder:

   ```
   <Application Project>/dms/Attachments
   ```

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. Expand **Rules Actions** and double click **Fair Isaac Blaze Advisor**.
5. In the **Action Editor - Set Rules** dialog, type a descriptive name and your notes for the Java Action.

6. Click the browse button and select the server configuration file that you placed in the `attachments` subfolder of Interstage BPM Studio.



Action Name: Discount Rules

Notes: This action is used to execute the Fair Isaac Blaze Advisor Rules.

Server Config File: iFlow34.server

Service Name: iFlow

Entry Point: setProcessAttribute

Figure 187: Integrating Blaze Rules

Once the server configuration file has been set in the **Server Config File** field, the services and entry points defined in it become available for selection.

7. Select the service and the entry point that you want to use.

   The service is the name of the Blaze Advisor Server that is running and has the rules defined in it. An entry point is an object in that server that will be invoked for rule evaluation.

8. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

Note: When sending the process definition to an Interstage BPM Server, make the following adjustments. Be aware that these steps are also required when you deploy a Workflow Application project on the Interstage BPM Server:

1. If it is the first time that you use Blaze Advisor with Interstage BPM, copy the following JAR files with the license directory from the Blaze Advisor `lib` directory into the `<Interstage BPM Domain>/lib` directory, e.g. `C:/bea/user_projects/domains/base_domain/lib`:
   - `AdvCommon.jar`, `Advisor.jar`, `AdvisorSvr.jar`
   - `collections.jar`
   - `InnovatorRT.jar`
   - `jaxen.jar`
   - `Ndkjc-2.1A-bin.jar`
   - `OROMatcher.zip`
   - `saxpath.jar`
   - `iFlow.jar`. This file is located in the client/lib folder of your Interstage BPM installation, e.g. `C:/Fujitsu/InterstageBPM/client/lib`.
2. On the Interstage BPM Server, copy the Blaze server configuration file to the directory as specified in the Java Action.
3. Include the Blaze license file in your CLASSPATH.

   If you are using Interstage BPM for WebLogic, update the script `setDomainEnv.cmd` or `setDomainEnv.sh` located in the `<Interstage BPM Domain>/bin` directory, e.g. `C:/bea/user_projects/domains/base_domain/bin`:

   On **Windows**, delete the line:

   ```
   set CLASSPATH=%PRE_CLASSPATH%;%WEBLOGIC_CLASSPATH%;
   %POST_CLASSPATH%;%WLP_POST_CLASSPATH%
   ```

   Add the following lines:

   ```
   set BLAZE_LICENSEPATH=<path to your license file>
   set CLASSPATH=%PRE_CLASSPATH%;%WEBLOGIC_CLASSPATH%;
   %POST_CLASSPATH%;%WLP_POST_CLASSPATH%;%BLAZE_LICENSEPATH%
   ```

   On **UNIX** or **Linux**, delete the line:

   ```
   CLASSPATH="{PRE_CLASSPATH}${CLASSPATHSEP}
   ${WEBLOGIC_CLASSPATH}${CLASSPATHSEP} ${POST_CLASSPATH}${CLASSPATHSEP}
   ${WLP_POST_CLASSPATH}"
   ```

   Add the following lines:

   ```
   BLAZE_LICENSEPATH=<path to your license file>
   export $BLAZE_LICENSEPATH CLASSPATH="{PRE_CLASSPATH}${CLASSPATHSEP}
   ${WEBLOGIC_CLASSPATH}${CLASSPATHSEP} ${POST_CLASSPATH}${CLASSPATHSEP}
   ${WLP_POST_CLASSPATH}${CLASSPATHSEP} ${BLAZE_LICENSEPATH}"
   ```
4. Stop and start the Interstage BPM Server. For instructions, refer to the *Interstage Business Process Manager Server Administration Guide*.

12.4.2 Integrating ILOG JRules

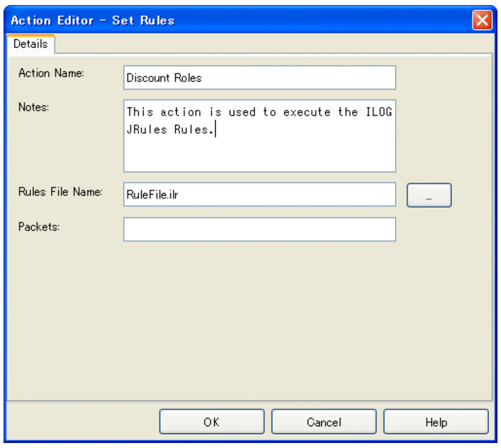
Prerequisite: You have a rules file that has been created with the ILOG JRules rules engine.

To integrate ILOG JRules:

1. Import the rules file (`*.ilr`) that you want to use in your Java Action to the following folder:

   `<Application Project>/dms/Attachments`

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. Expand **Rules Actions** and double click **ILOG JRules**.
5. In the **Action Editor - Set Rules** dialog, type a descriptive name and your notes for the Java Action.
6. Click the browse button and navigate to the rules file that you placed in the `attachments` subfolder of Interstage BPM Studio.



Action Name: Discount Roles
Notes: This action is used to execute the ILOG JRules Rules.
Rules File Name: RuleFile.ilr
Packets:

Figure 188: Integrating ILOG JRules

7. Type the name of one or more packets that identify the rule sets that need to be executed.
8. Click **OK**.

Note: It is not necessary to compensate this action using a compensation action, because changes made by this action are in Interstage BPM only and they will be rolled back after the process instance goes into error state. For details on compensation actions, refer to section *Dealing With Errors in Java Actions* on page 271.

12.5 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.

12.5.1 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.

Note: The following are disabled if the Send Email Java Action is created with the Interstage BPM Studio version 10.1 or prior versions.

- **From** field
- **Attachment** field
- **Attach Process Instance Attachments** checkbox

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**.
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 *Defining JavaScript Expressions* on page 371.

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**.

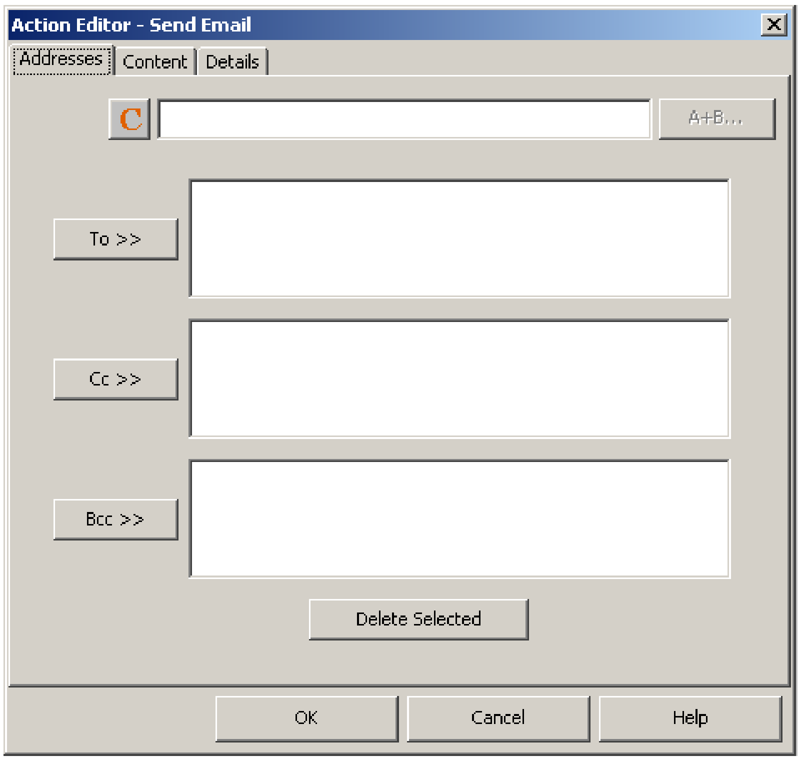


Figure 189: Using Addresses

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.

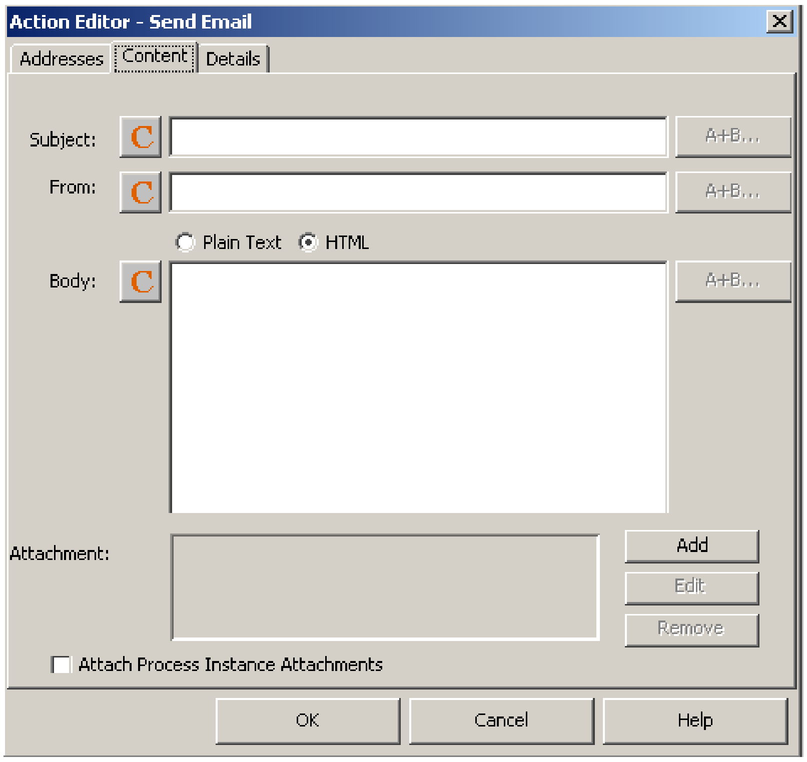


Figure 190: Specifying Email Content

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 *Defining JavaScript Expressions* on page 371 for more information about expression mode button.

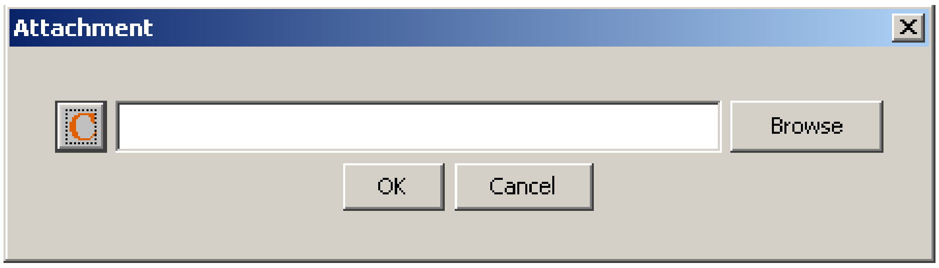


Figure 191: Attaching File

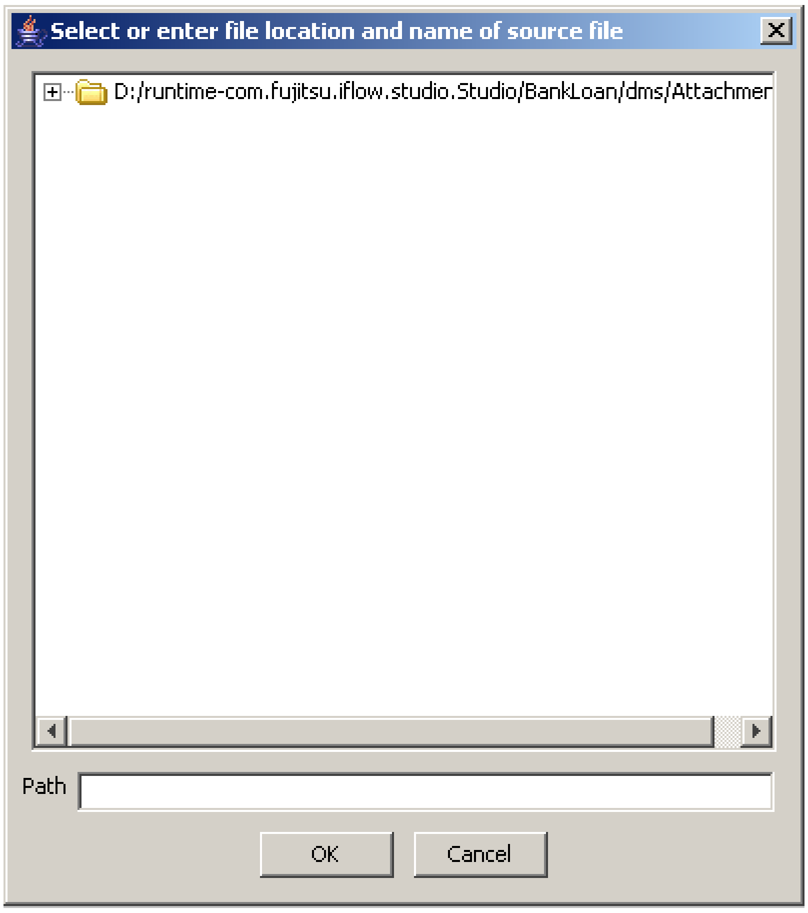

Select or enter file location and name of source file
D:/runtime-com.fujitsu.iflow.studio.Studio/BankLoan/dms/Attachmer
Path
OK
Cancel

Figure 192: Browsing for Attachment

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**.

12.6 Using Database Actions

Database Actions allow you to interact with external databases that are independent of Interstage BPM. You can retrieve data from a database and assign it to User Defined Attributes (UDAs). You can also send data back to the database, and you can delete data from the database.

Standard SQL queries are used to access the database.

The following sections explain how to configure Interstage BPM for Database Actions and how to use them.

12.6.1 Configuring Interstage BPM for Database Actions

Before you can use either of the Database Actions, you must set up a connection to the external database that you want to access. You can access any JDBC-compliant database.

To configure Interstage BPM for Database Actions:

1. In the Navigator view, right-click the Resources folder and select **Java Actions** >**Data Source** from the pop-up menu.

   A new `DataSourceDefinition.xml` file is created and stored in the Resources folder. The default file is automatically displayed in the Resources editor. The following example shows the default `DataSourceDefinition.xml` file:

```
[DataSourceDefinition.xml]

<DataSources>
     <!-- DataSource name="purchaseDB">
     </DataSource -->
</DataSources>
```

Note: You can create each resource file only once. After you have created a new `DataSourceDefinition.xml` file, the respective menu option will be disabled.

2. In the `DataSourceDefinition.xml` file, add a new `<DataSource>` section, specify the name of the data source that you want to access, and type in the properties required for the new data source.

   The name will be displayed in the dialogs of Interstage BPM Studio that you will use to define your Database Actions. The following example shows how to add a data source called `EmployeeDB`.

   The `DataSourceDefinition.xml` file then looks as follows (the new data source parameters are in bold below):

```
[DataSourceDefinition.xml]

<DataSources>
     <!-- DataSource name="purchaseDB">
     </DataSource -->

<DataSource name="EmployeeDB>
```

```
        <property name="UserName"
          type="String">EmployeeUserName</property>
        <property name="Password"
          type="String">EmployeePassword</property>
        <property name="URL"

type="String">jdbc:sqlserver://employeehostname:1433;databaseName=employeedb</property>

        <property name="DriverClassName">
         type="String">com.microsoft.sqlserver.jdbc.SQLServerDriver</property>

      </DataSource>
</DataSources>
```

Each new data source requires the following database connection properties:

- **Data Source Name**: Specifies the database to use for the new connection. When you specify a database name, the name must exist in the relational database directory on the server.

Note: If the specifies database does not exist, the connection fails.

- **User Name**: Name of database users required to log on a database instance and work with database objects.
- **Password**: Password required to log on a database instance and work with database objects.
- **URL**: Universal Resource Locator (URL) that specifies a particular type of database server (compatible with the local JDBC driver) and a particular host.
- **Driver Class Name**: The full Java class name of the JDBC driver to be used to connect to the database. The class must be present on the Classpath at the time you invoke this application.

Note: You can add multiple data sources to a Workflow Application project.

Note: When sending the process definition to an Interstage BPM Server, make the following adjustments. Be aware that these steps are also required when you deploy a Workflow Application project on the Interstage BPM Server:

1. Obtain the following information on the database to which you want to connect:
   - User name and password for a normal database user account
   - Database connection URL

     The following is a Microsoft SQL Server 2005 example:

     ```
     jdbc:sqlserver://<COMPUTER_NAME>:1433;databaseName=<yourdb>
     ```
   - Driver class

     The following is a Microsoft SQL Server 2005 example:

     ```
     com.microsoft.sqlserver.jdbc.SQLServerDriver
     ```
2. Add the data source to the data source definition file used by the Interstage BPM Server.

   ```
   <Interstage BPM Server Installation
   Directory>/server/instance/default/resources/DataSourceDefinition.xml
   ```

   Use the example currently in `DataSourceDefinition.xml` and the information that you gathered about your database.
3. If you are using the **Interstage BPM Enterprise Edition for WebLogic**:
   a. Check whether the jar files of the JDBC driver are already included in your WebLogic installation. If they are included, no further action is required.
   b. If the JDBC driver is not included in your WebLogic installation, copy it to the following folder: `<Interstage BPM Server Domain>/lib`. Then, restart the Application Server.
4. If you are using the **Interstage BPM Enterprise Edition for WebSphere**, copy the jar files of the JDBC driver to the following folder: `<Interstage BPM Server Installation Directory>/server/instance/default/lib`. Then, restart the Application Server.
5. If you are using the **Interstage BPM Enterprise Edition for JBoss**, copy the jar files of the JDBC driver to the following folder: `<JBoss Installation Directory>/server/ibpm/lib`. Then, restart the Application Server.
6. If you are using the **Interstage BPM Enterprise Edition for Interstage**, copy the jar files of the JDBC driver to the following folder: `<Interstage BPM Server Installation Directory>/server/instance/default/lib/ext`. Then, restart the Application Server.

12.6.2 Retrieving Data from External Databases

Prerequisites:

- You have registered the external database as a data source. For details, refer to *Configuring Interstage BPM for Database Actions* on page 323.
- You know the data model of the external database, and you know how to write SQL statements.
- The process definition has one or more User Defined Attributes (UDAs) to which the query results can be assigned.

You can query an external database and assign the results of the query as the value of UDAs.

To retrieve data from an external database:

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 **Database Actions**.
4. Double click **Select SQL Java Action**.
5. Choose the data source that you defined in `DataSourceDefinition.xml` from the **Choose Data Source** drop-down list.

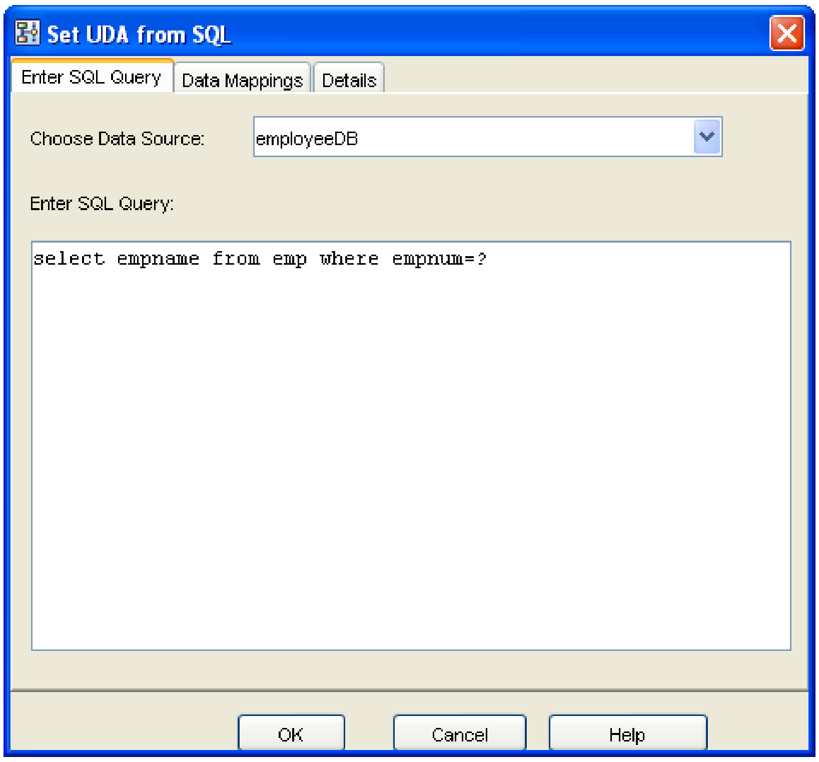
Note: If the data source you defined does not appear on this list, check your configuration in `DataSourceDefinition.xml`.

6. Type the SQL select statement.

   In the SQL statement, you can use input variables denoted with a question mark. In a later step, each of the input variables will be related to one of your UDAs.

   The following example might be used to find an employee's name from an employee number. `emp` is the database table being queried:

   ```
   select empname from emp where empnum=?
   ```



Choose Data Source: employeeDB

Enter SQL Query:

```
select empname from emp where empnum=?
```

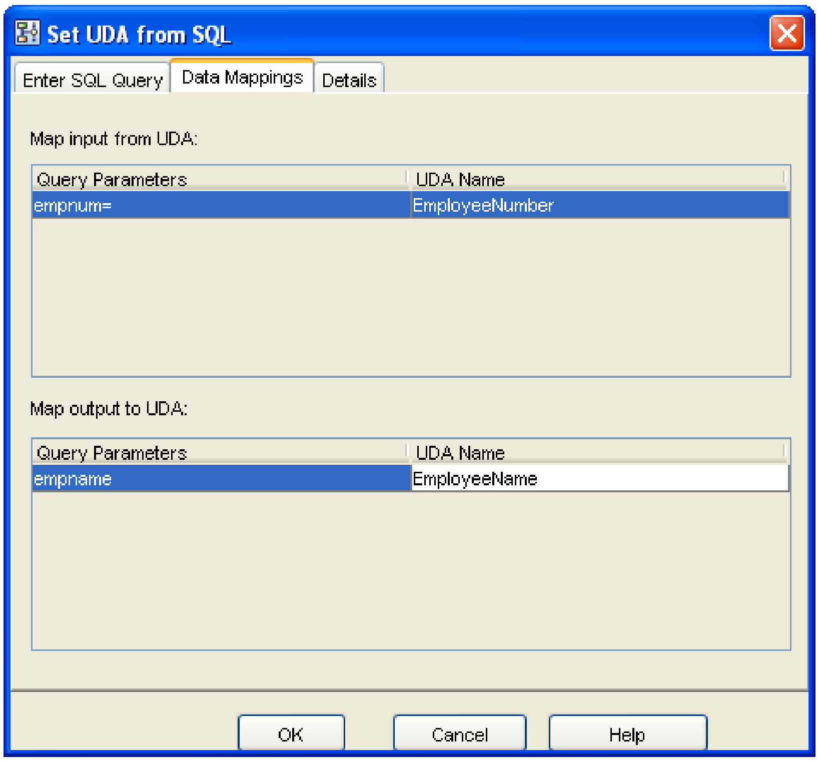
Figure 193: Specifying the Data Source and the SQL Select Statement

7. Select the **Data Mappings** tab.

Note that when you are changing an SQL query, the information on the **Data Mappings** tab is cleared.

8. If your SQL query contains input variables, map them to the corresponding UDAs in the **Map input from UDA** area.
9. In the **Map output to UDA** area, select the UDAs that you want to use for the results of your SQL query.

   In the example below, a UDA called `EmployeeNumber` is used in the SQL query, and a UDA called `EmployeeName` contains the result.



Map input from UDA:

Query Parameters	UDA Name
empnum=	EmployeeNumber

Map output to UDA:

Query Parameters	UDA Name
empname	EmployeeName

Figure 194: Specifying UDAs for the Query and Mapping Query Results to UDAs

Note: If your SQL query returns multiple results, only one of them will be assigned to the UDA.

10. On the **Details** tab, type a descriptive name and your notes for the Java Action.
11. Click **OK**.

12.6.3 Updating, Inserting, or Deleting Data in External Databases

Prerequisites:

- You have registered the external database as a data source. For details, refer to *Configuring Interstage BPM for Database Actions* on page 323.
- You know the data model of the external database, and you know how to write SQL statements.

You can update the data in external databases with information stored in User Defined Attributes (UDAs). You can also insert or delete data in the database.

To update, insert, or delete data in an external database:

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 **Database Actions**.
4. Double click the database action that you want to be performed, for example **Update SQL Java Action**. The **Update SQL from UDA** dialog is displayed.
5. Choose the data source that you defined in `DataSourceDefinition.xml` from the **Choose Data Source** drop-down list, in the **Enter Update Statement** tab.

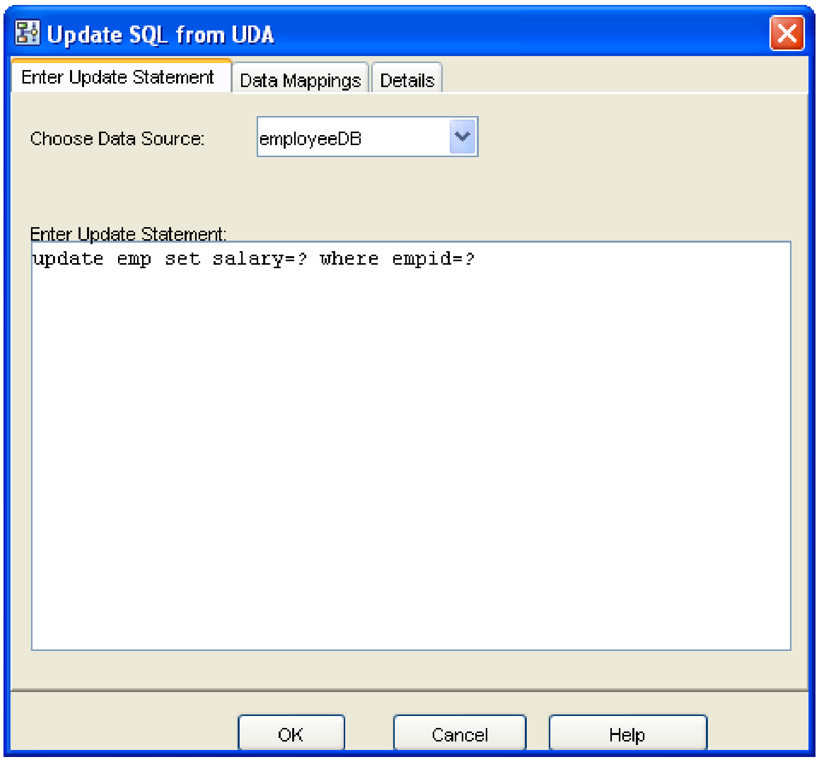
Note: If the data source you defined does not appear on this list, check your configuration in `DataSourceDefinition.xml`.

6. Type the SQL statement.

   In the SQL statement, you can use input variables denoted with a question mark. In a later step, each of the input variables will be related to one of your UDAs.

   The following sample shows an SQL update statement that might be used to update the salary of an employee in an external database. `emp` is the database table that stores the salary information.

   ```
   update emp set salary=? where empid=?
   ```



Choose Data Source: employeeDB

Enter Update Statement:

```
update emp set salary=? where empid=?
```

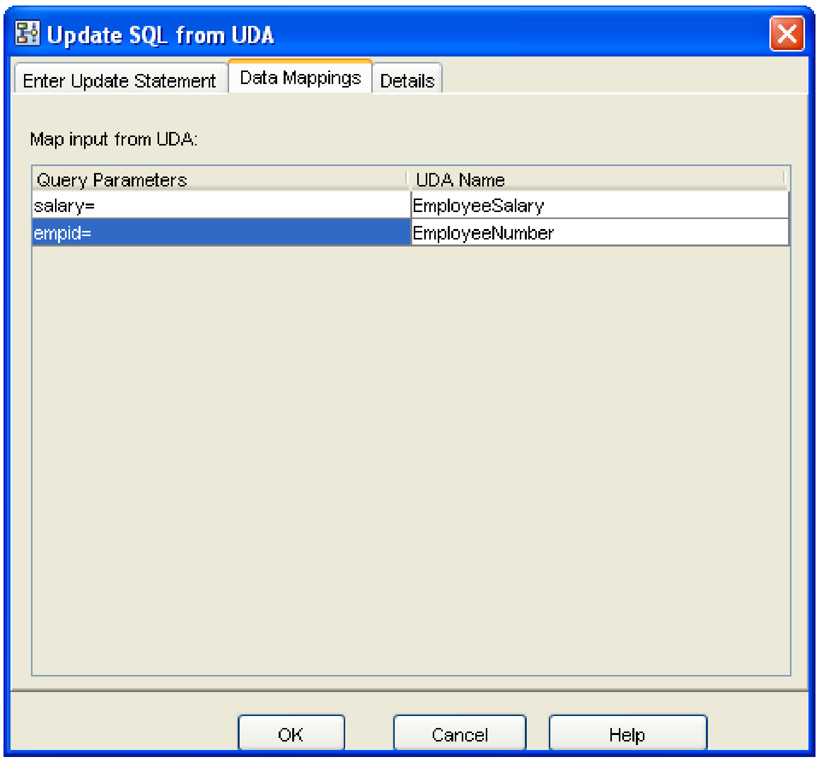
Figure 195: Specifying the Data Source and an SQL Statement

7. Select the **Data Mappings** tab.

   Note that when you are changing an SQL statement, the information on the **Data Mappings** tab is cleared.

8. Select the UDAs that supply the values to be used in your SQL statement.

   In the example below, a UDA called `EmployeeSalary` stores the salary that is to be updated in the external database. The `EmployeeName` UDA is used to determine the employee whose data are to be updated.



Map input from UDA:

Query Parameters	UDA Name
salary=	EmployeeSalary
empid=	EmployeeNumber

Figure 196: Specifying UDAs that Supply Values

9. On the **Details** tab, type a descriptive name and your notes for the Java Action.
10. Click **OK**.

12.7 Using Integration Actions

Integration Actions allow you to access external functions from within your process definition.

You can integrate any system that exposes its functions as a Web Service, and you can execute remote commands.

12.7.1 Executing Remote Commands

Using the `Remote Command Invoker` Java Action, you can execute a command on any computer that can be accessed by the Interstage BPM Server. You can run any command that can be executed from a command prompt.

A Remote Command Server is required to run on the computer upon which you want to execute the command. Your Interstage BPM installation includes a Remote Command Server. However, it is not enabled as a default.

To execute a remote command:

1. If you want to execute a command on the computer where the Interstage BPM Server is installed, make sure that the Remote Command Server runs on that computer.
   - **On Windows**, you can use the `StartJACommandServer.bat` file to start the Remote Command Server. This file can be found in the `<Interstage BPM Server Installation Directory>/server/instance/default/bin` folder.

     Usage: `StartJACommandServer.bat <port> <hostname>`
   - **On UNIX or Linux**, run the following Java program:

     `java com.fujitsu.iflow.utilities.JACommandServer <port>, <hostname>`

   `<port>` defines the port that the Remote Command Server will listen to.

   `<hostname>` is the name of the computer that is allowed to connect to the Remote Command Server.

   When running the following sample on Windows, a Remote Command Server will be started on the local host (that is, on the Interstage BPM Server) and listen to port 5037:

   ```
   cd <Interstage BPM Server Installation
   Directory>/server/instance/default/bin
   StartJACommandServer.bat 5037 localhost
   ```

2. **Windows only**: If you want to execute a command on a remote computer (that is a computer where the Interstage BPM Server is not installed):

   a) Copy the following files from your Interstage BPM installation to the remote computer:

   `<Interstage BPM Server Installation Directory>/server/instance/default/bin/StartJACommandServer.bat`

   `<Interstage BPM Server Installation Directory>/client/lib/iFlow.jar`

   b) Edit `StartJACommandServer.bat` on the remote computer. Make sure that the `IFLOW_DIR` variable points to the folder on the remote computer to which you copied the `/client/lib/iFlow.jar` file.

   For example, if you copied the file to a folder called `C:\RemoteCommandServer`, you would specify the following:

   `SET IFLOW_DIR=C:\RemoteCommandServer`

   c) Start the Remote Command Server by running the following batch file:

   `StartJACommandServer.bat <port> <hostname>`

   `<port>` defines the port that the Remote Command Server will listen to.

   `<hostname>` is the name of the Interstage BPM Server that is allowed to connect to the Remote Command Server. Alternatively, you can specify the IP address of the Interstage BPM Server.

   When running the following sample, a Remote Command Server will be started and listen to port 5037. An Interstage BPM Server called `ibpmhost` will be allowed to connect to the Remote Command Server.

   ```
   StartJACommandServer.bat 5037 ibpmhost
   ```

3. Add a User Defined Attribute (UDA) to your process definition and specify the command to be executed as its value.

   For example, you might want to open the Notepad editor as a remote command. You would add a UDA called `RemoteCommand` and set its value to the following command line:

   ```
   cmd /c "C:\\WINDOWS\\System32\\notepad.exe"
   ```

Note: When specifying the command's path, make sure that the path is valid on the computer where the command will be executed.

4. 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.
5. 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.
6. Expand **Integration Actions** and double click **Remote Command Invoker**.
7. In the **Action Editor - Remote Command** dialog, type a descriptive name and your notes for the Java Action.
8. From the **Command UDA** drop-down list, select the UDA that contains the remote command.
9. If additional input is needed after the execution of the remote command, type the input in the **Additional Input** field.

   Example:

   You can open Notepad by opening a command prompt, then starting Notepad from the command prompt. You would put `cmd` in your command UDA and the following in the **Additional Input** field:

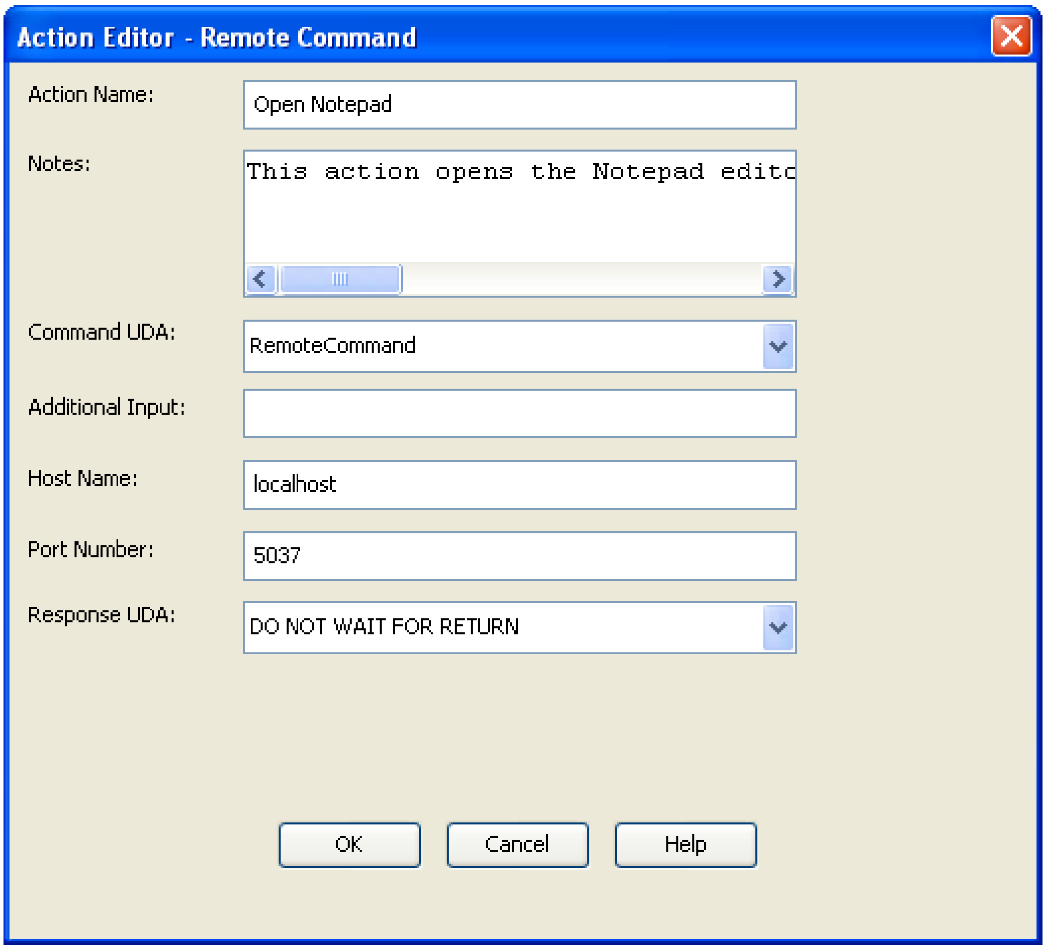
   ```
   C:\\WINDOWS\\System32\\notepad.exe
   ```

Note: The complete path must be included in this case.

   When your Java Action runs, first `cmd` executes, then Notepad is started after cmd has executed.

10. Type the hostname and port of the Remote Command Server that is running on the computer upon which the command will be executed.
11. From the **Response UDA** drop-down list, select the UDA to collect the response if there is any.

If the Java Action shall not wait for a return value, make sure that `DO NOT WAIT FOR RETURN` is selected.



Action Name: Open Notepad

Notes: This action opens the Notepad edito

Command UDA: RemoteCommand

Additional Input:

Host Name: localhost

Port Number: 5037

Response UDA: DO NOT WAIT FOR RETURN

Figure 197: Executing a Remote Command

12. Click **OK**.

12.7.2 Calling a Web Service

Prerequisites:

- You are familiar with the key concepts of Web Services.
- You have the following information about the Web Service:
  - Purpose of the Web Service
  - The URL of the WSDL document that describes the operations supported by the Web Service
  - Optional: The Inquiry URL of a UDDI Registry that you can query for the location of the WSDL document
  - Meaning and data type of the operation's input parameters
  - Meaning and data type of the operation's return value

  Contact the Web Service provider to obtain this information.

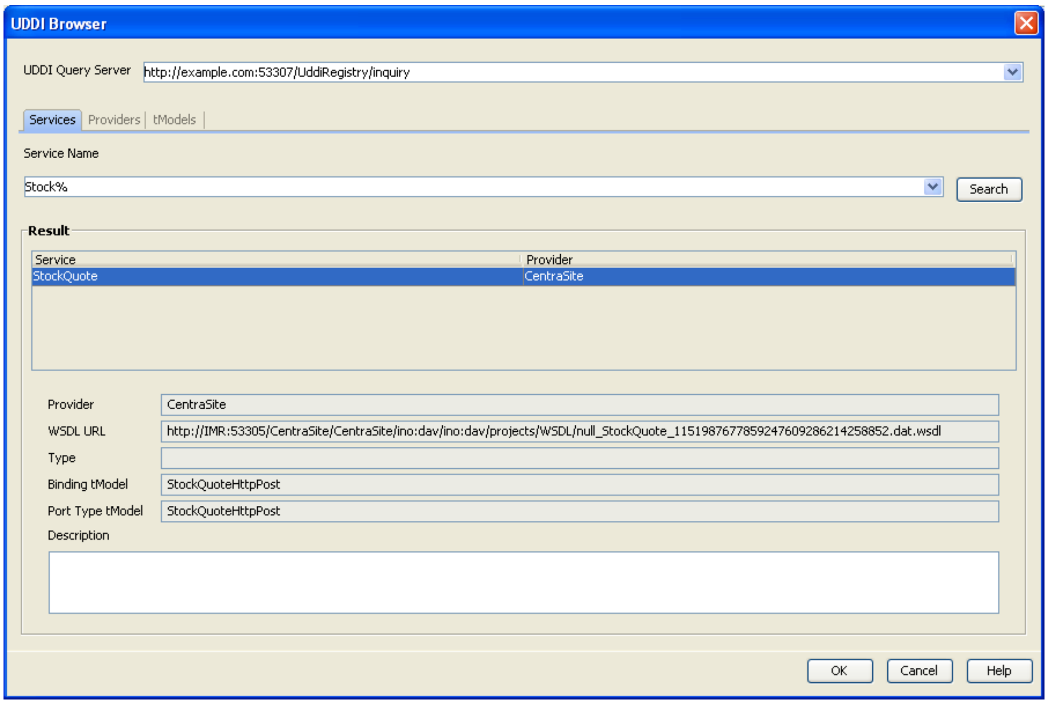
- You have specified User Defined Attributes (UDAs) for the parameters that you want to pass to the operation.
- You have specified UDAs that are to store the operation's return value.
- You are familiar with XPath expressions and know how to write them.

You can call a Web Service from Interstage BPM using the `Web Service Call` Java Action. You can assign `Web Service Call` to any node and to the process definition itself. When using Web Service Nodes, this is the only type of Java Action that can be assigned to them.

To call a Web Service:

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 **Integration Actions** and double click **Web Service Call**.
4. Type a descriptive name and your notes for the Java Action, in the **Action Editor - Web Services** dialog.
5. Do one of the following:
   - If you know the URL of the WSDL document, type it in the **WSDL Location** field.
   - If you don't know the URL, click **Browse** to query a UDDI Registry for the WSDL document.
   - If you know the URL of the WSDL document and you would like to change it dynamically, you can define it by the UDA or the Application Variable.

In this case, you set URL of WSDL to the value of UDA or Application Variable, and select it with WSDL Location.



UDDI Query Server: http://example.com:53307/UddiRegistry/inquiry

Service Name: Stock%

Service	Provider
StockQuote	CentraSite

Provider	CentraSite
WSDL URL	http://IMR:53305/CentraSite/CentraSite/ino:dav/ino:dav/projects/WSDL/null_StockQuote_1151987677859247609286214258852.dat.wsdl
Type	
Binding tModel	StockQuoteHttpPost
Port Type tModel	StockQuoteHttpPost
Description	

Figure 198: Searching for a WSDL Document

Note: You can configure default UDDI query servers to be displayed in the **UDDI Browser** dialog. To do so:

1. Exit Interstage BPM Studio.
2. Open the `twf.ini` configuration file located in the `<Interstage BPM Studio Installation Directory>\ibpm\Data\bin` folder.
3. Add the following entry:

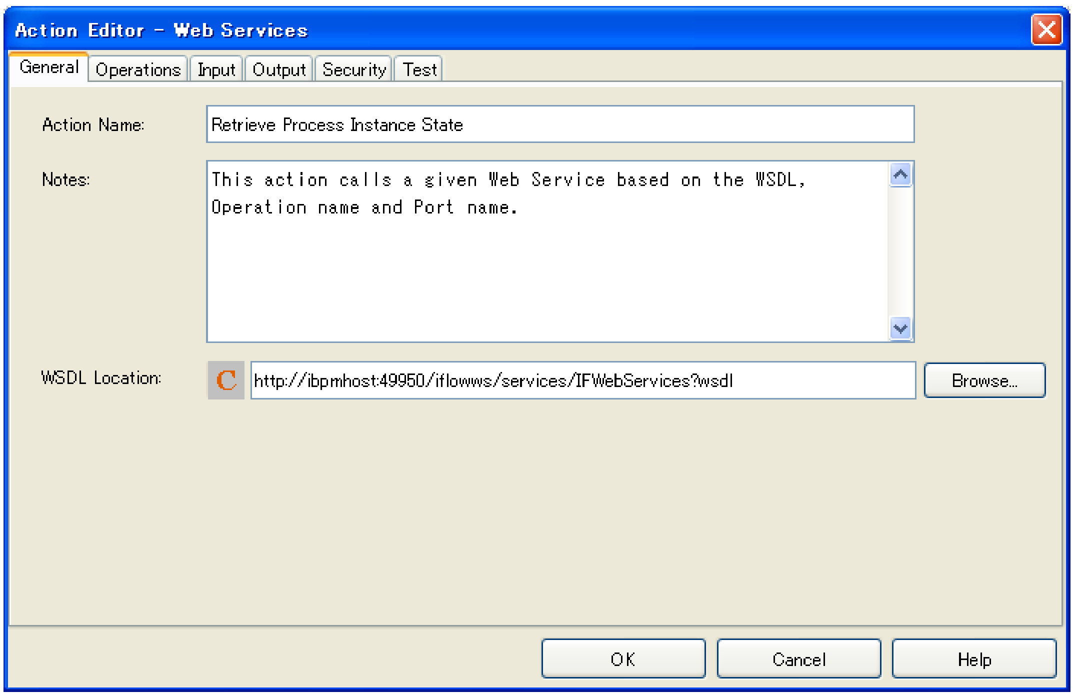
```
UDDIInquiryURL=<URL of a UDDI query server>[,<URL of a UDDI query server>
...]
```

4. Save the configuration file.
5. Start Interstage BPM Studio.

Note the following when querying a UDDI Registry:

- Type the Inquiry URL in the **UDDI Query Server** field.
- You can specify the Web Service name, Web Service provider, or tModel as search criteria.
- You may use the percent symbol (%) as a wildcard for any character or set of characters in the search criteria.

- Once you have found the Web Service that you want to use, make sure that the URL of the WSDL document is displayed in the **WSDL URL** field. Then, click **OK**.



Action Name: Retrieve Process Instance State

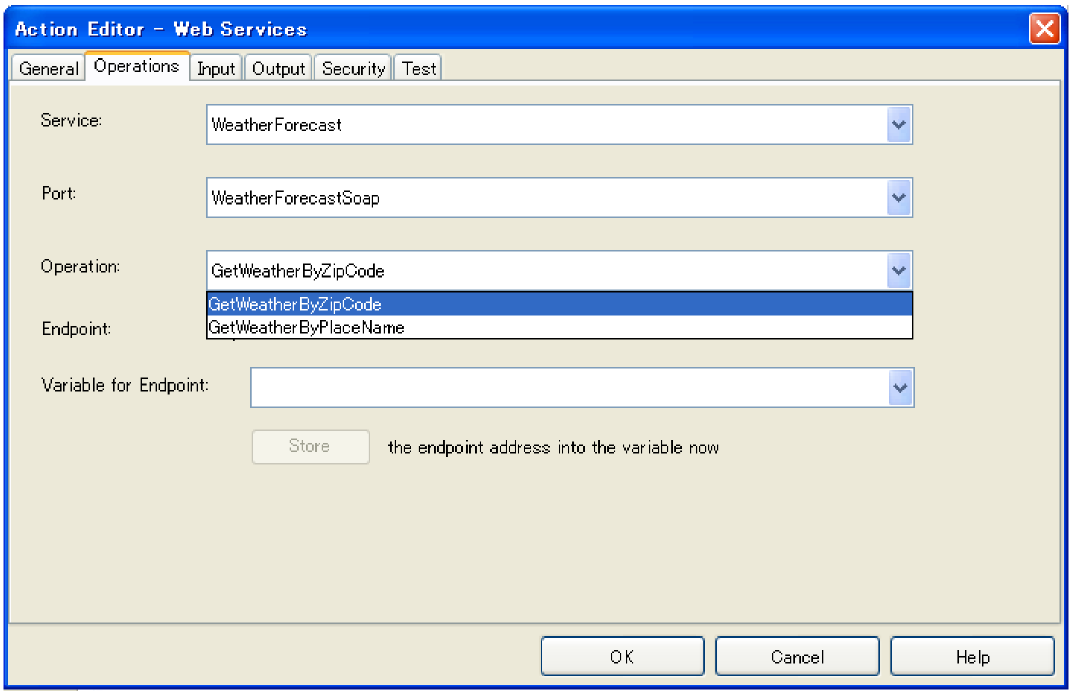
Notes: This action calls a given Web Service based on the WSDL, Operation name and Port name.

WSDL Location: http://ibpmhost:49950/iflowws/services/IFWebServices?wsdl

Figure 199: Specifying the Location of the WSDL Document

6. Specify the operation that you want to use:
   a) Select the **Operations** tab.

b) Select one of the operations that are available on the given Web Service and port.



Service: WeatherForecast

Port: WeatherForecastSoap

Operation: GetWeatherByZipCode

GetWeatherByZipCode
GetWeatherByPlaceName

Endpoint:

Variable for Endpoint:

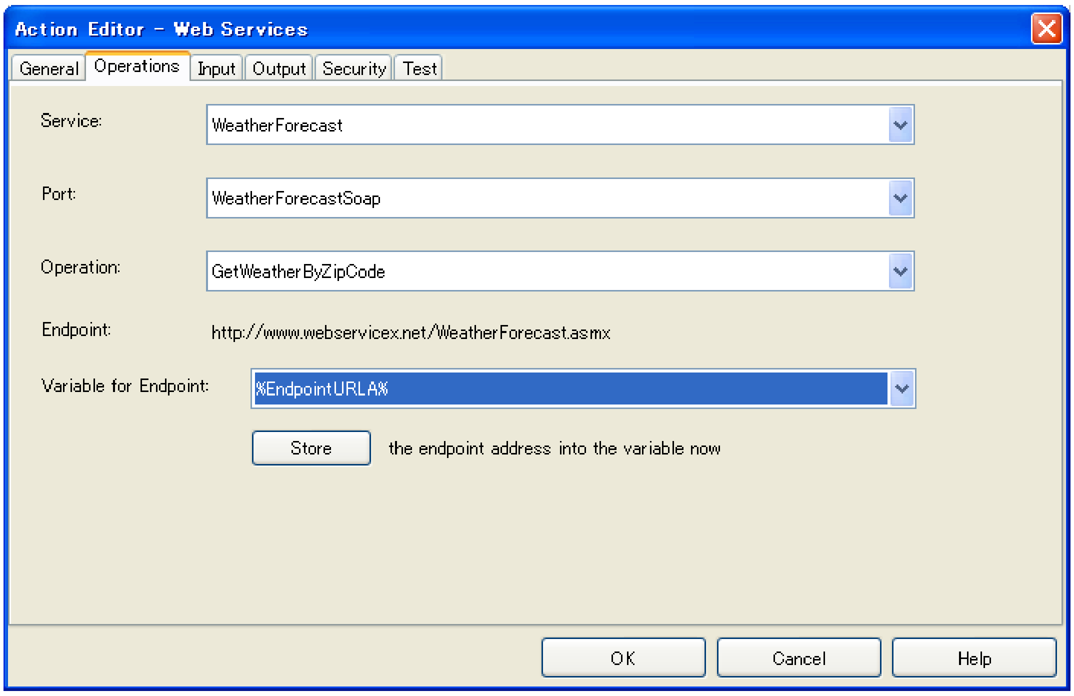
the endpoint address into the variable now

Figure 200: Selecting an Operation

c) You can specify the endpoint URL of the web service dynamically with a value of a UDA or an application variable. To specify the variable, select a UDA or an application variable in the **Variable for Endpoint** list.

Note:
- The endpoint URL which is specified by the given WSDL document is used if the **Variable for Endpoint** list is not set.
- If you want to set UDA in the **Variable for Endpoint** list, only STRING type UDA can be set.

The URL displyed in the **Endpoint** field can be set as a default value of a UDA or an application variable. Select the **Variable for Endpoint** list, click **Store**, and click **OK**.



Service: WeatherForecast

Port: WeatherForecastSoap

Operation: GetWeatherByZipCode

Endpoint: http://www.webservicex.net/WeatherForecast.asmx

Variable for Endpoint: %EndpointURLA%

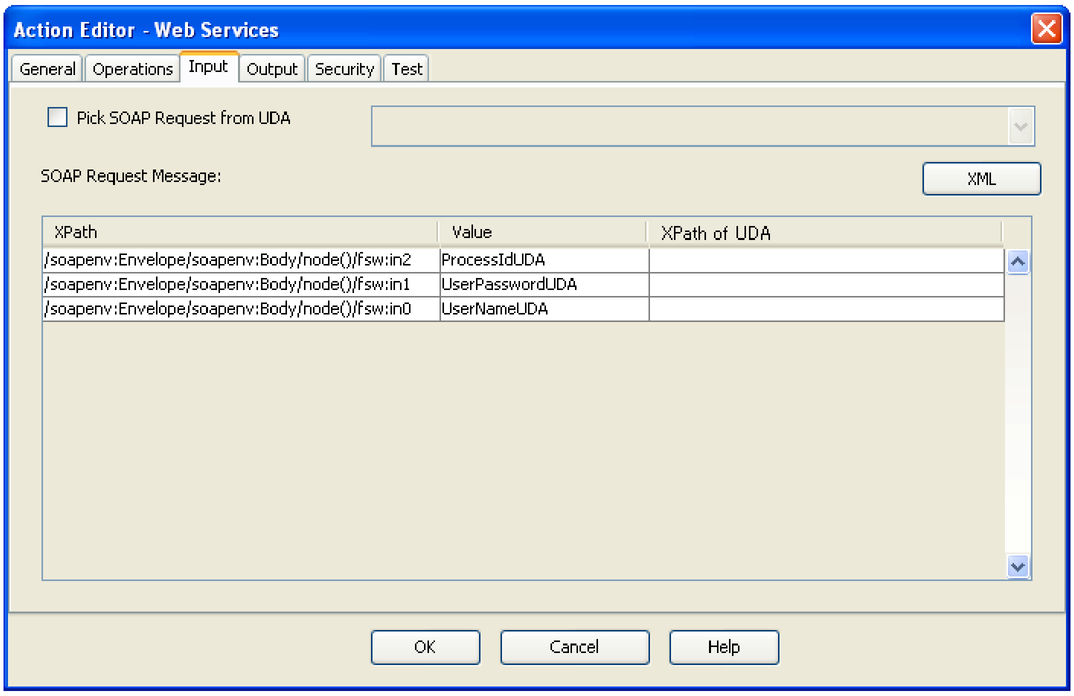
Store the endpoint address into the variable now

Figure 201: Specifying variable for Endpoint

7. Specify the input parameters for the operation:
   a) Select the **Input** tab.

      This tab shows the SOAP request message that will be sent to the Web Service. This message contains the input parameters that the operation needs. The **XPath** column shows the parameters as XPath expressions.

b) In the **Value** column, specify the UDAs that contain the values to be passed to the operation.



XPath	Value	XPath of UDA
/soapenv:Envelope/soapenv:Body/node()/fsw:in2	ProcessIdUDA	
/soapenv:Envelope/soapenv:Body/node()/fsw:in1	UserPasswordUDA	
/soapenv:Envelope/soapenv:Body/node()/fsw:in0	UserNameUDA	

Figure 202: Specifying the Operation's Input Parameters

c) When you specify UDA of type CUSTOM in **Value** column, from the drop-down list in XPath column, you can select the corresponding XPath which indicates the location of the value to pass the operation. This XPath field is enabled only when the UDA of type CUSTOM is selected.

After selecting the XPath, it is converted to the following JavaScript format: `{{JsXml}}format`. This XPath is saved to the process definition. For example, the XPaths given below are successfully converted to the `{{JsXml}}format` JavaScript format:

- `/myPurchase/Order/items[2]/item/partNo`
- `/po:order/po:locationId/po:postInfo`

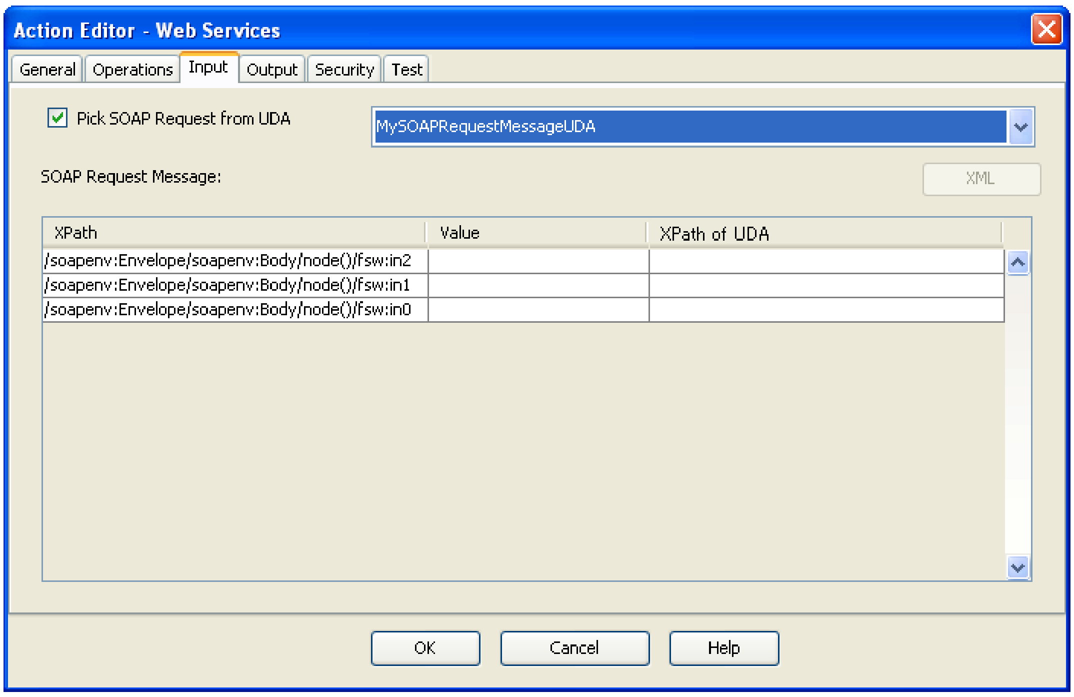
If this conversion is not done, the XPath is displayed with `{{JsXml}}format` in the **Value** column.

For an explanation of advanced options, refer to section *Web Services: Special Tags for SOAP Request Messages* on page 342.

8. If the SOAP request message is constructed by an external application, do the following:

a) Make sure that you have added a UDA that contains the SOAP request message.

b) On the **Input** tab, select **Pick SOAP Request from UDA**. From the drop-down list, select the UDA that contains the SOAP request message. The list only contains UDAs of type STRING, XML, or CUSTOM.



✓ Pick SOAP Request from UDA: MySOAPRequestMessageUDA

SOAP Request Message:

XPath	Value	XPath of UDA
/soapenv:Envelope/soapenv:Body/node()/fsw:in2		
/soapenv:Envelope/soapenv:Body/node()/fsw:in1		
/soapenv:Envelope/soapenv:Body/node()/fsw:in0		

Figure 203: SOAP Request Message Supplied by External System

In this case, the message shown in the **SOAP Request Message** area is not used.

Note: If a CUSTOM UDA is selected, XPath expressions specifying only leaf nodes (e.g. text and attribute nodes) will be shown in the XPath drop-down list.

9. Map the response of the Web Service to UDAs:
   a) Select the **Output** tab.
   b) Click in the **UDA Name** field and select the UDA that is to store the return value.

      The Web Service sends the operation's return value as a SOAP response message.
   c) Click in the **XPath of UDA** field and select an XPath expression of the UDA that stores the return value, from the drop-down list.

      The XPath field is activated only if you have selected a UDA of type XML or CUSTOM.
   d) Optional: If you want to edit the XPath expression that you selected in the drop-down list in **XPath of UDA**, click the **...** button located 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.

Note: The XPath expressions related to the selected XML or CUSTOM 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 XML 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.

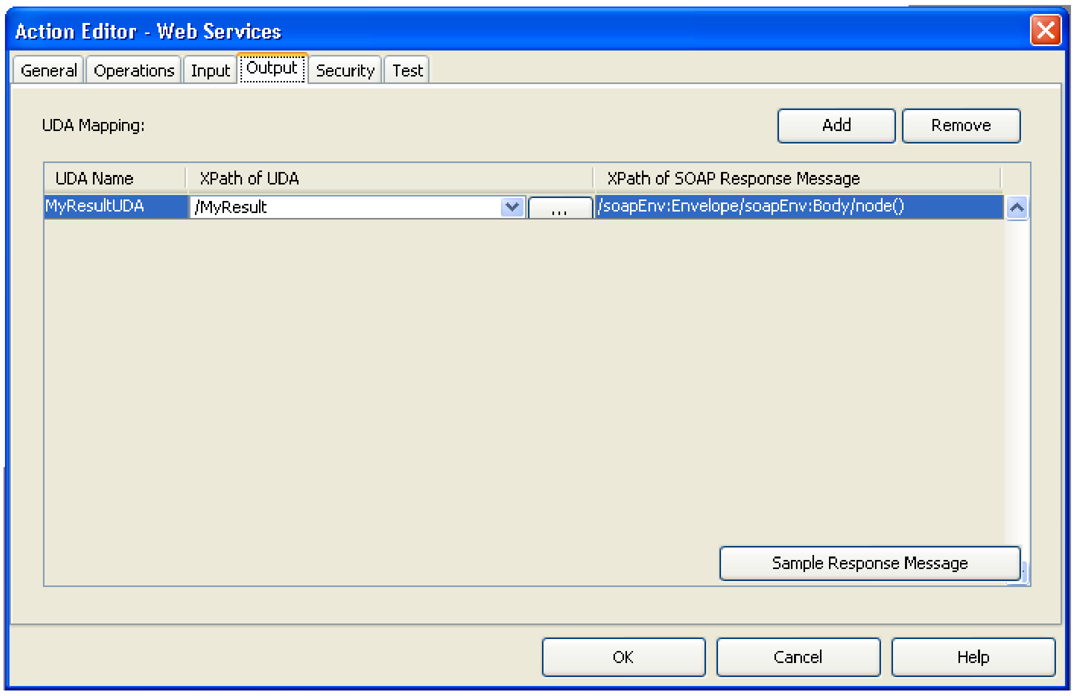
Note: **XPath Editor** only validates the syntax of the XPath. It does not check if the edited XPath expression exists or not.

e) In the **XPath of SOAP Response Message** field, specify where the return value can be found in the SOAP response message.

f) If you need to add additional rows, click **Add**.

If the return value is complex, you may need to map it to multiple UDAs. For example, if you call a Web Service that returns flight data, you need multiple UDAs to store the flight number, carrier, flight date, and so on.

g) If you want to remove a mapping, select it and click **Remove**.



UDA Name	XPath of UDA	XPath of SOAP Response Message
MyResultUDA	/MyResult	/soapEnv:Envelope/soapEnv:Body/node()

Figure 204: Mapping Return Values to UDAs

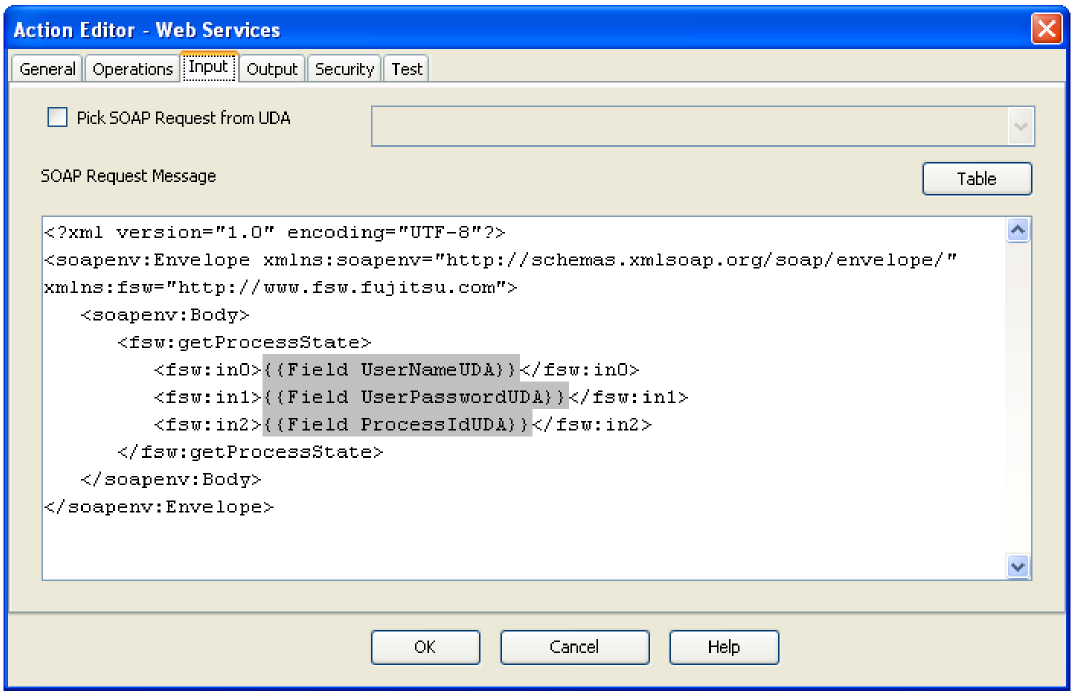
Note: If you have no information about the structure of the SOAP response message, test the Web Service call to obtain a sample message. Use this message to determine the XPath expression that you need.

10. If the Web Service can be accessed with a user account only:

a) Select the **Security** tab.

b) Type the user name and the password.

11. If you want to test the Web Service call with sample values, do the following:
    a) Select the **Input** tab.
    b) Click **XML** to switch to the XML view.
    c) Replace the `{{Field <UDA>}}` tags by sample values.
       The figure below shows a SOAP request message before the replacement.



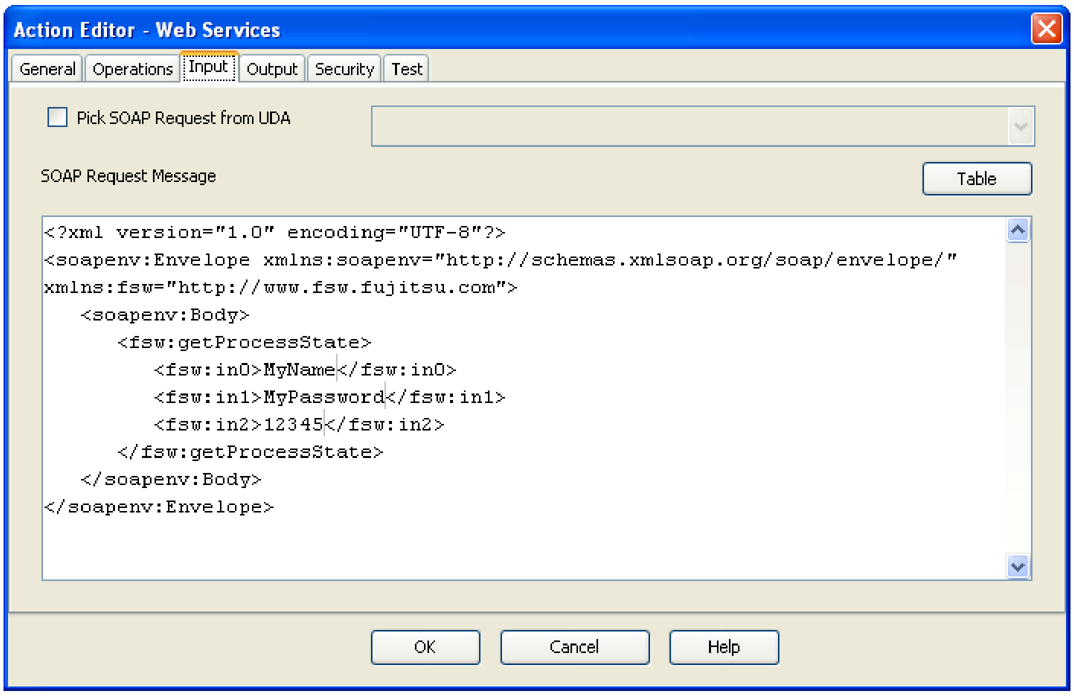
Pick SOAP Request from UDA

SOAP Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:fsw="http://www.fsw.fujitsu.com">
   <soapenv:Body>
      <fsw:getProcessState>
         <fsw:in0>{{Field UserNameUDA}}</fsw:in0>
         <fsw:in1>{{Field UserPasswordUDA}}</fsw:in1>
         <fsw:in2>{{Field ProcessIdUDA}}</fsw:in2>
      </fsw:getProcessState>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 205: SOAP Request Message with UDAs

This figure shows the SOAP request message after the replacement.



```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:fsw="http://www.fsw.fujitsu.com">
   <soapenv:Body>
      <fsw:getProcessState>
         <fsw:in0>MyName</fsw:in0>
         <fsw:in1>MyPassword</fsw:in1>
         <fsw:in2>12345</fsw:in2>
      </fsw:getProcessState>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 206: SOAP Request Message with Sample Values for Testing

d) Select the **Test** tab and click **Test**.

e) Check if the SOAP response message shows the result that you expected. If an error occurs, check your input data and repeat the test.

f) On the **Input** tab, restore the SOAP request message to its original state. To do so, right click the value and select the original expression.

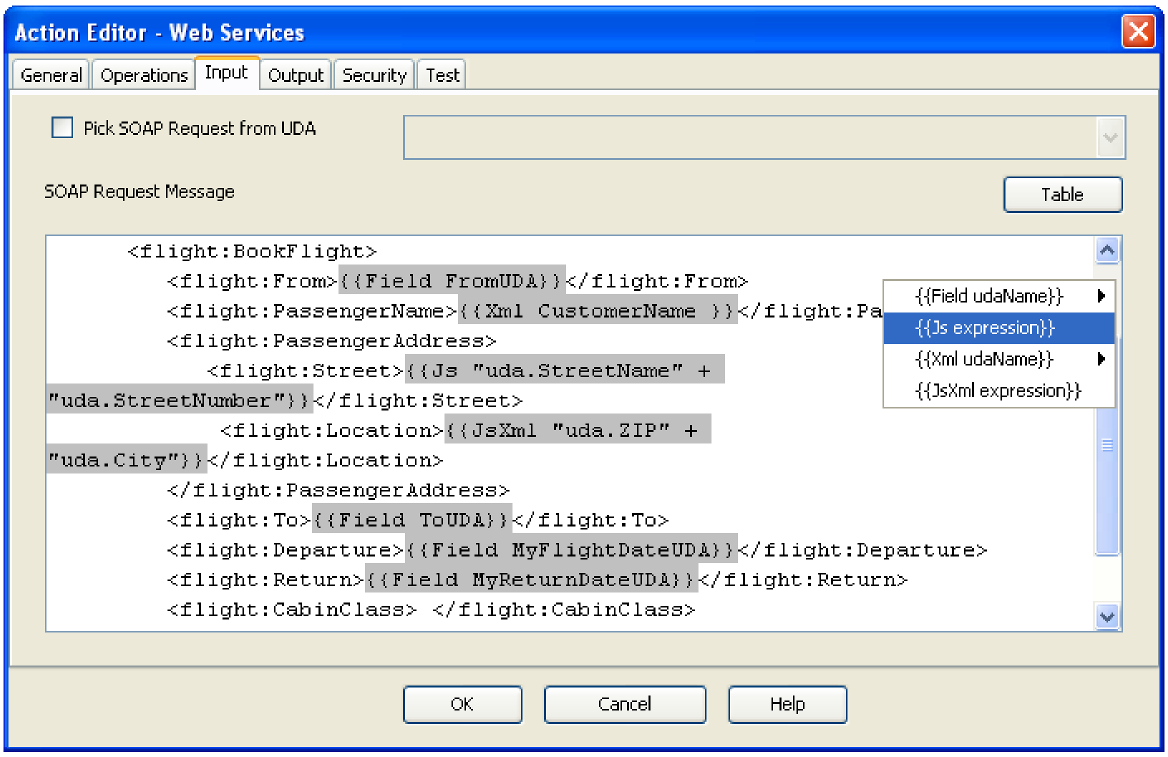
12. Click **OK**.

Note: If you want to call a Web Service on the Internet and you access the Internet through a Proxy Server, you must configure Interstage BPM Studio to use the Proxy Server. To do so, set the `WSHttpProxyHost` and `WSHttpProxyPort` parameters in the `twf.ini` configuration file. The configuration file is located in the `<Interstage BPM Studio Installation Directory>\ibpm\Data\bin` folder. The parameters for the Interstage BPM Server are also stored in the `IBPMProperties` table in the database. For more information, refer to the *Interstage Business Process Manager Server Administration Guide*.

12.7.3 Web Services: Special Tags for SOAP Request Messages

When specifying the SOAP request message, you can use special tags for parameters that you want to be passed to the Web Service.

The **Input** tab has an XML view and a Table view. You can add special tags in both views. However, adding special tags in XML view is easier because you can select them from a pop-up menu. To display the pop-up menu, click **XML** and then right click the **SOAP Request Message** area.



```
      <flight:BookFlight>
         <flight:From>{{Field FromUDA}}</flight:From>
         <flight:PassengerName>{{Xml CustomerName }}</flight:Pa
         <flight:PassengerAddress>
            <flight:Street>{{Js "uda.StreetName" +
"uda.StreetNumber"}}</flight:Street>
             <flight:Location>{{JsXml "uda.ZIP" +
"uda.City"}}</flight:Location>
         </flight:PassengerAddress>
         <flight:To>{{Field ToUDA}}</flight:To>
         <flight:Departure>{{Field MyFlightDateUDA}}</flight:Departure>
         <flight:Return>{{Field MyReturnDateUDA}}</flight:Return>
         <flight:CabinClass> </flight:CabinClass>
```

Figure 207: Adding Special Tags for SOAP Request Messages

Before the SOAP request message is sent, these tags are processed and replaced by the values that need to be passed. The following list explains available tags and how they are processed.

- `{{Field <UDAName>}}`

  The tag is replaced by the value of the User Defined Attribute (UDA). The value is XML encoded.

  XML encoding affects characters that have a special meaning in XML. These characters are: less than (<), greater than (>), ampersand (&) and double quotes ("). XML encoding ensures that these characters are transformed to `&lt;`, `&gt;`, `&amp;`, and `&quot;`.

- `{{Xml <UDAName>}}`

  The tag is replaced by the value of the UDA. The value is not XML encoded.

  Use this tag if the value is an XML fragment that you want to pass as such.

- `{{Js <JavaScriptExpression>}}`

  The JavaScript expression is evaluated and the tag is replaced by the evaluation result. The evaluation result is XML encoded.

- `{{JsXml <JavaScriptExpression>}}`

  The JavaScript expression is evaluated and the tag is replaced by the evaluation result. The evaluation result is not XML encoded.

Use this tag if the JavaScript expression contains XML fragments or elements of CUSTOM UDAs that you want to pass as such.

Example of {{Field <UDAName>}}

The following example shows how to pass the value of UDA `Company` to the Web Service:

`<flight:Comp>`**`{{Field Company}}`**`</flight:Comp>`

If the company's name is `Good & Fast`, it is passed as `Good &amp; Fast`.

Example of {{Xml <UDAName>}}

Say you want to call a Web Service for flight booking and need to pass the passenger's name. It is stored as an XML fragment in UDA `CustomerName`:

`<LastName>Jones</LastName><FirstName>Elizabeth</FirstName>`

To pass this XML fragment, you need to include the following tag:

`<flight:PassengerName>`**`{{Xml CustomerName}}`**`</flight:PassengerName>`

Before the message is passed, the tag is replaced as follows:

`<flight:PassengerName>`

`<LastName>Jones</LastName><FirstName>Elizabeth</FirstName>`

`</flight:PassengerName>`

Example of {{Js <JavaScriptExpression>}}

The Web Service needs the address of the passenger. The street name and street number are stored in separate UDAs, but need to be combined before being passed. This is achieved with the following JavaScript expression:

`<flight:Street>`**`{{Js "uda.StreetNumber " + "uda.StreetName"}}`**`</flight:Street>`

If `StreetName` has the value `Broad & Long Street` and `StreetNumber` is `123`, the JavaScript expression evaluates to:

`<flight:Street>`**`123 Broad &amp; Long Street`**`</flight:Street>`

Example of {{JsXml <JavaScriptExpression>}}

Say your process definition stores the parts of an address in XML format, for example `<ZIP>80000</ZIP>` or `<City>Munich</City>`. If you need to combine those parts and pass them in XML format, you could use the following JavaScript expression:

`<flight:Location>`**`{{JsXml "uda.ZIP" + "uda.City"}}`**`</flight:Location>`

When a value of the ZIP UDA is `<ZIP>80000</ZIP>` and a value of the City UDA is `<City>Munich</City>`, this JavaScript expression evaluates to:

`<flight:Location>`**`<ZIP>80000</ZIP><City>Munich</City>`**`</flight:Location>`, where ZIP is the postal code and Munich is the name of the city of Munich.

You can set elements of a UDA of type CUSTOM as part of the SOAP request message. However, this UDA is only a leaf element, which does not have any other elements. In this case, you can use `{{JsXml <JavaScriptExpression>}}` or `{{Js <JavaScriptExpression>}}`.

Consider the `myDestination` UDA of type CUSTOM:

```
<Destination>
  <ID>0001</ID>
  <Name>Munich Airport Branch</Name>
  <Loc>
     <ZIP>80000</ZIP>
     <City>Munich</City>
  </Loc>
</Destination>
```

The `{{JsXml <JavaScriptExpression>}}` tag is used as follows:

`<flight:DestCity>{{JsXml uda.myDestination.Destination.Loc.City}}</flight: DestCity>`

The JavaScript expression evaluates to:

`<flight: DestCity>Munich</flight: DestCity>`

For details on how to access elements of UDAs of type CUSTOM, refer the *Interstage Business Process Manager Developer's Guide*.

12.8 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.

Generic Java Actions allow you to execute Java methods that are outside the scope of Interstage BPM. They enable you to customize process execution using methods of Java classes that you have implemented yourself.

To assign a Generic Java Action:

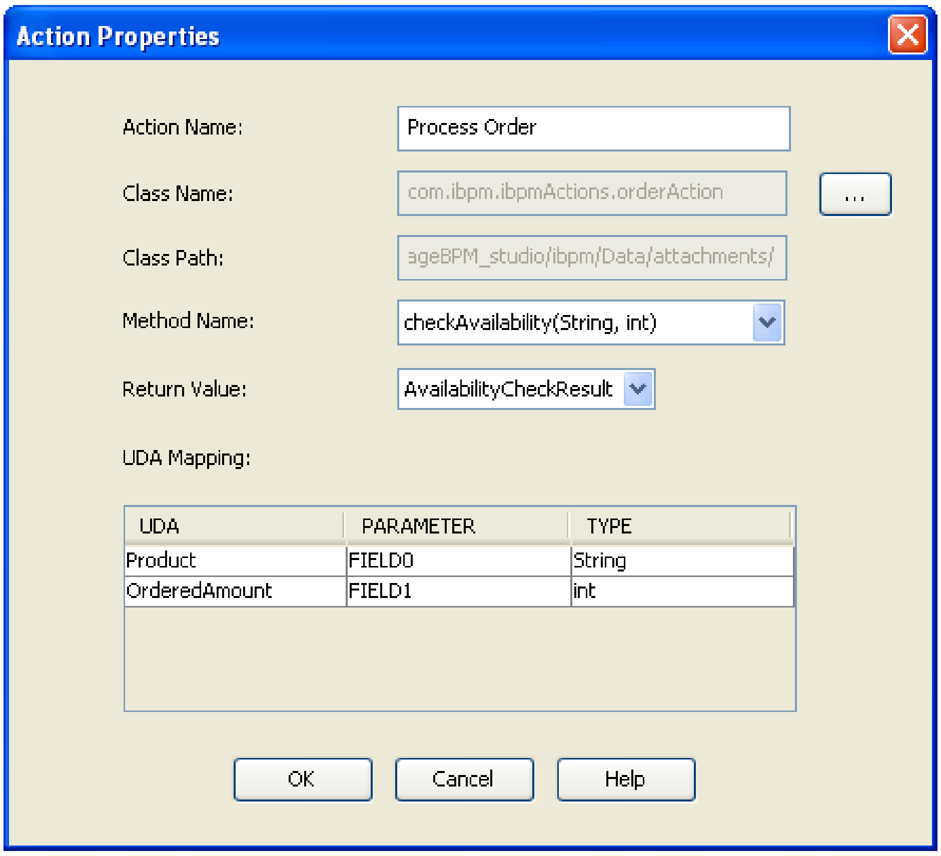
1. Copy the Java class that you want to use. You must differentiate whether you are working with a Workflow Application project or a Server project:
   - **Workflow Application project**:

     In the Workflow Application project's folder structure, store the Java class file(s) to the `Application Classes > engine > classes` folder in the Navigator view.
   - **Server project**:

     Copy the Java class to the `<Interstage BPM 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 Editor** 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.



Action Name: Process Order

Class Name: com.ibpm.ibpmActions.orderAction

Class Path: ageBPM_studio/ibpm/Data/attachments/

Method Name: checkAvailability(String, int)

Return Value: AvailabilityCheckResult

UDA Mapping:

UDA	PARAMETER	TYPE
Product	FIELD0	String
OrderedAmount	FIELD1	int

Figure 208: Assigning a Generic Java Action

10. Click **OK**.

Note: When you send the process definition to an Interstage BPM Server, or deploy your Workflow Application project, the generic Java Actions are automatically sent to the Interstage BPM Server.

12.9 Defining No-Operation Java Actions

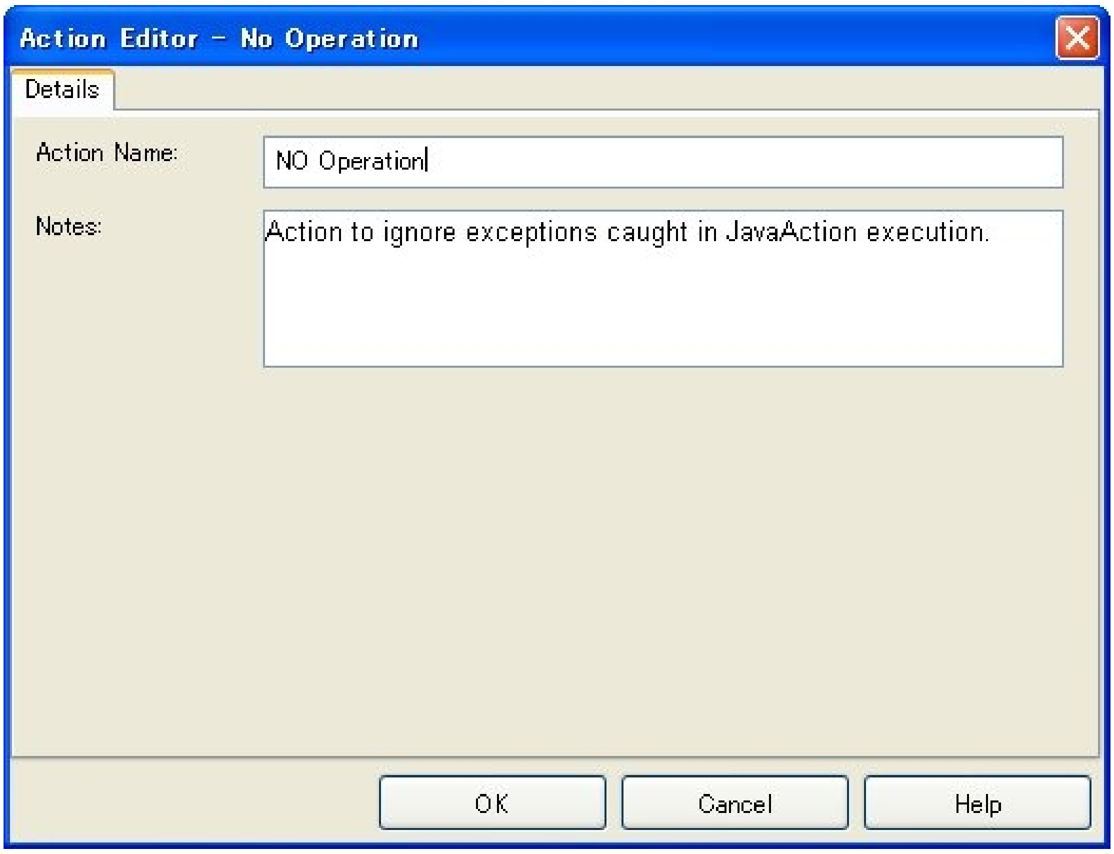
No-Operation Java Actions are built-in Java Actions that specify no operation. These Java Actions allow you to catch a Java Action exception without executing any additional actions. Using No-Operation Java Actions, Interstage BPM simply moves on to the next sequential instruction.

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.



Action Name: NO Operation

Notes: Action to ignore exceptions caught in JavaAction execution.

Figure 209: Defining a No-Operation Java Action

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.

12.10 Defining Exception Handling

With Interstage BPM 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, when the starting of a Remote Subprocess fails, or 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 *Using Error Actions on Process Definition Level* on page 348 for details.
  - On **Node level** (for Remote Subprocess Nodes only): An Error Java Action Set on this level becomes active if the remote subprocess fails to start. Refer to section *Using Error Actions on Node Level* on page 349 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 *Dealing With Errors in Java Actions* on page 271.
- **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 *Using OnSuspend, OnResume, and OnAbort Actions* on page 352 for details.

12.10.1 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.

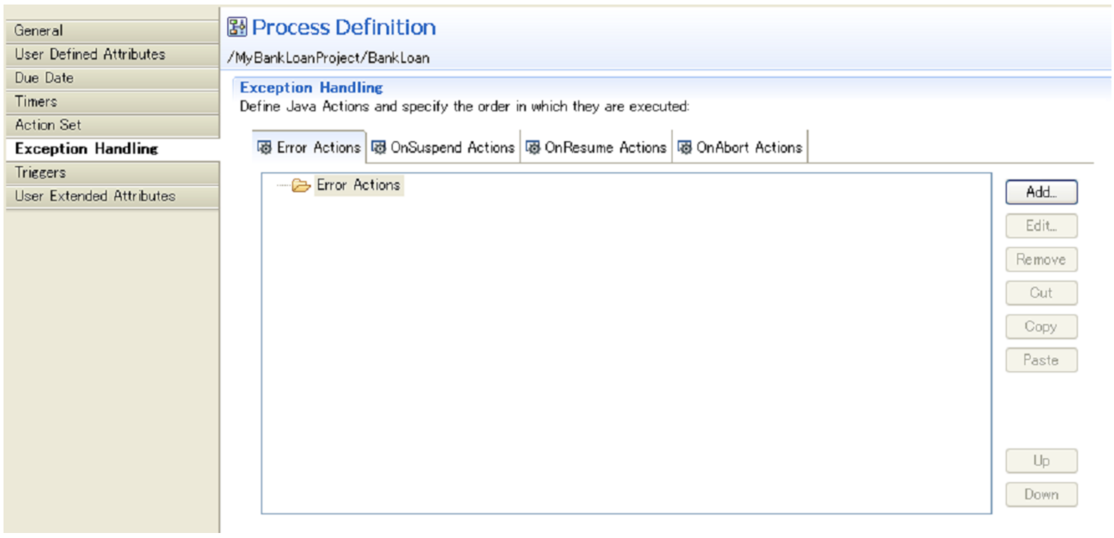


Figure 210: Displaying the Exception Handling Tab

Note: This dialog also allows you to create OnSuspend, OnResume, and OnAbort Actions. These actions are not used to define a specific error handling behavior. They refer to situations when the state of a process instance is changed by an Administrator. Refer to section *Using OnSuspend, OnResume, and OnAbort Actions* on page 352 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 *Using Java Actions* on page 265 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 *Dealing With Errors in Java Actions* on page 271 for more details.

12.10.2 Using Error Actions on Node Level

Prerequisite: Since this feature is available for Remote Subprocess Nodes only, you must have added a Remote Subprocess Node to one of your process definitions.

To define Error Actions on Remote Subprocess Nodes:

1. Select the Remote Subprocess Node to display the Properties view for it.
2. Select the **Exception Handling** tab.

Here you can add new Error Actions to your Remote Subprocess Node. If you have not yet defined any Error Action, the dialog displays an empty Error Actions folder.

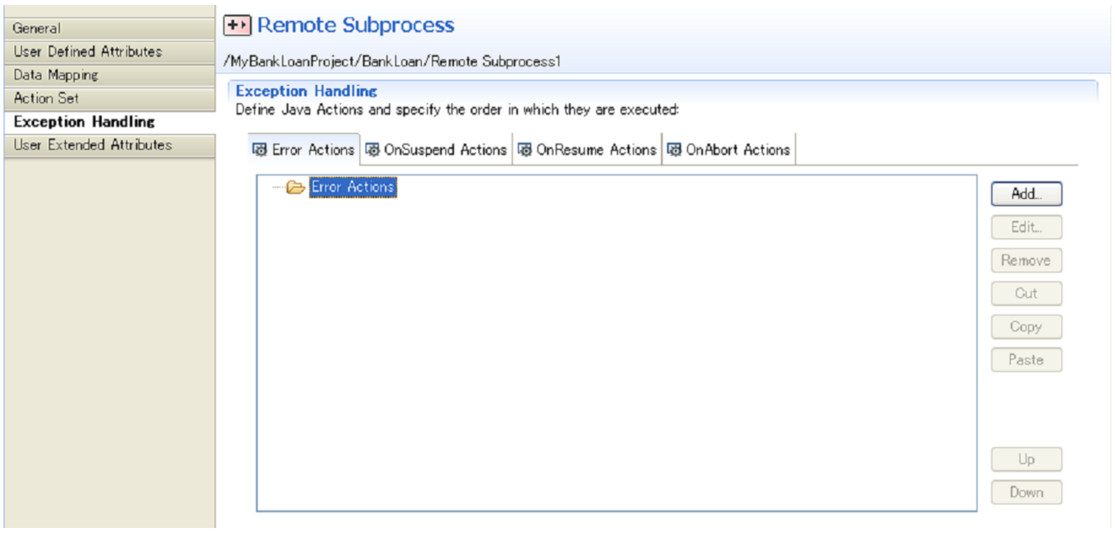


Figure 211: Displaying the Exception Handling Tab

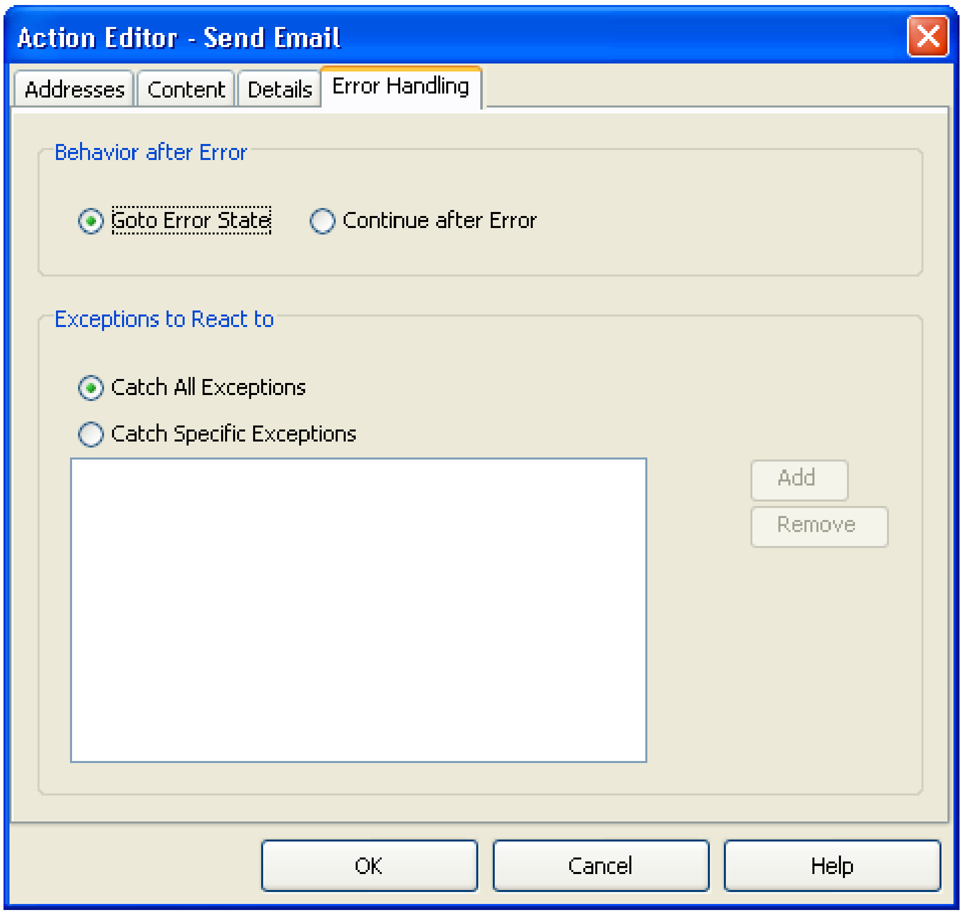
Note: This dialog also allows you to create OnSuspend, OnResume, and OnAbort Actions. These actions are not used to define a specific error handling behavior. They refer to situations when the state of a process instance is changed by an Administrator. Refer to section *Using OnSuspend, OnResume, and OnAbort Actions* on page 352 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 *Using Java Actions* on page 265 for details.

5. In the Action Editor for the selected Java Action, click the **Error Handling** tab. Here you can specify the following error handling settings:



Behavior after Error
[✓ Goto Error State] Continue after Error

Exceptions to React to
[✓ Catch All Exceptions]
Catch Specific Exceptions

Figure 212: Displaying the Error Handling Tab

- **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).

6. In the **Action Editor** dialog, click **OK** to confirm all Error Action parameters.

   The new Error Action will be executed if the Remote Subprocess Node fails to start. You can assign the Error Action to a regular Java Action specified for a Remote Subprocess Node. Refer to section *Dealing With Errors in Java Actions* on page 271 for more details.

12.10.3 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 Interstage BPM 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:

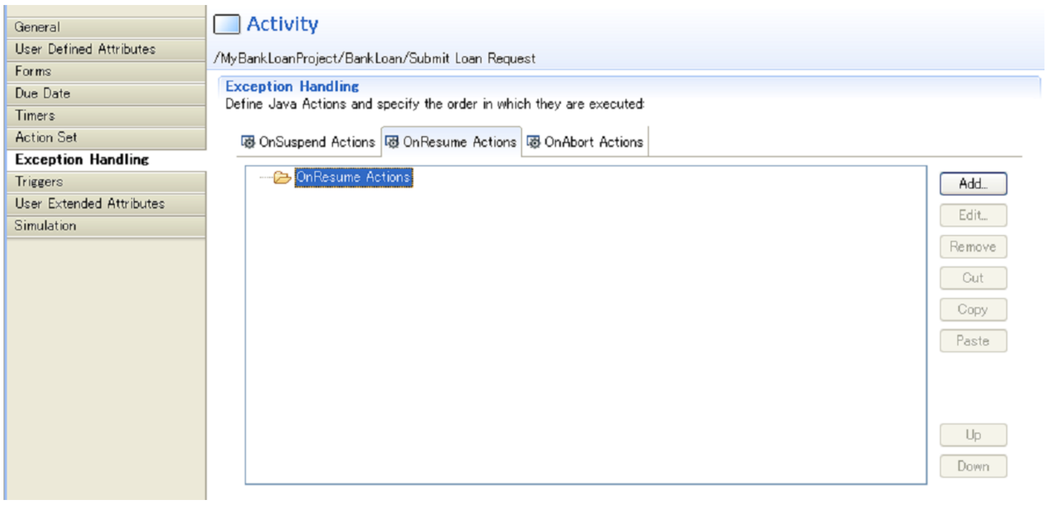


Figure 213: Displaying On* Actions

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 *Using Java Actions* on page 265 for details.
5. Click **OK**.

12.11 Using Triggers

In Interstage BPM, 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 Interstage BPM 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
- as a separate Trigger Node in the process definition for defining an event activity.

The following sections explain how to prepare for triggers and how to define them.

12.11.1 Preparing for Using Triggers

Prerequisite: You know how to define XML schemas or how to write XPath expressions.

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 Interstage BPM 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) Make sure 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 Interstage BPM Server, make sure that the File Listener is configured to check for incoming files. Refer to the *Interstage Business Process Manager Developer's Guide* for information on how to configure the File Listener.

12.11.2 Defining Start Process Triggers

Prerequisites: Refer to section *Preparing for Using Triggers* on page 354.

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.

Note: As an alternative, you can drag and drop the Trigger Node from the Palette onto the Start Node of your process definition. In this case, you need to enter a name for the trigger only and can define it, as described below, at a later point in time via the process definition properties.

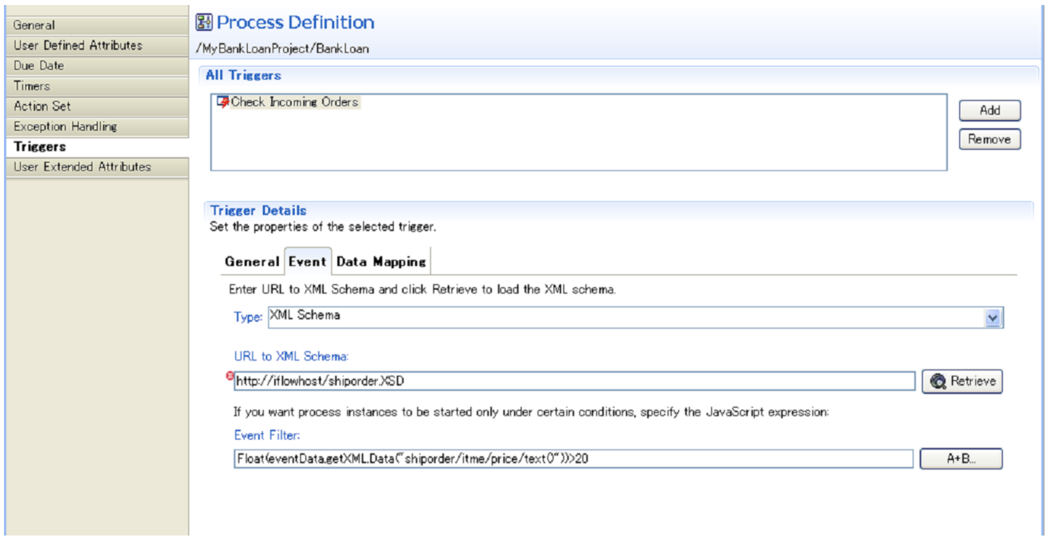
7. Optional: Describe the purpose of your trigger on the **General** sub-tab.
8. Select the **Event** sub-tab.
9. If you have an XML schema that describes the format of the incoming data, type its URL in the **URL to XML Schema** field. Click **Retrieve** to load the XML schema.

   The icon next to the **URL to XML Schema** field tells you if the XML schema has been loaded.
10. If you want 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 *Defining JavaScript Expressions* on page 371.

    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
```



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://iflowhost/shiporder.XSD

If you want process instances to be started only under certain conditions, specify the JavaScript expression:

Event Filter: Float(eventData.getXMLData("shiporder/itme/price/text()"))>20

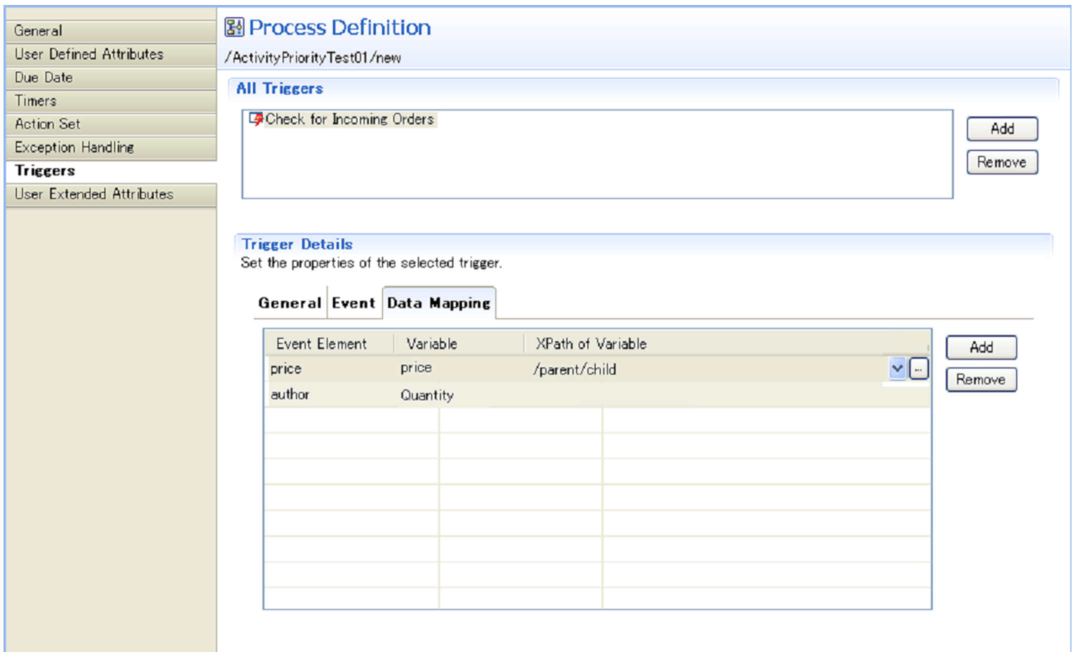
Figure 214: Specifying an Event Filter

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. Otherwise, specify the location of the XML element using an XPath expression.
    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) 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 or CUSTOM from the **Variable** list.



Event Element	Variable	XPath of Variable
price	price	/parent/child
author	Quantity	

Figure 215: Defining Data Mappings

e) If you want to remove a mapping, select it and click **Remove**.

12. When you have completed the data mapping, select the **General** sub-tab. Select the **Enable** check box to enable your trigger.

12.11.3 Defining Make Choice Triggers

Prerequisites: Refer to section *Preparing for Using Triggers* on page 354.

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 *Defining Start Process Triggers* on page 355. For information on JavaScript expressions, refer to section *Defining JavaScript Expressions* on page 371.

11. Select the **Data Mapping** sub-tab. Map the elements of the incoming XML file to UDAs. To map an element:

    a) Click **Add**.

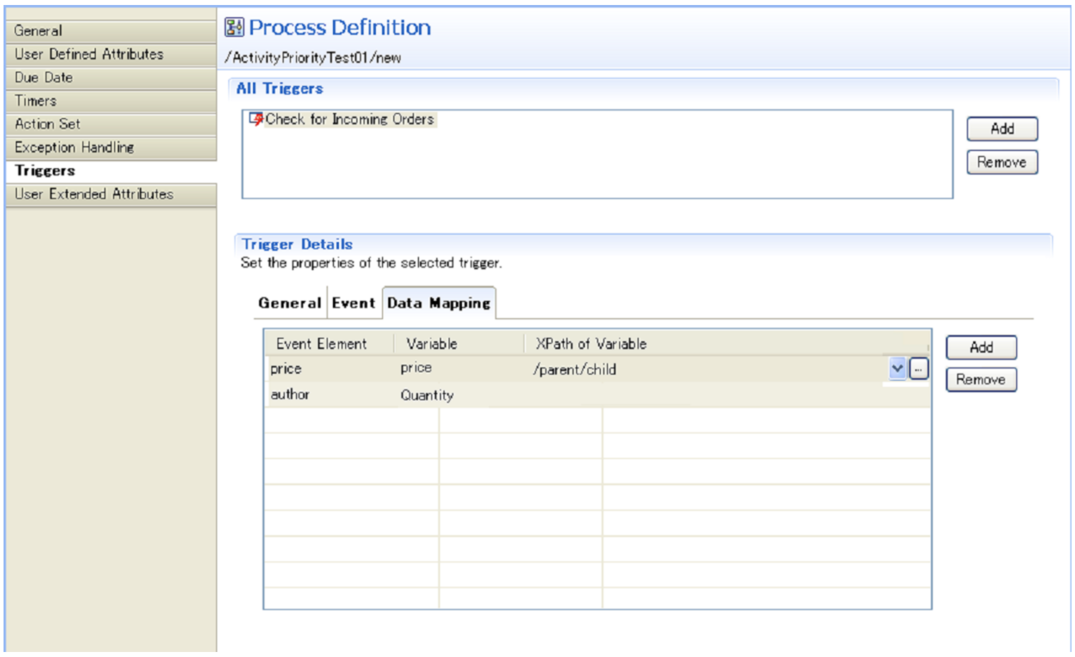
    b) If you specified an XML schema, select an XML element from the **Event Element** list. Otherwise, specify the location of the XML element using an XPath expression.

    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. A drop-down list is displayed. Select from the drop-down list, the XPath expression for the UDA selected in the **Variable** column. 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.



Event Element	Variable	XPath of Variable
price	price	/parent/child
author	Quantity	

Figure 216: Defining Data Mappings

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 or CUSTOM 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 XML 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.

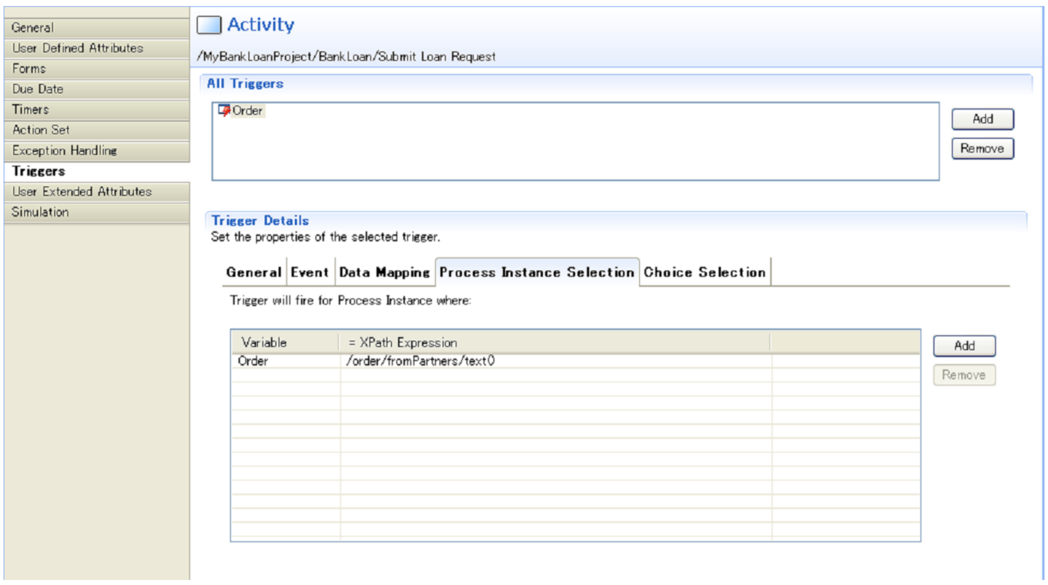
When the trigger is fired, only those process instances are selected in which the value of the UDA matches the value of the corresponding element in the incoming XML file. If you specify multiple conditions, the trigger will only fire when all conditions are satisfied.

Note: Make sure 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:



/MyBankLoanProject/BankLoan/Submit Loan Request

Set the properties of the selected trigger.

Trigger will fire for Process Instance where:

Variable	= XPath Expression
Order	/order/fromPartners/text()

Figure 217: Filtering on process instances

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:

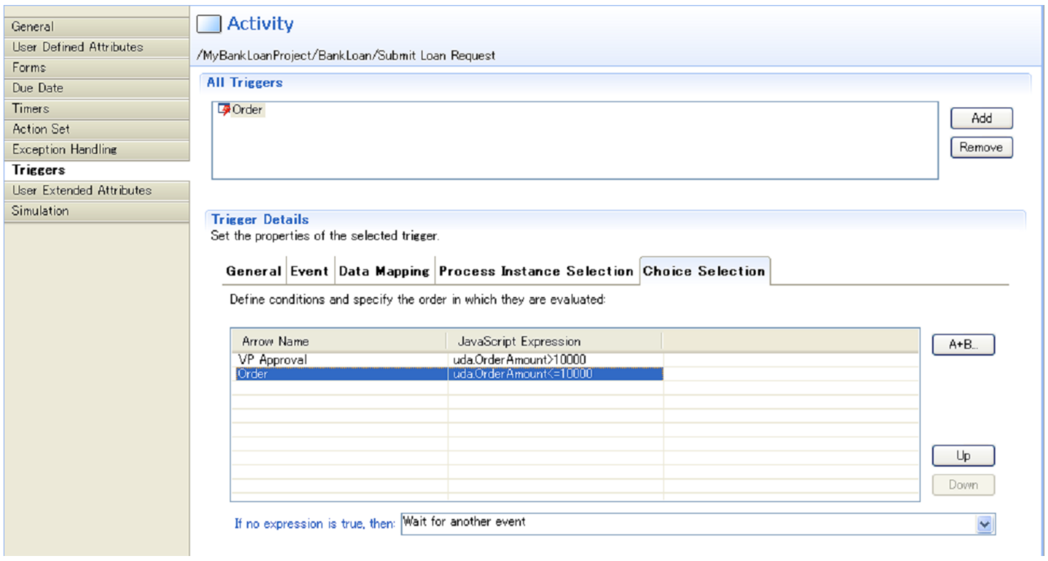
```
<order>
  ...
  <fromPartners>false</fromPartners>
  ...
</order>
```

13. If you want a particular choice to be made only under certain conditions:
    a) Select the **Choice Selection** sub-tab.
    b) From the **Choice** 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:



Activity

/MyBankLoanProject/BankLoan/Submit Loan Request

All Triggers

Order

Trigger Details

Set the properties of the selected trigger.

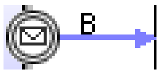
Define conditions and specify the order in which they are evaluated:

Arrow Name	JavaScript Expression
VP Approval	uda.OrderAmount>10000
Order	uda.OrderAmount<=10000

If no expression is true, then: Wait for another event

Figure 218: Defining Conditions for Arrows

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: 

(Add Trigger Intermediate Event).

12.11.4 Using Trigger Nodes

Prerequisites: Refer to section *Preparing for Using Triggers* on page 354.

A Trigger Node represents a step in a process which is driven by an external event. Each Trigger Node has a trigger defined that is supposed to fire when date, typically XML files, comes in from an external system. Once the data arrives, the trigger moves the data into User Defined Attributes (UDAs) according to your trigger definition. The trigger then makes a choice and process execution moves forward to the next node.

This node type does not involve any human interaction. Hence, no work items are generated when a Trigger Node becomes active. It can only be completed by the trigger defined on the node.

A process definition can have any number of Trigger Nodes. A Trigger Node can have one or more incoming arrows and exactly one outgoing arrow.

For every Trigger Node, you need to define one or more Make Choice triggers. Prologue Action, Epilogue Actions and timers can be used as required.

Note that a Trigger Node cannot be assigned to users, Roles, or agents; and it does not support Role Actions or forms.

To define a Trigger Node:

1. Add the User Defined Attributes (UDAs) you will need for your trigger to the process definition.

2. From the Palette, drag and drop a Trigger Node in the Process Definition editor.
3. Select the Trigger Node to display the Properties view for it and select the **Triggers** tab.
4. Click **Add**.

   A new trigger with a default name is added.
5. Proceed as described in section *Defining Start Process Triggers* on page 355.

12.12 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 or Web services. Using Java Agents, you can incorporate these external services into your Interstage BPM process instances.

Interstage BPM Studio allows you to add several Java Agents to your Workflow Application projects. For configuring new Java Agents, Interstage BPM Studio provides an XML file called `agentsConfig.xml`. Once a new Java Agent is configured, it can run as an activity in an Interstage BPM 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 Interstage BPM 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 `agentsConfig.xml` file is automatically stored in the Resources folder of your Workflow Application project. The default `agentsConfig.xml` file automatically opens in the Resource editor. The file looks as follws:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Action agents configuration file -->
<!-- Begin sample agent configuration

For Windows:
<ActionAgentList>
   <ActionAgent>
     <Name>@CRChk</Name>
     <Description>Invoke the CR check inform</Description>
     <RetryInterval<1>/RetryInterval>
     <EscalationInterval>1</EscalationInterval>
     <ClassName>com.fujitsu.iflow.tests.AgentSimulator</ClassName>
     <ClassPath>C:\Fujitsu\InterstageBPM\classes</ClassPath>

<ConfigFile>C:\Fujitsu\InterstageBPM\server\instance\default\attachments\AUTOEXEC.BAT</ConfigFile>

   </ActionAgent>
</ActionAgentList>

For Solaris:
<ActionAgentList>
   <ActionAgent>
     <Name>@CRChk</Name>
     <Description>Invoke the CR check inform</Description>
     <RetryInterval<1>/RetryInterval>
     <EscalationInterval>1</EscalationInterval>
     <ClassName>com.fujitsu.iflow.tests.AgentSimulator</ClassName>
     <ClassPath>/opt/FJSVibpm/classes</ClassPath>
```

```
<ConfigFile>/opt/FJSVibpm/server/instance/default/attachments/AUTOEXEC.BAT</ConfigFile>

   </ActionAgent>
</ActionAgentList>
```

2. Navigate to the `agentsConfig.xml` file and type in the properties that are required for the new Java Agent.

   Set the following properties:

   - **Name**: Name of the agent. Make sure that the Agent name always starts with `@`. This designates it as a Java Agent.
   - **Description**: Explains form and function of the Agent.
   - **RetryInterval**: Specifies the amount of time, in seconds, between retry attempts when a failure occurs.
   - **EscalationInterval**: Specifies the number of failures after which the System Administrator is notified via email.
   - **ClassName**: Name of the Java class associated with the Agent.
   - **ClassPath**: Classpath of the Java class associated with the Agent. If this entry is not specified, Interstage BPM Studio will use the `C:\Fujitsu\InterstageBPM\classes` Classpath.
   - **ConfigFile**: Fully qualified name of the configuration file used by the Java Agent.

   The sample configuration file `BankLoanAgentsConfig.xml` is located in the `<Interstage BPM Studio installation folder>\samples\BankLoan` directory. This file contains the configuration parameters for a Java Agent and a list of possible Java Agent implementations.

3. Type in the properties for the new Java Agent, and 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.

12.13 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 Interstage BPM 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 Interstage BPM Server machine can connect.

Refer *Working With Resources* on page 94 for more information about the location of FTP Agents in Interstage BPM Studio.

Refer the topic *Configuring FTP Agents* in *Interstage Business Process Manager Developer's Guide* for more information about FTP Agents.

To define a new FTP Agent:

1. Right-click your project in the **Navigator** view and select **New >> FTP Agent**.

New FTP Agent dialog is displayed. The following figure displays the **New FTP Agent** dialog.

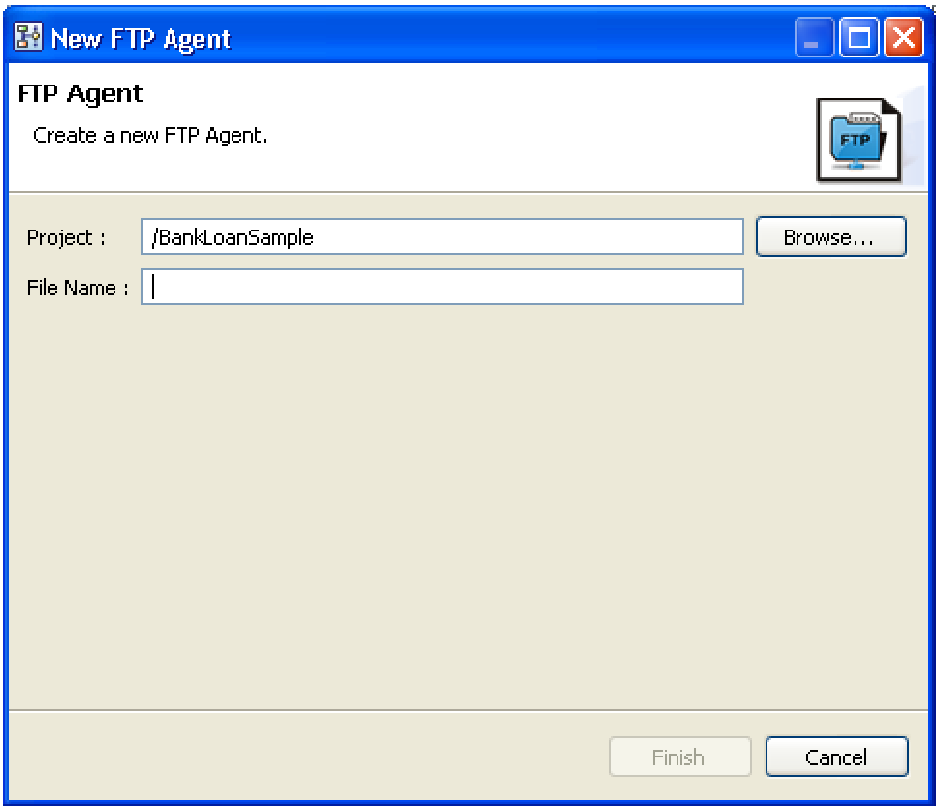


Figure 219: Defining FTP Agent

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. The FTP configuration file is displayed as given below.

```
<Services>
   <Service>
     <ServiceType>FTP</ServiceType>
     <ServiceStatusUDA>AgentServiceStatus</ServiceStatusUDA>
     <ServiceResultUDA>AgentServiceResult</ServiceResultUDA>
     <ServiceSpecificInfo>
        <FTPHost>127.0.0.1</FTPHost>
        <FTPPort></FTPPort>
        <FTPUser>anonymous</FTPUser>
        <FTPPassword></FTPPassword>
        <FTPType>BINARY</FTPType>
        <FTPAppend>FALSE</FTPAppend>
        <FTPDirectory>%%REMOTE_FTP_DIRECTORY%%</FTPDirectory>
        <FTPFileNames>%%REMOTE_FTP_FILES%%</FTPFileNames>
        <Documents>%%OUTGOING_FILES%%</Documents>
      </ServiceSpecificInfo>
   </Service>
 </Services>
```

6. Configure the `FTPAgent.xml` file using its tags. The following table describe the tags used in `FTPAgent.xml` file.

Tag	Description
`<ServiceType>`	For the FTP Agent, the service type is always FTP. If there is no service type specified or the service type is invalid, the process instance goes into error state. This tag is used by all agents using the ServiceAgent class.
`<ServiceStatusUDA>`	Name of the User Defined Attribute (UDA) that stores the status of the FTP Agent activity. Possible values are Done or Failed. Removing this tag has no effect on the FTP Agent; however, the user cannot see the status of the FTP Agent activity.
`<ServiceResultUDA>`	Name of the UDA that stores the error message, if the FTP Agent throws an exception. Removing this tag has no effect on the FTP Agent; however, if the FTP Agent fails, the user cannot see the error that occurred.
`<FTPHost>`	Host name or IP address of the machine that receives the transferred file. The default value is 127.0.0.1, i.e. the local host. If there is no FTP host specified or the FTP host is invalid, files are transferred to the local FTP server, if available. This tag is specific to the FTP Agent.
`<FTPPort>`	Port used by the FTP server. If no port is specified, the default FTP port 21 is used. This tag is specific to the FTP Agent.
`<FTPUser>`	User name of the FTP user that transfers the file. If the user name is invalid, file transfer fails and the process instance goes into error state. This tag is specific to the FTP Agent.
`<FTPPassword>`	Password of the FTP user that transfers the file. This tag is specific to the FTP Agent.

Tag	Description
`<FTPType>`	Type of files to be transferred. Allowed values are `ASCII` or `BINARY`. Default value is BINARY. Use `ASCII` if you are transferring text files. Use `BINARY` if you are transferring binary files. If any other value than `ASCII` or `BINARY` is specified, the FTP Agent throws an exception. This tag is specific to the FTP Agent.
`<FTPAppend>`	Specifies whether a file to be transferred is appended to a remote file. Allowed values are `TRUE` or `FALSE`. Default value is `FALSE`. Use `TRUE` if you want to append the contents of the local file to an already existing remote file. Use `FALSE` if you want to overwrite an already existing remote file. If any other value than `TRUE` or `FALSE` is specified, the FTP Agent throws an exception. This tag is specific to the FTP Agent.
`<FTPDirectory>`	Name of the User Defined Attribute (UDA) that specifies the directory to which files are transferred. A path relative to the FTP server is specified. For example: if `/iflow` is the value of the `REMOTE_FTP_DIRECTORY UDA` and you are working on a Windows system, the file is transferred to `c:\Inetpub\ftproot\iflow`. If an invalid directory is specified, the process instance goes into error state. This tag is specific to the FTP Agent.
`<FTPFileNames>`	Name of the UDA that specifies the file names to be used for the remote files. This parameter is used to rename files while transferring them. For example, if `LocalFile.txt` is attached to the process instance and `REMOTE_FTP_FILES` has `RemoteFile.txt` as its value, `LocalFile.txt` is transferred to the FTP directory and renamed to `RemoteFile.txt.` `REMOTE_FTP_FILES` can contain several file names separated by semicolon (;). If `REMOTE_FTP_FILES` contains no file names, the original file names are used.

Tag	Description
`<Documents>`	Name of the UDA that specifies the files that are transferred to the FTP directory. `OUTGOING_FILES` can contain several file names separated by semicolon (;). If `OUTGOING_FILES` contains no file names, all files attached to the process instance are transferred.

Note: Files that have a semicolon (;) in their names cannot be transferred to an FTP directory.

Semicolons are considered as file name delimiters and are therefore not allowed within names of files that are to be transferred.

12.14 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.

Refer *Working With Resources* on page 94 to more information about the location of HTTP Agents in Interstage BPM Studio.

Refer the topic *Configuring HTTP Agents* in *Interstage Business Process Manager Developer's Guide* for more information about FTP Agents.

To define a new HTTP Agent:

1. Right-click your project in the **Navigator** view and select **New >> HTTP Agent**.

New HTTP Agent dialog is displayed. The following figure displays the **New HTTP Agent** dialog.

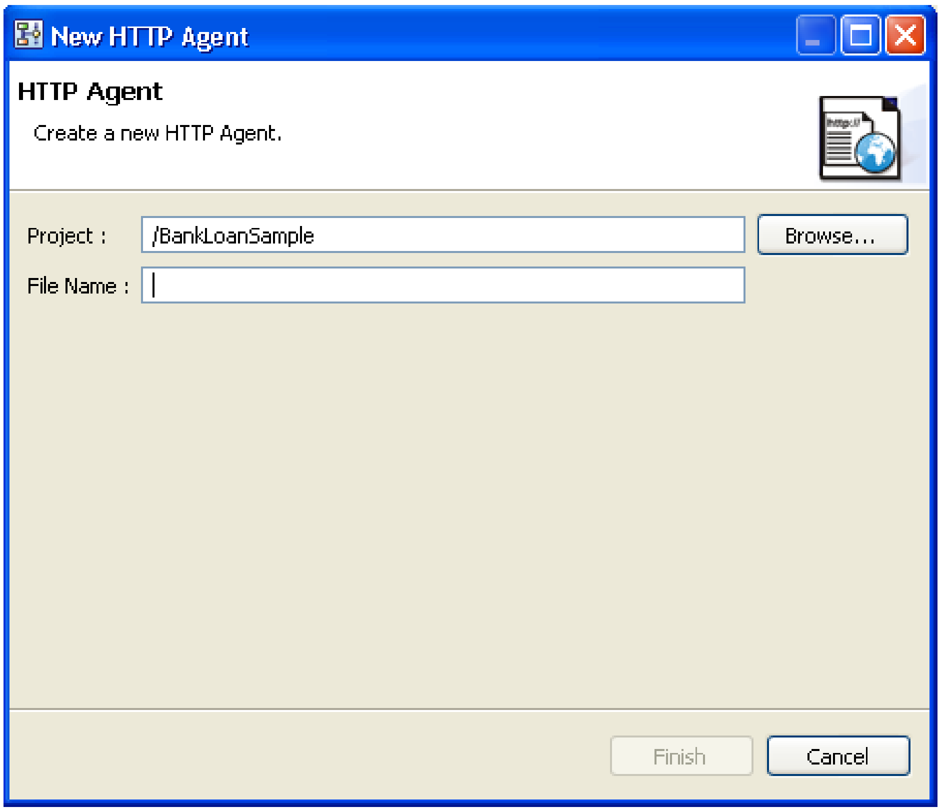


Figure 220: Defining HTTP Agent

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. The HTTP configuration file is displayed as given below.

```
<HTTPAgent>
  <HTTPBaseURL method="POST"
    followRedirects="true">{{Field HTTP_URL}}</HTTPBaseURL>
  <HTTPHeaderParams name="Content-Type">text/xml;
    charset=UTF-8</HTTPHeaderParams>
  <HTTPHeaderParams name="User-Agent">I-BPM 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>
```

6. Configure the `HTTPAgent.xml` file using its tags. The following table describe the tags used in `HTTPAgent.xml` file.

Tag	Description
`<HTTPBaseURL>`	URL end point. The value for the `<HTTPBaseURL>` tag specifies the external service to be called by the HTTP Agent, e.g. a Servlet. This value can also be defined to come from a process instance UDA as QuickForm expression, such as `{{Field HTTP_URL}}`. Attributes: `method="POST"` or `"GET"`. Defines the request method to be used for the HTTP request. If you do not specify the `method` attribute, the process instance will go into the error state. If you specify an attribute value other than `"POST"` or `"GET"`, the HTTP agent will fail. `followRedirects="true"`. Since the default value for `followRedirects` is `false`, you must specify this attribute.
`<HTTPHeaderParams>`	Sets the HTTP header properties like the encoding style, the content type, etc. HTTP headers are used to define a variety of web object properties, for example, the properties of request and response objects for the current HTTP session. Attributes: `name="Content-type" or "User-Agent"`. `"Content-type"` defines the type of requested content and takes the following value: `value="text/xml", charset=UTF-8`. This value can also be defined to come from process instance UDAs as QuickForm expression such as `{{Field contentType}}`. `"User-Agent"` contains information about the client (user agent) from where the request originates. Note that currently there is no restriction for the value that can be specified in the `name` attribute of the `<HTTPHeaderParams>` tag. Any valid values for this attribute are valid HTTP headers.

Tag	Description

`<QueryParams>`	This tag is required in case the `"GET"` method is used with the HTTP Agent. The value of this tag is passed to the external service as Query String. Note that the value of the UDA specified in the `<HTTPRequestUDA>` tag is not used when the `"GET"` method is used.
	Correct usage of this tag is as follows:
	`<QueryParams name= "ParamName">Param value </QueryParams>`
	This will result in the following URL specification during HTTP Agent execution:
	`URL?ParamName=Param Value`
	You can specify any valid string for the name attribute of the `<QueryParams>` tag.

`<HTTPRequestUDA>`	Value of the UDA specified in this tag is part of the query string for the HTTP request.
`<HTTPAgentStatusUDA>`	UDA where the HTTP service status will be stored.
`<HTTPResponseUDA>`	UDA where the HTTP response will be stored.

12.15 Defining File Listener

File Listeners monitor files in specified directory locations. When they detect new or newly modified files, they notify the Interstage BPM file handlers, so they can perform automatic functions like starting Interstage BPM process instances or making choices on activities. File listeners are typically used for integrating Interstage BPM with other enterprise applications.

Refer *Working With Resources* on page 94 for more information about the location of File Listener file in Interstage BPM Studio.

Refer the topic *File Listeners* in *Interstage Business Process Manager Developer's Guide* for more information about File Listeners.

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 each project.

Note: In non-SaaS mode, you must not create the File Listener file. Refer to the *Interstage Business Process Manager Developer's Guide* for information on how to configure the File Listener. It is necessary to change the following files of the server to use File Listener.

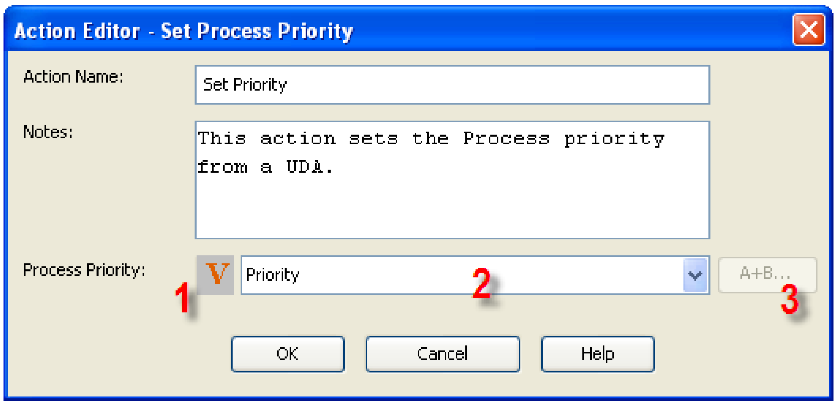
`<Interstage BPM Server Installation Directory>/server/instance/default/tenants/Default/attachments/apps/System/fileListenerConf.xml`

12.16 Defining JavaScript Expressions

In Interstage BPM 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:



Action Editor - Set Process Priority

Action Name: Set Priority

Notes: This action sets the Process priority from a UDA.

Process Priority: V Priority A+B...

Figure 221: Defining JavaScript Expressions

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.

Mode	Button	Description
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.

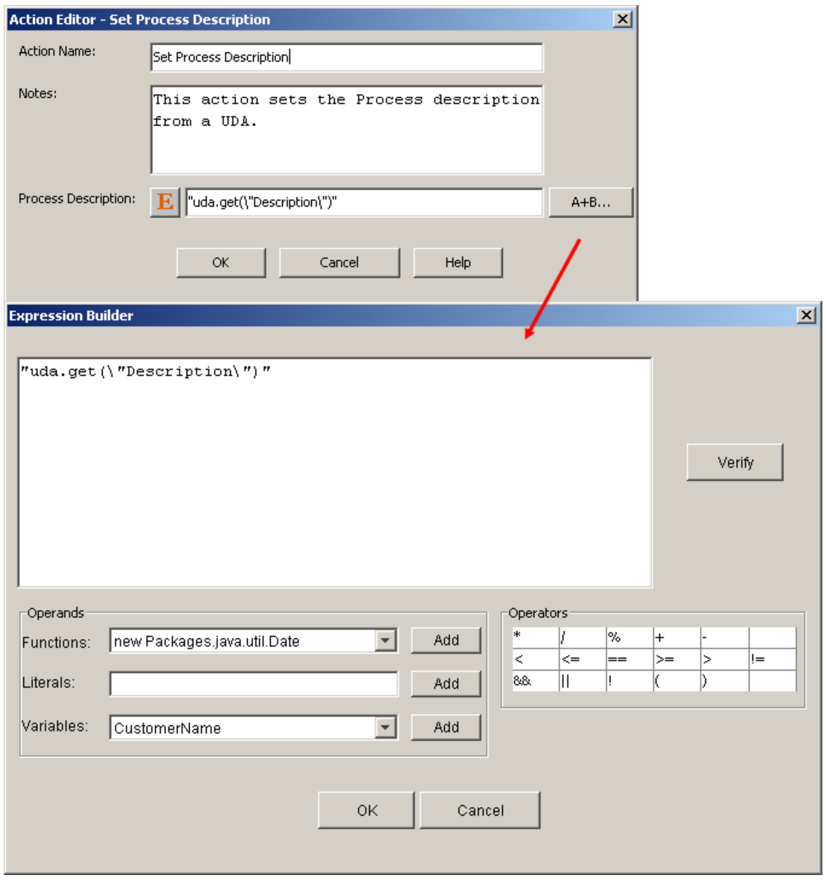
Expression Field

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.



Action Editor - Set Process Description

Action Name: Set Process Description

Notes: This action sets the Process description from a UDA.

Process Description: "uda.get(\"Description\")"

Expression Builder

```
"uda.get(\"Description\")"
```

Operands

Functions: new Packages.java.util.Date

Literals:

Variables: CustomerName

Operators

*	/	%	+	-	
<	<=	==	>=	>	!=
&&	||	!	(	)	

Figure 222: Expression Builder

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.

Note: Application Variable is not displayed in **Variables** drop-down list. When Application Variable is used, it defines it as follows: for example, if Variable1 is an Application Variable then its expression will be `sec.getApplicationVariable("Variable1")`.

- **Operators** area

In this area, you can click an operator to add it to your JavaScript expression. The standard JavaScript operators are available.

12.16.1 Example: Building a JavaScript Expression Using the Expression Builder

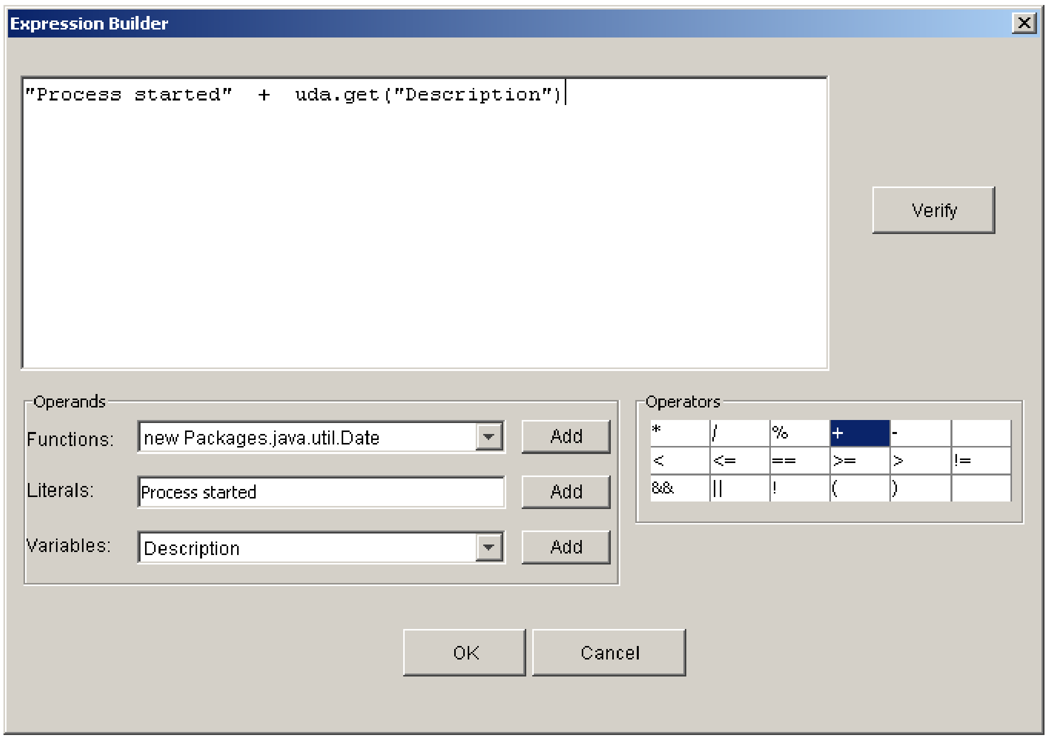
This example uses the Expression Builder to build a JavaScript expression that defines a process instance description. The process instance description formed by this example tells the purpose of the process instance.

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 *Specifying User Defined Attributes* on page 151.
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:



```
"Process started"  +  uda.get("Description")
```

Functions: new Packages.java.util.Date

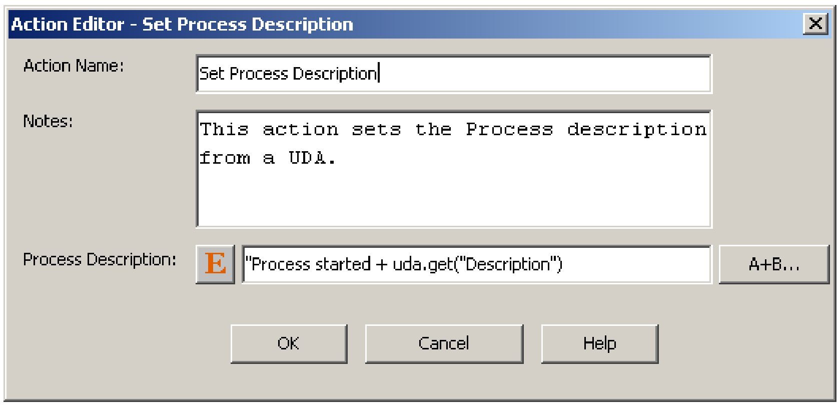
Literals: Process started

Variables: Description

Figure 223: Building a Sample JavaScript Expression

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 .



Action Name: Set Process Description

Notes: This action sets the Process description from a UDA.

Process Description: "Process started + uda.get("Description")

Figure 224: Expression Field with Sample

13 Simulating Processes

This chapter explains how to simulate the execution of processes.

13.1 Introduction to Process Simulation

The Interstage BPM 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 process definition whose execution is to be simulated must not contain one or several Remote Subprocess Nodes. If it does, an error will be thrown and process simulation will fail.
- 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.

13.1.1 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:

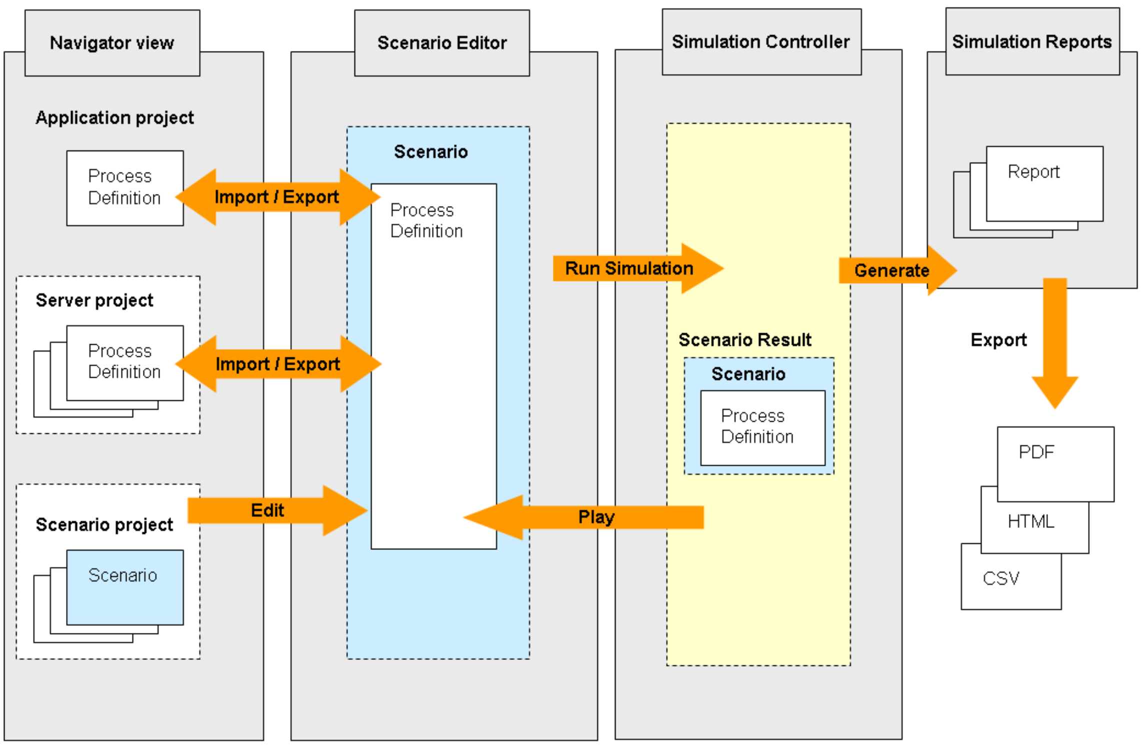


Figure 225: Simulation Process Overview

13.1.2 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 *Defining Simulation Properties for Nodes* on page 388 for details.

The processing of all nodes except Remote Subprocess Nodes can be simulated. For example, in case of a Delay Node its timer is simulated.

2. You create a new simulation scenario. Refer to section *Creating a Simulation Scenario* on page 378 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 *Defining Simulation Properties for a Scenario* on page 384 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 *Preparing a Simulation Result* on page 390 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 *Using the Simulation Controller* on page 392 for details.
6. The simulation results can then be evaluated using the extensive simulation report functionality. Refer to section *Generating Simulation Reports* on page 394 for details.

13.2 Defining a Simulation Scenario

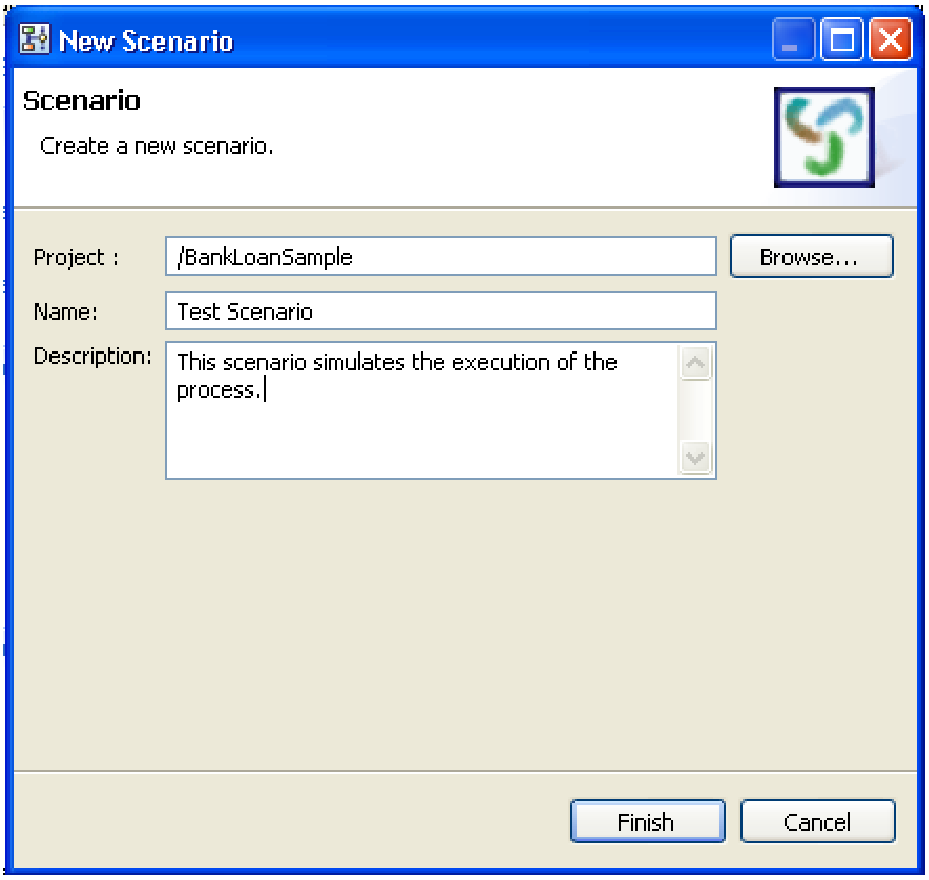
This section describes how to create a scenario and define simulation properties.

13.2.1 Creating a Simulation 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:



Project : /BankLoanSample
Name: Test Scenario
Description: This scenario simulates the execution of the process.

Figure 226: Creating a Scenario

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 *Defining Simulation Properties for a Scenario* on page 384 for details.

13.2.2 Using Simulation Values From History

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. **Server Connection Setting** 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* on page 81 to know more about various parameters.

Note: In SaaS mode, the `<tenant>` in the URL connects you to the specific tenant on the Interstage BPM Server. Refer *Interstage BPM Studio Features* on page 17 to know more about SaaS mode.

Note: If you want to connect to Interstage BPM Server in non-SaaS mode, enter **default** in `<tenant>` in the URL. This will, through the default tenant, directly connect you to the server specified in `<hostname>` in the URL.

4. Click **OK** on **Select Server Connection** dialog.

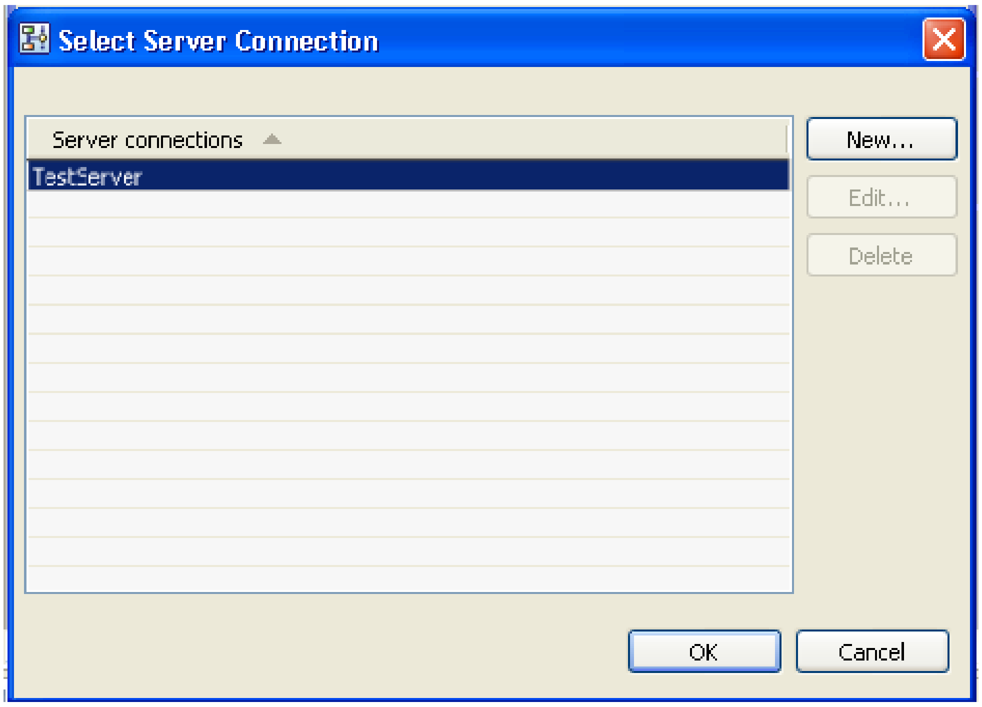
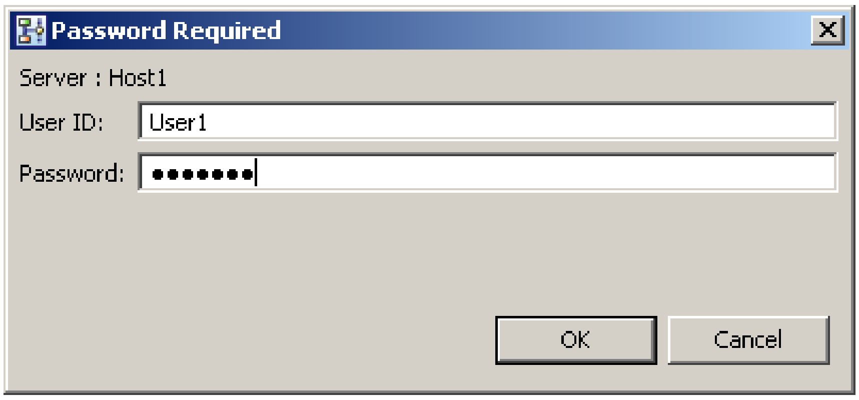


Figure 227: Selecting a server connection

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.


Password Required

Server : Host1

User ID: User1

Password: •••••••

Figure 228: Entering your 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**.

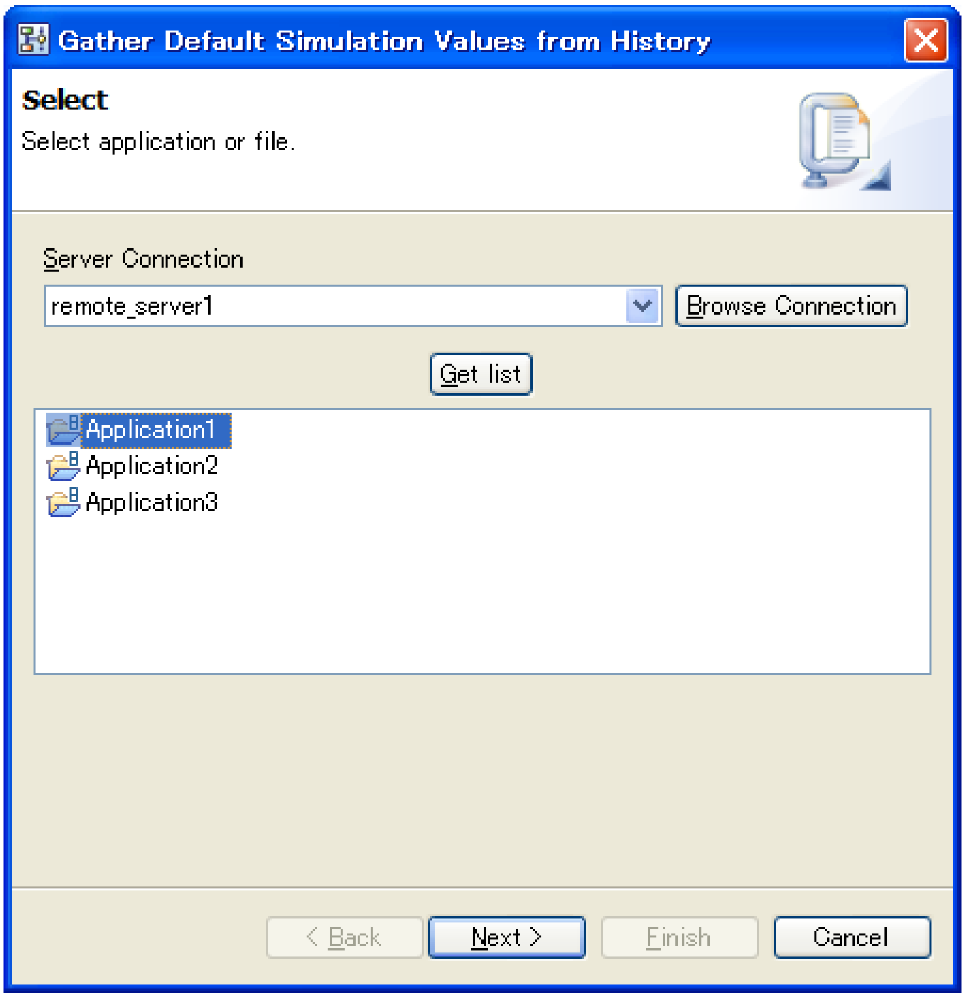


Figure 229: Displaying applications on the selected server

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**.

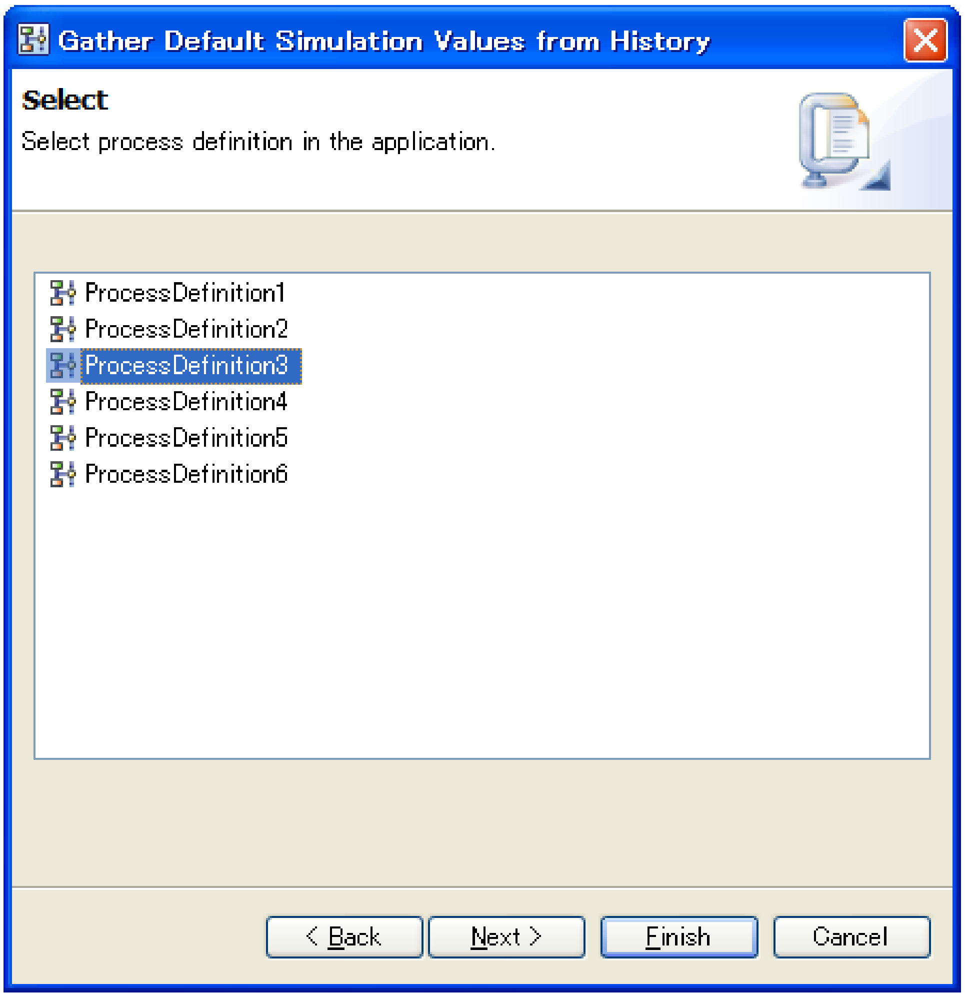


Figure 230: Displaying process definitions

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.

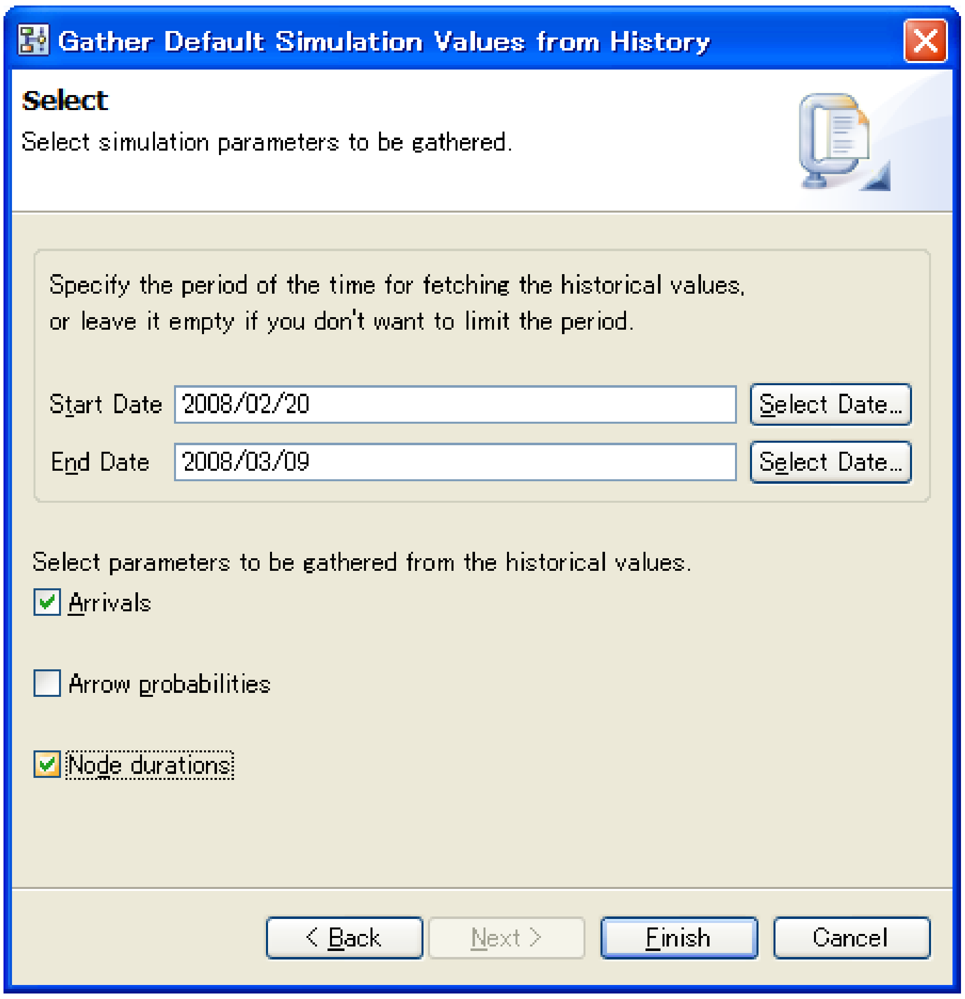
Note: You can update the list of process definitions using the pop-up menu or pressing the F5 key on your keyboard.

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 the Interstage BPM Studio starts getting historical simulation values.
   - **End Date**: Refers to the date until which the Interstage BPM 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.


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 2008/02/20

End Date 2008/03/09

Select parameters to be gathered from the historical values.

[✓] Arrivals

[ ] Arrow probabilities

[✓] Node durations

Figure 231: Specifying simulation dates

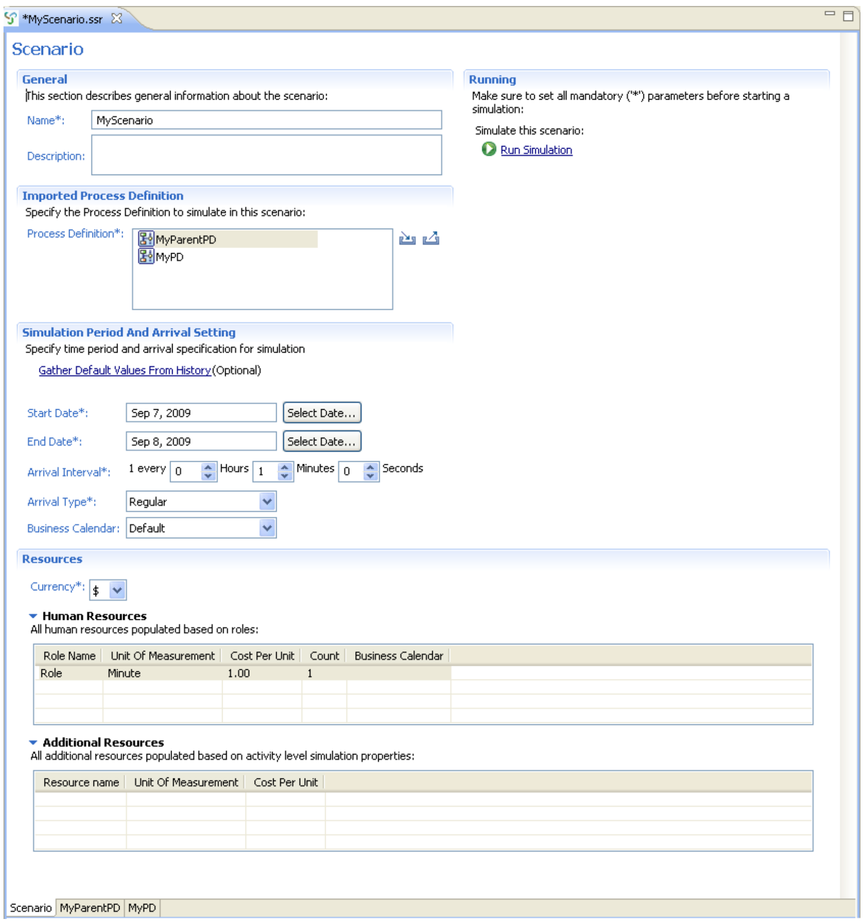
Note: If you have selected a process definition having Subprocess Nodes or Chained-Process Nodes for simulating its execution, the **Arrivals** checkbox will be disabled.

Note: If you do not select any dates, the time period for fetching historical values will be unlimited.

13.2.3 Defining Simulation Properties for a Scenario

Prerequisite: You have created a scenario in the **Simulation Scenarios** project or a Workflow Application project. Refer to *Creating a Simulation Scenario* on page 378 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.


*MyScenario.ssr

Scenario

General
This section describes general information about the scenario:

Name*:	MyScenario
Description:	

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

Simulate this scenario:
Run Simulation

Imported Process Definition
Specify the Process Definition to simulate in this scenario:

Process Definition*: MyParentPD, MyPD

Simulation Period And Arrival Setting
Specify time period and arrival specification for simulation

Gather Default Values From History (Optional)

Start Date*:	Sep 7, 2009
End Date*:	Sep 8, 2009
Arrival Interval*:	1 every 0 Hours 1 Minutes 0 Seconds
Arrival Type*:	Regular
Business Calendar:	Default

Resources

Currency*: $

Human Resources
All human resources populated based on roles:

Role Name	Unit Of Measurement	Cost Per Unit	Count	Business Calendar
Role	Minute	1.00	1	

Additional Resources
All additional resources populated based on activity level simulation properties:

Resource name	Unit Of Measurement	Cost Per Unit

Scenario | MyParentPD | MyPD

Figure 232: The Scenario Editor

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. **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.

Note: 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.

Note: A process definition with subprocess or chained-process definitions cannot be simulated in the **Simulation Scenarios** project.

Note: 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 displayed in the **Process Definition** field. These subprocess or chained-process definitions are retrieved from the application project to which the scenario belongs. In other words, even if you import the primary process definition from another application project, Studio tries to find subprocess or chained-process definitions from the current application 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 *Defining Simulation Properties for Nodes* on page 388.

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 *Using Simulation Values From History* on page 380.

- **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 the Studio. The default calendar defines business hours as 8:30 AM to 6:00 PM. The default calendar is stored at the following location:

  ```
  <Interstage BPM Studio 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.

  Refer to the *Interstage Business Process Manager Server Administration Guide* for more details and examples.

Note: When simulating a process definition, Interstage BPM Studio automatically uses the Calendar that has been specified for the Workflow Application project the process definition belongs to.

  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.

13.2.4 Defining Simulation Properties for Nodes

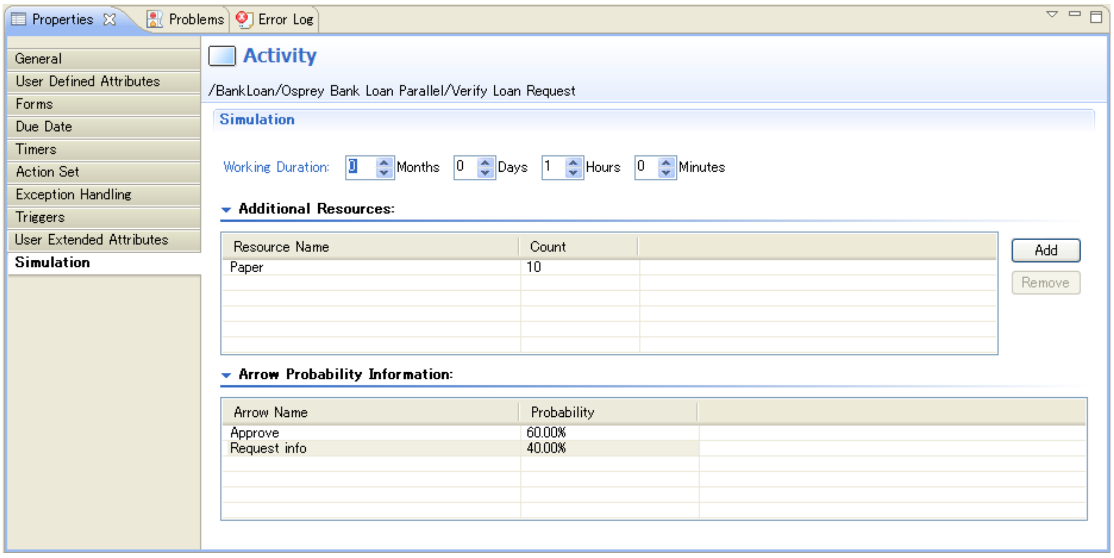
Prerequisite: You have created a project with a process definition whose execution you want to simulate. It contains nodes of one or more of the following types:

- 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:



Properties | Problems | Error Log

- General
- User Defined Attributes
- Forms
- Due Date
- Timers
- Action Set
- Exception Handling
- Triggers
- User Extended Attributes
- Simulation

Activity

/BankLoan/Osprey Bank Loan Parallel/Verify Loan Request

Simulation

Working Duration: 0 Months 0 Days 1 Hours 0 Minutes

Additional Resources:

Resource Name	Count
Paper	10

Add Remove

Arrow Probability Information:

Arrow Name	Probability
Approve	60.00%
Request info	40.00%

Figure 233: Simulation Node Properties view

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. Make sure 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%.

If you want to simulate a process that contains two or more nodes with the same name, then the calculation of how long the node execution takes is aggregated across the branches. As an example, a process branches into three possibilities. On each branch, there is an **Invoice Customer** node. The three nodes are different because they belong to different threads. However, the calculation of how long an **Invoice Customer** node takes will be aggregated across the three branches because the nodes have the same name. Consequently, the same average duration is put into each of the three nodes that have the same name.

13.2.5 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.

13.3 Executing a Simulation Scenario

This section explains how to execute a simulation scenario, and describes the simulation results.

13.3.1 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:

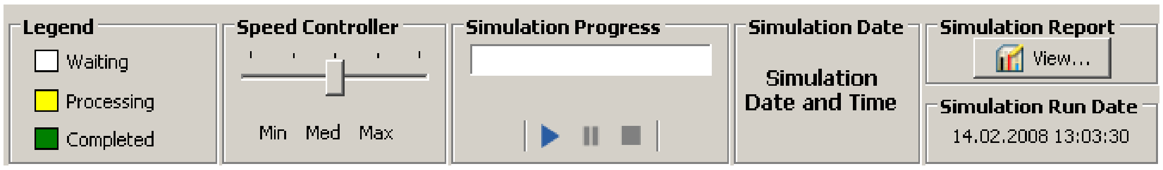
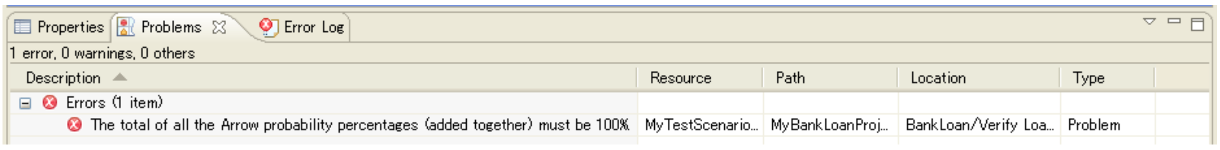


Figure 234: Displaying the Simulation Controller

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 occured, error location and project are displayed in this view. These details are displayed in **Decsription**, **Resource**, **Project**, **Path**, **Location**, and **Type** 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 (/).


Properties | Problems | Error Log

1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
The total of all the Arrow probability percentages (added together) must be 100%	MyTestScenario...	MyBankLoanProj...	BankLoan/Verify Loa...	Problem

Figure 235: Error Displayed in Problems View

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.

Button	Action	Description
■	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 *Using the Simulation Controller* on page 392.

13.3.2 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:

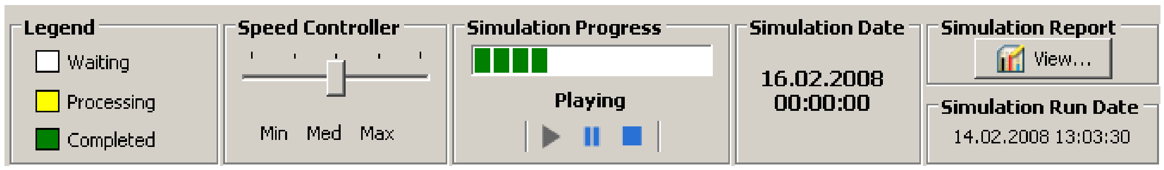


Figure 236: Simulation Controller

To execute a simulation scenario:

1. Prepare a simulation, as described in section *Preparing a Simulation Result* on page 390.
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 `BankLoan.xpdl` process definition has been imported into the `MyTestScenario.ssr` file:

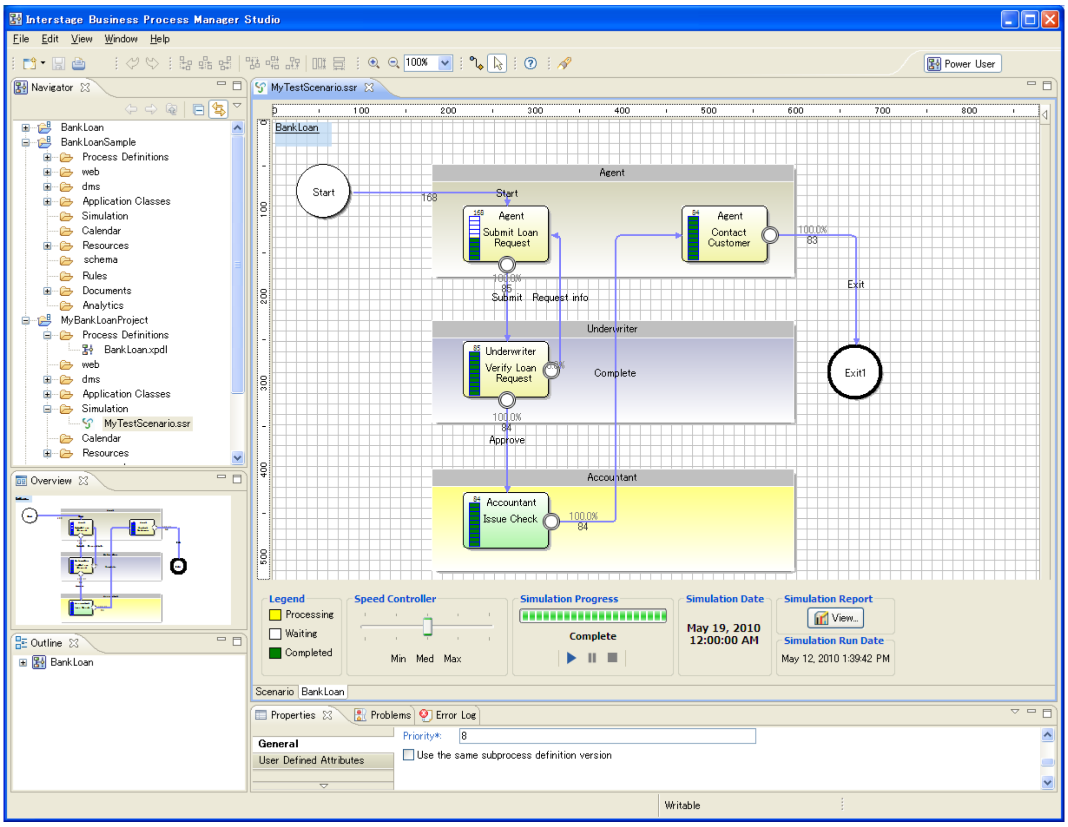


Figure 237: Simulating a process definition

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.

   To **stop** the execution, click the **Stop** button. This will cancel the entire simulation run.

Note: 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 *Generating Simulation Reports* on page 394 for details.

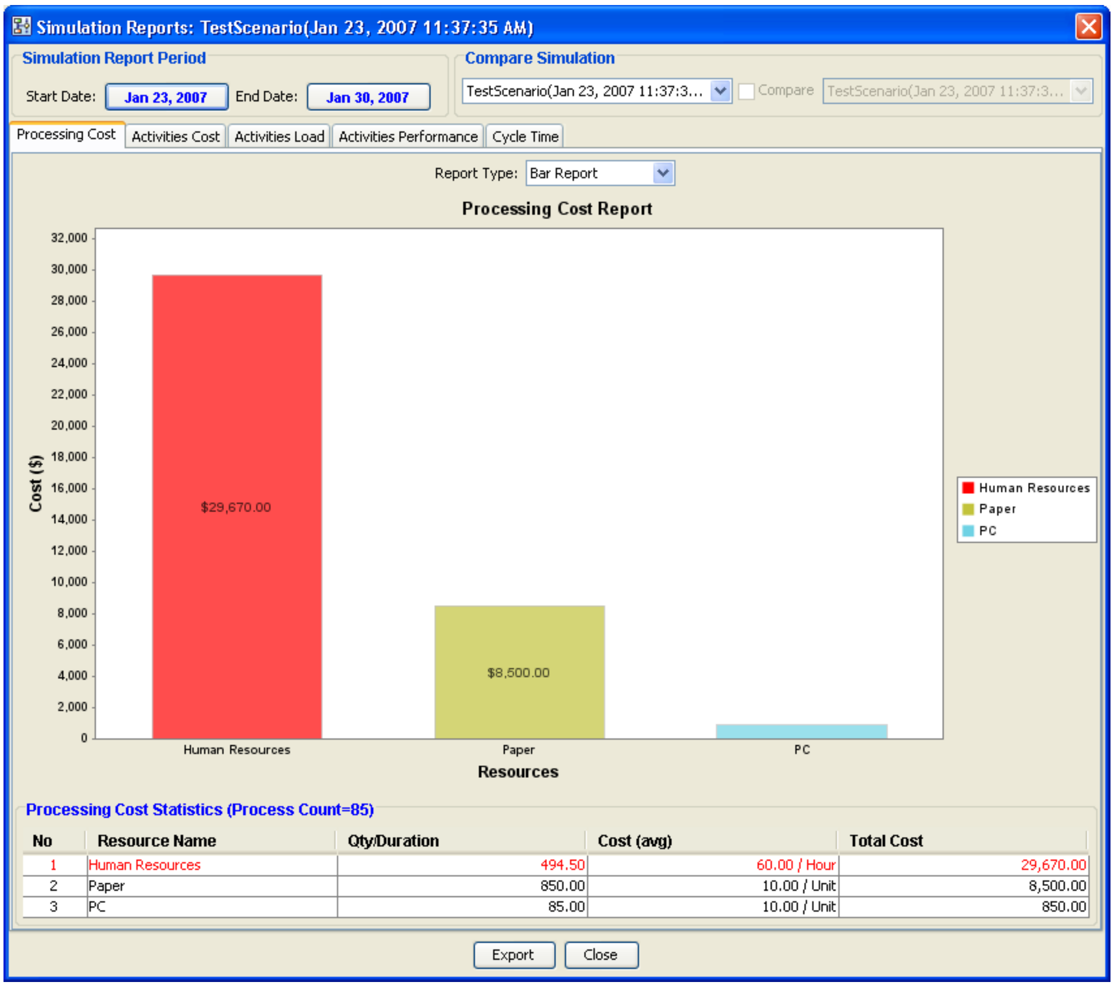
13.3.3 Generating Simulation Reports

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:



Processing Cost Statistics (Process Count=85)

No	Resource Name	Qty/Duration	Cost (avg)	Total Cost
1	Human Resources	494.50	60.00 / Hour	29,670.00
2	Paper	850.00	10.00 / Unit	8,500.00
3	PC	85.00	10.00 / Unit	850.00

Figure 238: Simulation Reports Window

Note: Simulation Report is displayed only in the Similation 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 *Simulation Report Types* on page 396 for details 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.

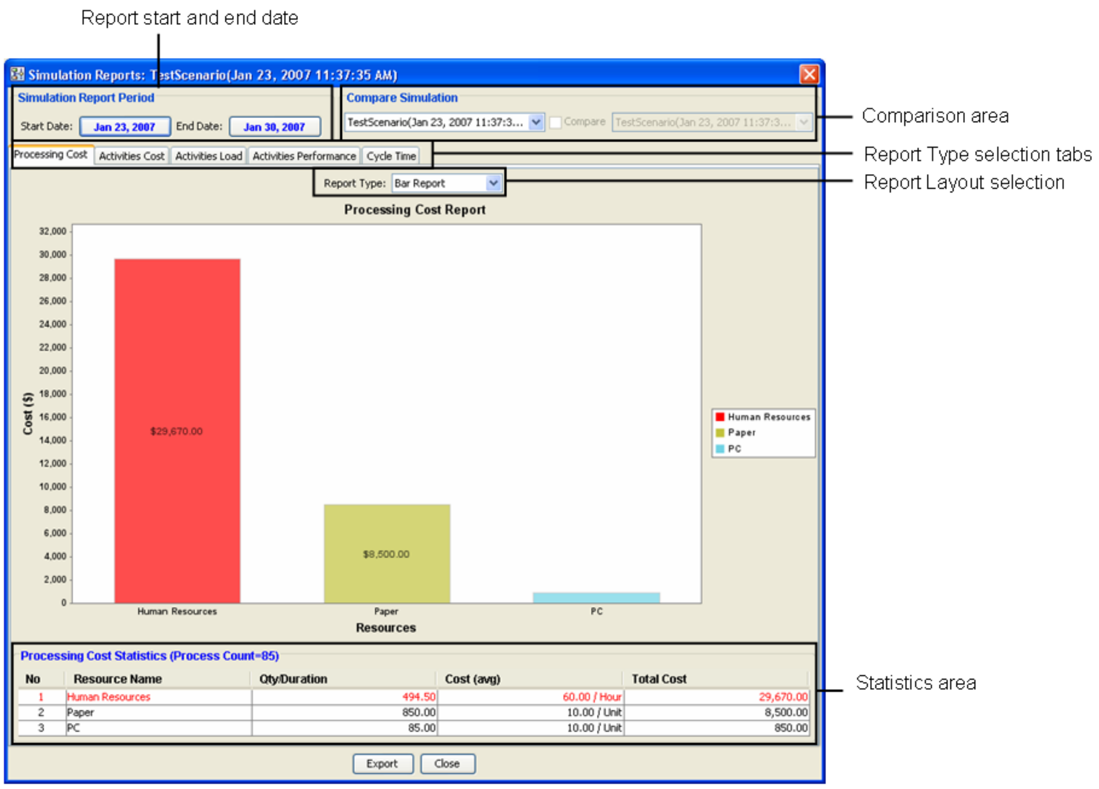
   Refer to section *Comparing Simulations* on page 399 for details.

5. You can export your simulation reports by clicking the **Export** button.

   Refer to section *Exporting Simulation Reports* on page 398 for details.

13.3.4 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:



Processing Cost Statistics (Process Count=85)

No	Resource Name	Qty/Duration	Cost (avg)	Total Cost
1	Human Resources	494.50	60.00 / Hour	29,670.00
2	Paper	850.00	10.00 / Unit	8,500.00
3	PC	85.00	10.00 / Unit	850.00

Figure 239: Simulation Report Parts

The various types of report can be displayed by clicking the respective tab:

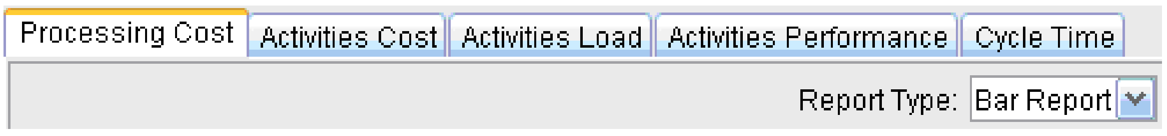


Figure 240: Report Types

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

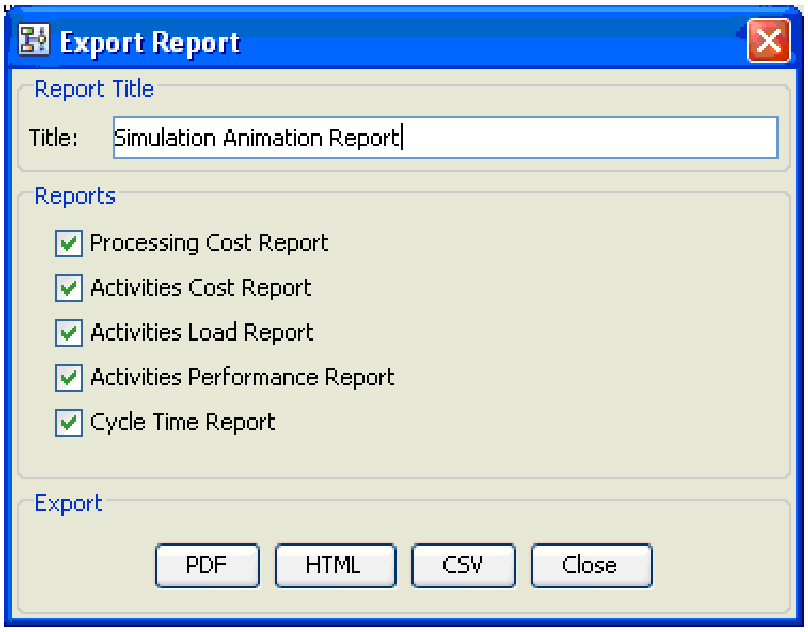
Simulation reports can be generated on two sets of simulation data for comparison purposes. Such a comparison results in the generation of comparative reports. You can consider only one process definition at a time and both of the simulations must have a result corresponding to the same process definition, 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.

13.3.5 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.



Export Report

Report Title

Title: Simulation Animation Report

Reports

- ✓ Processing Cost Report
- ✓ Activities Cost Report
- ✓ Activities Load Report
- ✓ Activities Performance Report
- ✓ Cycle Time Report

Export

Figure 241: Export Report Dialog

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 *Simulation Report Types* on page 396 for details 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.

13.3.6 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.

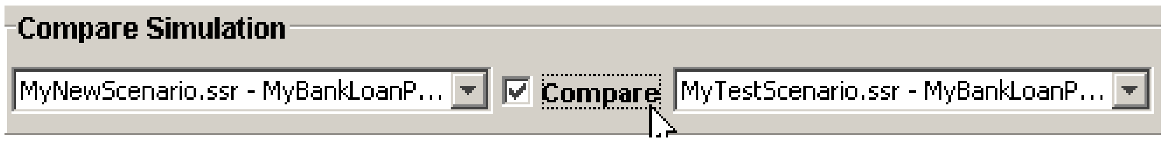

Compare Simulation
MyNewScenario.ssr - MyBankLoanP...
Compare
MyTestScenario.ssr - MyBankLoanP...

Figure 242: Comparing simulation results

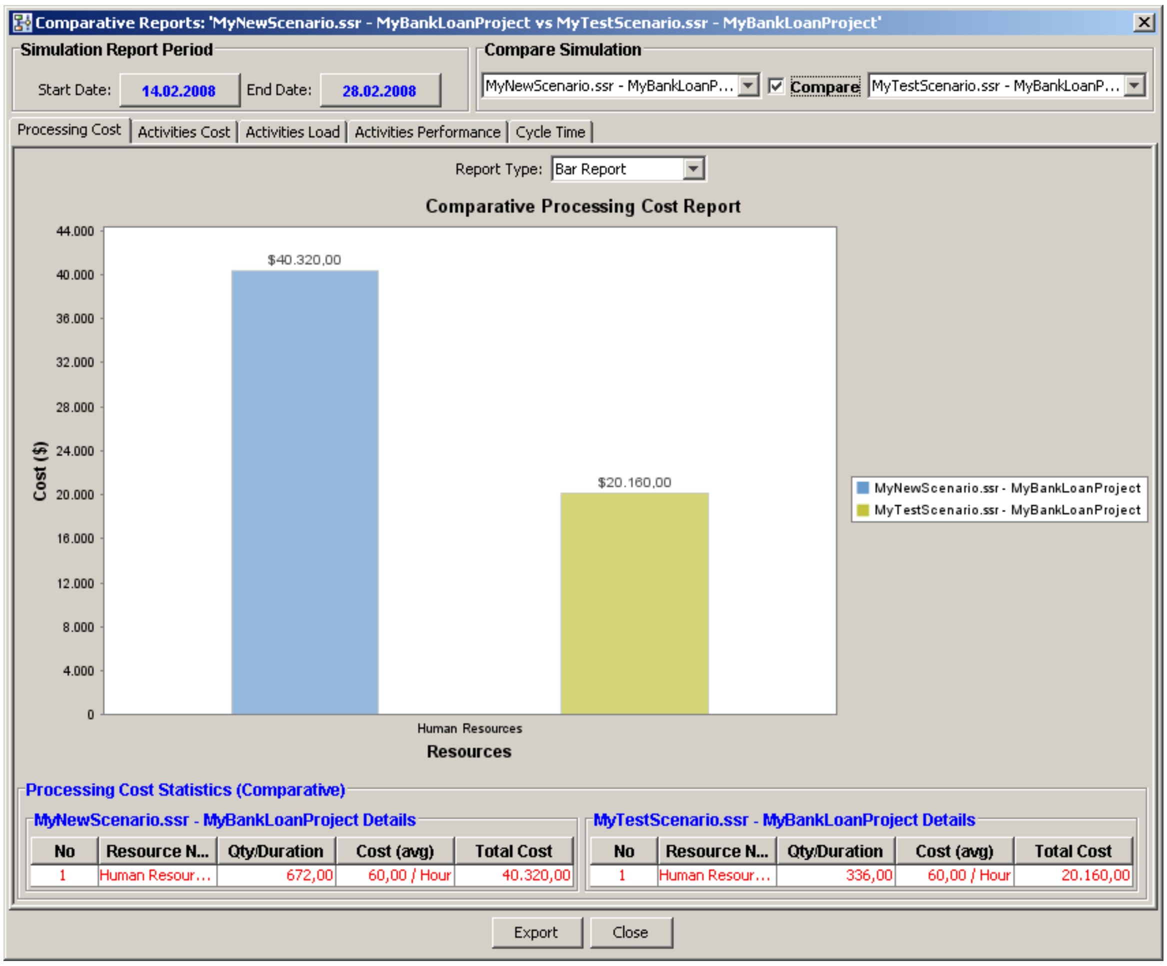
To compare two simulation results:

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 *Simulation Report Types* on page 396 for details. The following example shows a comparative report.



Start Date: 14.02.2008 End Date: 28.02.2008

MyNewScenario.ssr - MyBankLoanP... ☑ Compare MyTestScenario.ssr - MyBankLoanP...

Report Type: Bar Report

Processing Cost Statistics (Comparative)

MyNewScenario.ssr - MyBankLoanProject Details

No	Resource N...	Qty/Duration	Cost (avg)	Total Cost
1	Human Resour...	672,00	60,00 / Hour	40.320,00

MyTestScenario.ssr - MyBankLoanProject Details

No	Resource N...	Qty/Duration	Cost (avg)	Total Cost
1	Human Resour...	336,00	60,00 / Hour	20.160,00

Figure 243: Displaying a comparative report

13.4 Managing Simulation Scenarios

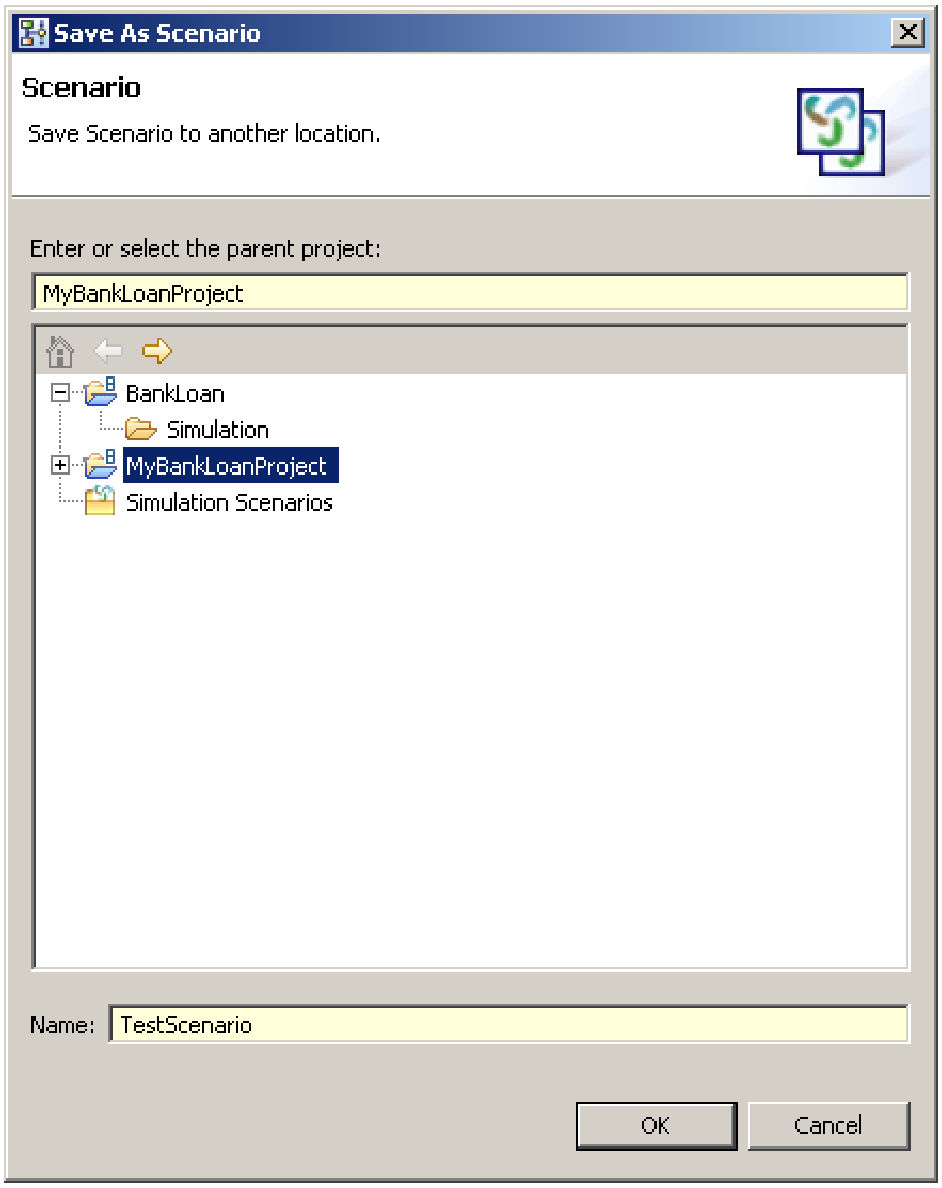
This section explains the functions available for managing simulation scenarios.

13.4.1 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**.



Save As Scenario

Scenario

Save Scenario to another location.

Enter or select the parent project:

MyBankLoanProject

Name: TestScenario

Figure 244: Saving a Scenario to a Different Name

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.

13.4.2 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**.

13.4.3 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.

13.4.4 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**.

13.4.5 Importing a Scenario

You can import scenarios in XML format into Interstage BPM 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.

13.4.6 Exporting a Scenario

You can export scenarios from Interstage BPM 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**.

13.4.7 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.

13.4.8 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.

13.4.9 Removing a Scenario

When you remove a scenario, it is removed from the Interstage BPM 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**.

Appendix A: Supported JavaScript Functions

Interstage BPM provides a set of JavaScript functions that you can use with Java Actions, triggers, Complex Conditional Nodes, and Sequential Loop Nodes.

This appendix explains which functions you can use with these elements. Also, it provides a description of some of these functions.

Apart from functions listed in this appendix, you can use the JavaScript functionality that is defined in the ECMA Standard. For information about the ECMA Standard, refer to the document `ecma-262.pdf` included with the Interstage BPM installation.

Note: The size that a method used in a JavaScript may have is limited by the Java Virtual Machine (JVM). Currently, the method byte code size is limited to 65535 bytes (64 KBytes). If you use a method with a larger size, the JVM will throw an error and you have 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, and Complex Conditional Nodes.

Note: Since numeric values are treated as values of type Number, you can use only the following range of integers in JavaScript:

- Minimum: -9007199254740992
- Maximum : 9007199254740992

Note that the following minimum and maximum values of type Long 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 that comes with the J2SE Development Kit (JDK).

Date DateAdd(Date or Number date, Number offset, String field)

Returns a JavaScript Date object containing the date and time that is the result of adding the offset to the date. `field` determines the unit of time measure 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: The example works correctly only if the User Defined Attribute (UDA) `Date` is of type DATE.

Say `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 type Number, its value is interpreted as 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 type Number, their values are interpreted as milliseconds since January 1, 1970.

Number DateDiff(Date or Number date1, Date or Number date2, String field)

Subtracts `date2` from `date1`. 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 type Number, their values are interpreted as 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 parameters specified. The result inherits the precision from the parameter with the most significant figures.

If you want to assign the result to a UDA, make sure that the UDA is of type BIGDECIMAL.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal.

Example:

```
int x = 39;
var z = DecimalAdd ("3.1416",x);
```

If `z` is a BigDecimal, 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, make sure that the UDA is of type BIGDECIMAL.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal.

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. `scale` determines the number of significant digits for rounding. Any Number can be used as a `scale` value. Default value 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, make sure that the UDA is of type BIGDECIMAL.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal.

BigDecimal DecimalMultiply(BigDecimal value1, BigDecimal value2, Number scale)

Multiplies `value1` by `value2` and returns the result of the multiplication. `scale` determines the number of significant digits for rounding. Any number can be used for `scale`. Default value 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, make sure that the UDA is of type BIGDECIMAL.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal.

BigDecimal DecimalSubtract(BigDecimal value1, BigDecimal value2)

Subtracts `value2` from `value1`. Returns the difference between these values as a BigDecimal.

boolean toBoolean(String or Number value)

Converts `value` to a JavaScript Boolean. The value can be either a String containing `"true"` or `"false"`, or a Number containing zero (for true) or non-zero (for false). Returns `true` or `false` depending on the parameter passed. If `value` cannot be converted into a Boolean, `false` is returned.

BigDecimal toDecimal(BigDecimal value, Number scale)

Converts `value` to a BigDecimal object. Returns the result of the conversion to the number of significant figures specified with `scale`. Any Number can be used for `scale`. Default value is 2.

If you want to assign the result to a UDA, make sure that the UDA is of type BIGDECIMAL.

If you pass any other data type as a parameter, the function converts the parameter to a BigDecimal.

Packages.java.lang.Float.parseFloat

For a description, refer to the Javadoc that comes with the J2SE Development Kit (JDK).

Packages.java.lang.Integer.parseInt

For a description, refer to the Javadoc that comes with the J2SE Development Kit (JDK).

Packages.java.lang.String.valueOf

For a description, refer to the Javadoc that comes with the J2SE Development Kit (JDK).

A.2 JavaScript Functions Supported with Java Actions

With Java Actions, you can use the functions explained in section *General JavaScript Functions* on page 404 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)
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()
String sec.getProcessTitle()
String sec.getProcessXMLAttributeElementValue(String udaName, String xPath)
String sec.joinString(Array)
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.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. Use the following syntax:

`uda.<udaIdentifier>`. Note that you must use the identifier and not the name of the UDA, as 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 value of the UDA to the specified value:

  ```
  uda.set("<udaName>", "<udaValue>");
  ```

When assigning a JavaScript return value to a UDA, make sure that their data type matches. Otherwise, the assignment fails.

Note: If the assignment of a value to a target UDA fails due to conversion or any other kind of errors, error details are logged in `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 type BIGDECIMAL are mapped to `Packages.java.math.BigDecimal` objects.
- UDAs of type DATE are mapped to `Packages.java.util.Date` objects.

For details about the Java objects, refer to the Javadoc that comes with the J2SE Development Kit (JDK).

A.3 JavaScript Functions Supported with Triggers

With triggers, you use JavaScript expressions to specify control conditions. Control conditions narrow down when the trigger fires.

You can use the functions explained in section *General JavaScript Functions* on page 404 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

Interstage BPM supports a set of files that you need for running a specific process solution:

This appendix provides a complete list of files supported by Interstage BPM, 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, and design (`.css`) files, and QuickForm (`.jsp` and `.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

Folder Name	Description of Files	Location in the Application Project Directory
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 • Configuration file for Java Agents used by the application (`agentsConfig.xml`) • Configuration files for new data sources used by Interstage BPM Studio (`DataSourceDefinition.xml`) • File Listener file (`fileListenerConf.xml`) • Process Scheduler File (`ProcessScheduler.xml`)	<Project Root>/Resources
schema	The **schema** folder contains XML schema defined as CUSTOM UDA.	<Project Root>/schema
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 empty. After installing the Interstage Analytics plugin, this folder contains the process definitions (`.xpdl` files) defining the Analytics setting.	<Project Root>/Analytics

Glossary

ACID properties	A transaction is a set of actions that obeys the four so-called ACID properties: atomic, consistent, isolated, and durable.
Activity	The description of a task, logical step, or work to be performed in a process. An activity is represented by a work item.
Activity Node	A graphical representation of an activity .
Activity Time	The time it takes to perform a particular activity.
Agents	Components that asynchronously access systems external to Interstage BPM.
AND Node	A node that synchronizes multiple branches in a process.
Annotation	An addition to a process definition allowing for adding explanatory comments to the process definition.
API	Application Programming Interface. The interfaces and classes that programmers may use in their own customized applications to access the server.
Application Variable	Dynamic variables that can be defined at application project level to share them across all the processes within a specific application project. This saves effort for creating separate variables for each process.
ASAP	Asynchronous Service Access Protocol. ASAP is a communication protocol based on SOAP and is used to start, manage, and monitor long running services.
Arrow	A connector between nodes. Arrows guide the process flow from one node to another.
Assignee	The person(s) assigned to perform an activity.
Attachment	A document file generated by any application, which has been associated with a process.
BPR	Business Process Reengineering. The field of study which concentrates on how work may be redefined in terms of processes.
Business Calendar	A calendar that specifies working days and times.
Business Process	See *Process*.
Chained-Process Node	A node representing a complex task that can be accomplished independently from the tasks defined in the parent process definition.
Compensation Action	A Java Action that can be defined as compensation for a regular Java Action, e.g. for cleaning up the system and ensuring a consistent state of external systems involved in a transaction.
Complex Conditional Node	A Conditional Node where the condition is specified as a JavaScript expression.
Compound Activity Node	A node that encompasses other nodes in a process; used to group nodes allowing them to be tracked together as a single compound activity.

Conditional Node	A node that directs the process flow along one of several branches, depending on specified criteria.
Database Action	A Java Action allowing for the interaction with external databases that are independent of Interstage BPM.
DB Node	A node that accesses an external database using JDBC.
Delay Node	A node that suspends process execution for a certain amount of time.
Directory Service (DS)	Repository for the entire network's authentication and configuration data. Provides access to services, file servers, databases, and other applications. User and application access to the repository is controlled.
Document Management System (DMS)	The system integrated with Interstage BPM which is used to store attachments, forms, etc. The DMS Adapter is the communication link between the DMS and Interstage BPM.
Due Date	Specifies when an activity is due to be completed once it has become active. A due date also specifies what will happen when it is reached and the activity has not been completed.
EJB	Enterprise JavaBeans.
Email Node	A node that sends out predefined emails.
Error Action	A Java Action that is used to handle specific errors on process definition level, on Remote Subprocess level, and on Java Action level.
Exit Node	A node that identifies the end of a process branch. A process definition has at least one Exit Node.
Form	An HTML or XML file which may be associated with an activity, process instance, or process definition. Forms can be created using Interstage BPM; their appearance can be modified using any XML or HTML editing tool.
Framework Adapter	The Framework Adapter, also called DD Adapter, connects the Directory Service and the Document Management System Adapters. The "DD" is short for "document" and "directory". It handles authentication of the user and manages a consistent user authentication to the Directory Services and Document Management System.
Future Work Item	Work Items that the users may be assigned in future so that the tasks can be planned in advance.
Groups	Visual grouping of activities of the same category.
Groupware	A type of software, which facilitates collaboration.
GUI	Graphical User Interface.
Initiator	The person who starts a new process instance.
Integration Action	A Java Action allowing for accessing external functions from within a process definition.
Interstage BPM Console	User interface which allows a user to create process instances, process definitions and access and respond to work items. It is also used by Interstage BPM Administrators to administrate Interstage BPM.

Iterator Node	A node that generates multiple node instances upon specifying the iterator count.
Java Action	A custom programmed component used for performing the same function several times.
LDAP	Lightweight Directory Access Protocol.
Naming Service	A service that allows clients to find objects based on names.
Node	A graphical representation of a step in a process. Interstage BPM supports different node types, e.g. Activity Nodes, AND Notes, Subprocess Nodes, Conditional Nodes.
No-Operation Action	A built-in Java Action that specifies no operation.
Notification Action	A Java Action allowing for notifying users on events related to process execution. Users can be notified by email, for example, that a process or a single activity has been started.
OnAbort Action	A Java Action that will be executed before a process instance is aborted.
OnResume Action	A Java Action that will be executed before a process instance is resumed.
OnSuspend Action	A Java Action that will executed before a process instance is suspended.
OR Node	A node that splits process flow into multiple parallel branches.
Owner	See *Process Definition Owner* and *Process Instance Owner*.
Participant	A person involved in a process.
Priority	Priority of execution of workflow elements (Process Definition and Activity) set using positive integers.
Process	A sequence of steps that are performed to reach a business goal. Processes are modeled in process definitions.
Process Definition	The representation of a business process in a form that supports automated manipulation. A process definition defines the behavior and properties of the process instances created from it including the flow of control within the process.
Process Definition Owner	The person who created (or last edited) the process definition.
Process Initiator	See *Initiator*.
Process Instance	Represents a single enactment of a specific process definition. The structure of a process instance is exactly the same as the structure of the process definition on which it is based.
Process Instance Owner	By default, the owner of a process instance is the owner of the process definition from which the process instance was created.
Process Participant	See *Participant*.
Project	A container for process definitions, forms, simulation scenarios, attachments, etc. On file system level, a project corresponds to a folder.
QuickForm	A structured, field-based HTML file created using the Interstage BPM Studio.

Remote Subprocess Node	A node representing a subprocess that runs on a remote workflow server.
Role	The name of a person or group (for example, the Manager Role). Each task (activity) in a process is assigned to a role for completion. Roles are equivalent to groups, as defined in Interstage BPM's local group store or in a Directory Service.
Rule	Method used to determine choices on activities for which the rules are defined.
SaaS Mode	The 'Software as a Service' mode of Interstage BPM; deploying and using Interstage BPM in this mode allows you to create multiple tenants, and lease out Interstage BPM to these tenant organizations, who will use it as a service.
Server	In the Interstage BPM context, the component of the workflow management system installed on the computer where the workflow engine provides the run-time environment for a process.
Server Action	A Java Action allowing for interacting with the Interstage BPM Server.
Simulation Scenario	A defined setup for simulating the execution of a process definition on your local computer.
SOAP	Simple Object Access Protocol. SOAP is a standard communication protocol that allows one application to send an XML message to another application. It is used, for example, to access Web Services.
SQL	Structured Query Language.
Start Node	A node that identifies the beginning of a process. Every process definition has one and only one Start Node.
Subprocess Node	A node that represents a complex task. The details of that task are defined in another process definition.
SWAP	Simple Workflow Access Protocol. SWAP passes XML messages over HTTP between workflow servers.
Swimlane	Visual grouping of activities performed by the same Role.
Task	An activity in a process; it typically requires a human decision to be made.
Timer	Expires after a specified interval or at a specified time and date. Timers trigger certain actions when they expire.
User Defined Attribute (UDA)	Data that process participants need to access, modify, or add, such as customer data, order numbers etc. User Defined Attributes are specified in the process definition.
User Extended Attributes	Information that user can specify as properties. User Extended Attributes are specified in the process definition, nodes, and arrows.
User Groups	Sets of individuals who typically share common characteristics. User groups are defined in Interstage BPM's local group store, in a Directory Service or in both systems.

User Profile	User-specific configuration information. This includes information such as whether a users wishes to receive email notifications, email address, and default directory, etc.
Voting Activity Node	A node that allows users to work on an activity in collaboration with one another.
Voting Rule	Rule defined on a Voting Activity Node to determine the outcome of the vote.
Web Service Node	A node that retrieves data from a Web Service and makes it available for further processing.
Workflow	The sequence of activities within a business process.
Workflow Application	A process solution consisting of process definitions, forms, simulation scenarios, attachments, etc. Interstage BPM allows for the creation of Workflow Application projects having a predefined structure. Such applications can be deployed on an Interstage BPM server that can be accessed by Interstage BPM clients.
Workflow Server	An Interstage BPM component that provides the run-time environment for process instances.
Work Item	An activity in the worklist.
Worklist	The list of activities.
WSDL	Web Services Description Language. WSDL is an XML-based language that describes the Web Services an organization offers. It also describes how to access the Web Services.
XML Action	A Java Action that allows you to perform specific operations on UDAs of type XML, for example adding XML substructures, setting text or attribute values in XML, or extracting UDA values from XML data.
XPath	XML Path Language. XPath is a language for finding information in an XML document. It is used to navigate through elements and attributes.
XPDL	XML Process Definition Language

Index