



bonitasoft

open your processes

Bonita Open Solution

Version 5.3

User & Reference Guide

Version 4.0

Change Notice

This document now describes the following new and improved features in Bonita Open Solution 5:

Bonita Studio

- New BPMN2 functionalities:
 - Timer: now includes a timed start event and an intermediate boundary event
 - Messages now include intermediate boundary events
 - Error: new in this version, end and intermediate boundary error events
 - Signal: new in this version, start, intermediate (catch and throw), intermediate boundary and end signal events
- The Palette of elements for Process design has been improved – there are more elements, and the designer can now choose from among 3 views of Palette display
- Filter Actors for a Step by selecting a list of Actor who have acted on previous Steps
- Apply XML elements in Data types and Groovy expressions
- Java Objects can be defined and used more easily; browse function has been added
- “Export Processes and Application” feature now allows easy selection to export the contents of the Process Diagram as a *.bar and/or *.war, with or without Runtime information

Form Builder

- Create read-only forms that show the data as entered by the User (*View Form*) or the data as entered, calculated or otherwise processed up to that Step in the Process (*Recap Form*). *Recap Form* provides a way to see the state of the Process at a given Step.
- A Rich Text Area field allows the User to format text entered in a Form as with a document writer (bold, italic, font, color, etc).
- The table widget now has a graphic interface
- Select rows from a table and save this selection as a variable.
- Improved field validators allow the use of groovy in the error message, and can now be positioned above or below the field/page
- An Editable Grid widget in the form of a table allows the User to edit cells, add/remove rows/columns.

Web applications for User Forms

- Change or import and save/re-use the “look and feel” of the GUI in end user Web applications

User XP

- Any User can choose to open and close an open Step in the Case list, from the list of open Steps stacked up in condensed form.
- The Process Administrator can filter Users, Roles, and Groups lists
- The Process Designer and/or the Process Administrator can add categories that appear with each Case of the Process in Users’ inboxes (this is in addition to the user-defined “labels” function) which cannot be deleted by Users, nor manually assigned / unassigned to cases.
- The Process Administrator can expand User profiles by adding more information.
- The Process Administrator can choose between a history view (which shows a condensed stack of Steps completed to date in the Case) or an overview (ie classic view showing the entire User form) for any Case

API Access

- the Bonita API type access can be set to REST to request data via standard HTTP POST.

The **Bonita Open Solution Connectors Reference Guide** and **Bonita Open Solution Simulation Reference Guide** are available at www.bonitasoft.com/products/BPM_download.php.

Migration Notes

Workspace Migration

Export all your BOS 5.2 Processes in *.bar files, and import them into Bonita Studio 5.3.

Database Migration

As the database schema have changed between BOS 5.2 and 5.3, a migration tool is provided so you can migrate the databases you may have created with version 5.2 using the init-db tool.

To run it, go to Bonita Studio, **Process -> Export Application** and select **Export Runtime**.

In the conf subdirectory, modify `bonita-environment.xml`, `hibernate-core.properties` and `hibernate-history.properties` to fit your database configuration as described in Part 7, How to Configure Bonita Open Solution for a Production Environment.



Be sure to remove the property `hibernate.hbm2ddl.auto`. If you do not, the migration won't work and you might lose the data in your database.

Place the JDBC driver for your database product and version in the server subdirectory.

Run the ant task `migrate-db` using the command

```
ant migrate-db [db-name] 5.2 5.3
```

in the runtime directory where `[db-name]` can be one of the following supported products: H2, Mysql, Postgresql, Oracle, or Sqlserver.

When prompted, you will need to enter the name of the tenant (domain).

Engine Migration

See [How to install Bonita Open Solution](#).

It is recommended that the Bonita Execution Engine not be called by any Processes that may be running during migration.

While the engine is re-indexing, the filter function in User Experience is not usable.

Bonita Open Solution V5.3 User & Reference Guide

Contents

Change Notice.....	2
Migration Notes.....	4
Part 1. Bonita Open Solution Overview.....	15
1.1 Description.....	15
1.1.1 Bonita Studio.....	15
1.1.2 Bonita Forms (Customizable).....	16
1.1.3 Bonita User Experience.....	16
1.2 Licenses.....	17
1.3 How to download, install, and launch Bonita Open Solution 5.3.....	17
Part 2. How to Use Bonita Studio.....	18
2.1 Overview.....	18
2.1.1 Welcome to Bonita Studio.....	18
2.1.2 How to define or change Bonita Studio preferences.....	19
2.1.2.1 Change language.....	19
2.1.2.2 Change to non-Latin alphabet for Bonita Studio labels.....	19
2.2 How to design a process.....	20
2.2.1 Design the Process Steps and Transitions graphically.....	20
2.2.1.1 Use the Palette to add elements to the Whiteboard.....	21
2.2.2 Define Details for a Step.....	23
2.2.2.1 Activity types.....	24
2.2.2.2 Other General Details for a Step.....	24
2.2.3 Define Details for an Event.....	25
2.2.4 Define Details for a Gate.....	25
2.2.5 Define Details for a Transition.....	26
2.2.6 Define a Text Annotation.....	27
2.2.7 Define a Boundary Event.....	27
2.2.8 Define Details for a Diagram.....	27
2.2.9 Define Details for a Pool.....	28
2.2.10 Define Details for a Lane.....	29
2.3 How to Assign an Actor to a Step (Role Mapping).....	30
2.3.1 Assign the Process Initiator to a Step (dynamic).....	30
2.3.2 Assign a group from an LDAP directory to a Step (dynamic).....	31

2.3.2.1	Inputs	32
2.3.3	Assign one or more groups to a Step	33
2.3.3.1	Assign a group from an itemized list (dynamic)	33
2.3.3.2	Assign a manually defined Group (static)	35
2.3.4	Filter a Group	35
2.3.4.1	Select random candidate(s) from a Group list with a predefined Filter	35
2.3.4.2	Create a new Filter	36
2.3.5	Assign an individual Actor (static)	36
2.3.5.1	Assign a previous Actor to a Step (dynamic)	36
2.3.5.2	Assign a list of previous Actors to a Step (dynamic)	38
2.3.6	Edit / Remove Group	39
2.3.7	Use example users / Administrator default for development	40
2.4	How to define Data variables	41
2.4.1	Overview of Data variables	41
2.4.2	Write expressions using the Groovy script editor	43
2.4.3	Promote a local variable to become a global variable	44
2.4.4	Use Java Objects (* . jar) as Variables	44
2.5	How to call a subprocess	46
2.5.1	Map data between parent process and subprocess	47
2.6	How to Loop a Step	48
2.7	How to execute multiple instances of the same Step (multi-instantiation)	49
2.7.1	Configure an Instantiator	49
2.7.2	Configure a Join Checker	51
2.7.3	Define a new Multi-Instantiation	51
2.7.4	Link a SubProcess to a Multi-Instantiation	52
2.8	How to set Timers	53
2.8.1	Create a Start Timer	53
2.8.1	Create an Intermediate Timer	54
2.8.1	Create a Boundary Timer	55
2.9	How to Send and Receive Intermediate Links within a Pool	56
2.9.1	Create a Throw Link	56
2.9.2	Create a Catch Link	57
2.10	How to send and receive messages across Pools	58
2.10.1	Create a Throw Message	58
2.10.2	Create a Catch Message	60
2.10.3	Create an End Message	61
2.10.4	Create a Start Message	62
2.10.5	Create a Boundary Message	62

2.11	How to define an Error Event	63
2.11.1	Create an End Error.....	63
2.11.2	Create a Boundary Error	64
2.12	How to define a Signal Event	65
2.12.1	Create a Throw Signal	65
2.12.2	Create a Catch Signal	66
2.12.3	Create an End Signal	67
2.12.4	Create a Start Signal.....	67
2.12.5	Create a Boundary Signal.....	68
2.13	How to define Transactions	68
2.14	How to configure Connectors	69
2.15	How to define Categories	69
2.16	Install (import) an extension shared in the BonitaSoft Community Contributions	69
2.17	How to manage Processes in Bonita Studio	70
2.17.1	Find Process files.....	70
2.17.2	Create a New Process or Open an existing Process.....	71
2.17.3	Save a Process / Save all Processes.....	71
2.17.4	Duplicate a Process	71
2.17.5	Import a Process	71
2.17.6	Export a Process.....	71
2.17.7	Print a Process.....	72
2.17.8	Close a Process / Close all	73
2.17.9	Delete a Process.....	73
2.17.10	Rename a Process Diagram.....	73
2.18	How to run a Process	74
2.19	How to configure and run a Simulation	75
Part 3.	How to create and customize forms for end users.....	76
3.1	Overview	76
3.2	How to create a Form for a Step.....	77
3.2.1	Design a Step Form	78
3.2.1.1.	Add / remove rows of cells in a Form.....	79
3.2.1.2	Move cells.....	80
3.2.1.3	Merge cells.....	80
3.2.1.4	Define and arrange data fields in a Form	81
3.2.1.5	Define or redefine the contents of a field in a Form	85
3.2.1.6	Restrict the type of data a field will accept	86
3.2.1.7	Resize columns and rows.....	87
3.2.1.8	Change the appearance of a cell, label, or field	87

3.2.2	Create multiple page Forms.....	88
3.3	How to create a Form for a Process.....	90
3.4	How to customize templates for Web Forms	91
3.4.1	Customize the Process Template (Web application)	91
3.4.2	Customize a Step level Template	94
3.5	How to customize the Error template, Welcome page, and Log-in page.....	96
3.5.1	Customize error messages	96
3.5.2	Customize Welcome page.....	97
3.5.3	Customize Log in page	98
3.6	How to customize the Confirmation template and messages.....	99
Part 4.	How to use Bonita User Experience.....	100
4.1	Overview	100
4.2	Bonita User Experience – Developer’s View	100
4.2.1	How to define or change Bonita User Experience preferences.....	102
4.2.1.1	Change Web Browser for User Experience.....	103
4.2.1.2	Change user for whom User Experience will open when a Process is Run	103
4.2.1.3	Keep User Experience contents after closing Bonita Studio	103
4.2.1.4	Change language.....	103
4.3	Bonita User Experience – End User’s View	104
4.3.1	How to execute and manage Steps in a Case	104
4.3.1.1	Assign a Step to another Actor	107
4.3.1.2	Unassign a Step (take back from another Actor).....	107
4.3.1.3	Suspend a Step	107
4.3.1.4	Resume a Step	107
4.3.1.5	Change a Step Priority	107
4.3.2	How to Manage Cases.....	107
4.3.2.1	Define and apply Labels to Cases (User).....	109
4.3.2.2	Hide or unhide Labels in the User Experience Control Panel	109
4.3.2.3	View Case overview	110
4.3.3	Consult workload status (Dashboard).....	110
4.3.3.1	Change Dashboard graph	111
4.3.4	Change User Profile.....	112
4.3.5	Start a new Case.....	112
4.4	Bonita User Experience – Administrator’s View	112
4.4.1	Manage Processes as Administrator.....	113
4.4.2	Manage Cases as Administrator.....	113
4.4.2.1	See Process graphically from inside a Case	114
4.4.3	Configure Global Settings of User Experience	114

4.4.3.1	Show or hide User Experience Settings & Calculate Steps at Risk	114
4.4.3.2	Configure how Users can use Stars and Labels	115
4.4.3.3	Specify the URL for a custom web application	115
4.4.3.4	Customize Process label in Inbox.....	116
4.4.3.5	Fix the count of cases (synchronization)	116
4.4.4	Consult status of all Steps, Cases, and Processes	117
4.4.5	Define Users and User Roles	117
4.4.5.1	Define Users and User Roles.....	118
4.4.5.2	Define Groups.....	118
4.4.5.3	Define User Metadata	119
4.4.6	Define Categories.....	119
4.5	Externally deployed *.war files – End User’s View	120
Part 5.	Bonita Open Solution Hardware, Software and Configuration Requirements.....	120
5.1	Hardware Required	120
5.2	Operating Systems, Databases, Application Servers	120
5.3	Subdirectory files	121
Part 6.	How to install Bonita Open Solution.....	122
6.1	Standard installation (Bonita Open Solution as a library).....	122
6.2	Enterprise installation (Bonita Open Solution as a server)	122
6.2.1	Server side.....	122
6.2.2	Client side.....	123
6.2.3	Jboss 4.x and 5.x installation and deployment	124
6.2.4	JOnAS 4.x and 5.x installation and deployment.....	124
6.3	ReST Installation.....	125
6.3.1	Server side.....	125
6.3.2	Client side.....	125
6.4	Environment switch through APIs	126
6.4.1	Using the ReST API	126
6.5	Configuring logs in Bonita Runtime	126
Part 7	How to configure Bonita Open Solution for a production environment.....	126
7.1	How to configure a database	126
7.2	How to initialize a database	127
7.3	How to configure security.....	128
7.4	How to configure authentication (BonitaAuth LoginContext).....	128
7.5	How to configure Login (BonitaStore LoginContext)	129
7.6	How to configure multi-tenancy	130
Part 8.	How to analyze a problem in Bonita Open Solution.....	132

Figures

Figure 1. Bonita Open Solution	15
Figure 2. Welcome to Bonita Studio	18
Figure 3. The Whiteboard ready to design a Process	19
Figure 4. Change Bonita Studio preferences.....	19
Figure 5. Tooltips provide reminders when defining Details.....	20
Figure 6. The Context Palette appears to the right of a highlighted element.....	20
Figure 7. Events.....	21
Figure 8. Tools Icon	21
Figure 9. Tools Icon	21
Figure 10. Three views of Bonita Studio Palette.....	22
Figure 11. Move Lanes or Pools up or down on the Whiteboard.....	23
Figure 12. Define Details for a Step	23
Figure 13. Define Details for an Event	25
Figure 14. Define Details for a Gate.....	26
Figure 15. Define Details for a Transition	26
Figure 16. Define a Text Annotation	27
Figure 17. Define Details for a Process Diagram.....	28
Figure 18. Define Details for a Pool	28
Figure 19. Define Details for a Lane	29
Figure 20. If an Actor is defined for all Steps in a Lane, you can override this for individual Steps	30
Figure 21. Assign an Actor to a Step	30
Figure 22. Assign the Process Initiator to a Step.....	31
Figure 23. Assign the Process Initiator to a Step via a Role Mapper	31
Figure 24. Assign a group from an LDAP directory to a Step	32
Figure 25. Assign a group to a Step.....	33
Figure 26. Assign a manually define Group to a Step	34
Figure 27. Enter the Name and Description of the Group.....	34
Figure 28. Enter the individuals' names.....	35
Figure 29. Assign an individual Actor	36
Figure 30. Assign an Actor based on a previous assignment.....	37
Figure 31. Name the filter	37
Figure 32. Identify the previous Step in which the Actor is assigned	38
Figure 33. Assign an list of candidates based on a previous assignment	38
Figure 34. Identify previous Steps from which to choose Actors	39
Figure 35. Delete a Group.....	40
Figure 36. Bonita User Experience welcome page (default).....	41
Figure 37. Add a Data variable	42
Figure 38. Add an XML variable	43
Figure 39. Groovy editor	43
Figure 40. Select defined variables to use in Groovy expressions.....	44
Figure 41. Find predefined Groovy functions	44
Figure 42. Evaluate to see result of variables in Groovy expression	44
Figure 43. Browse to find Java Objects to define a Data type	45
Figure 44. Add *.jar files from Menu Bar.....	45
Figure 45. Add *.jar files from within Dependencies.....	46
Figure 46. Create a subprocess Step.....	46
Figure 47. Associate the Subprocess with its parent Process.....	47
Figure 48. Map data between Process and Subprocess	47
Figure 49. Loop a Step.....	48

Figure 50. A looped Step	48
Figure 51. Define an Instantiator	50
Figure 52. Configure Instantiator	50
Figure 53. Define a Join Checker	51
Figure 54. Configure the Join Checker	51
Figure 55. Define parameters for a custom multi-instantiator.....	52
Figure 56. Create a Start Timer	53
Figure 57. Set a Start Timer	53
Figure 58. Configure a Start Timer	54
Figure 59. Create an Intermediate Timer.....	54
Figure 60. Configure an Intermediate Timer	55
Figure 61. Choose a Boundary Timer	55
Figure 62. Set a Boundary Timer.....	55
Figure 63. A Boundary Timer requires an alternate path	56
Figure 64. Select a Throw Link	56
Figure 65. Create a Throw Link	57
Figure 66. Select a Catch Link.....	57
Figure 67. Create a Catch Link.....	57
Figure 68. Select a Throw Message.....	58
Figure 69. Create a Throw Message.....	58
Figure 70. Add Throw Message content	59
Figure 71. Define each data value to be carried by Message	60
Figure 72. Select a Catch Message.....	60
Figure 73. Create a Catch Message.....	61
Figure 74. Select an End Message.....	62
Figure 75. Select a Start Message	62
Figure 76. Create a Boundary Message	63
Figure 77. A Boundary Message requires an alternate path	63
Figure 78. Select an End Error	64
Figure 79. Create a Throw Error.....	64
Figure 80. Create a Boundary Error	65
Figure 81. Select a Throw Signal	65
Figure 82. Create a Throw Message.....	66
Figure 83. Select a Catch Signal.....	66
Figure 84. Create a Catch Signal.....	66
Figure 85. Select an End Signal	67
Figure 86. Select a Start Signal	67
Figure 87. Create a Boundary Signal	68
Figure 88. A Boundary Signal requires an alternate path	Error! Bookmark not defined.
Figure 89. Define a Category for a Process.....	69
Figure 90. Select an extension from the Contributions page	70
Figure 91. Import an extension.....	70
Figure 92. Process management tasks are located in the Task bar	70
Figure 93. Export Application.....	72
Figure 94. Delete a Process	73
Figure 95. Rename a Process Diagram.....	74
Figure 96. Bonita Open Solution initializing the User Experience	75
Figure 97. Open the <i>Bonita User Experience</i> inbox in the default web application	75
Figure 98. Step Form with no customization	76
Figure 99. Three types of Forms in a Process	77
Figure 100. Create an Entry or View form for a Step.....	77
Figure 101. Create a New Form	78

Figure 102. Begin with the Form Builder	79
Figure 103. Add or remove rows and columns in the grid.....	80
Figure 104. Click and drag to move a cell, click the arrow to merge cells	81
Figure 105. Add General Details to a form field	81
Figure 106. Change data field type	82
Figure 107. Form Builder Palette	83
Figure 108. Define a table	84
Figure 109. Configure the appearance of a table	84
Figure 110. Define an Editable Grid	84
Figure 111. Example: Data definitions for a List field	85
Figure 112. Use an uploaded attachment as an image.....	86
Figure 113. Add a Validator to a field	86
Figure 114. Change the size of a column or row.....	87
Figure 115. Change the appearance of a call, a label, or a field	88
Figure 116. Several forms defined in the same step create a multipage form.....	89
Figure 117. Each subsequent page in a multi-page form	89
Figure 118. For preceding Forms, change Submit buttons to Next Buttons	89
Figure 119. Navigate between multiple forms	90
Figure 120. Add an Overview Form at the Process level	90
Figure 121. Customizable Process template and web application template.....	91
Figure 122. Customizable Page template	91
Figure 123. Change the Look and Feel of all process Forms.....	92
Figure 124. Change template for all forms presented in this Process.....	92
Figure 125. Resources for default “black” template.....	93
Figure 126. Download Step-level form template generated by Bonita Open Solution	94
Figure 127. Upload modified Global “Entry” or “View” Template	94
Figure 128. Upload your custom html Form template	95
Figure 129. Upload a custom template for error messages	96
Figure 130. Upload a custom template for the Welcome page.....	98
Figure 131. Upload a custom template for the Log in page	98
Figure 132. Upload a custom html template for a confirmation page at the Step level	99
Figure 133. a custom html template for a confirmation page at the Process level.....	99
Figure 134. Open Bonita User Experience from the Menu bar	101
Figure 135. Open Bonita User Experience from the Bonita Web application	101
Figure 136. Bonita User Experience as deployed by developer/admin, showing Inbox.....	102
Figure 137. Change Bonita User Experience preferences.....	103
Figure 138. Change language from within User Experience	103
Figure 139. Bonita User Experience inbox as seen by end user	104
Figure 140. Managing Steps in Bonita User Experience	105
Figure 141. Pending Step awaiting action by the Actor	106
Figure 142. Managing Cases in Bonita User Experience	108
Figure 143. Apply a Label to a Case	109
Figure 144. Change the characteristics of a Label	109
Figure 145. Apply a Label to Cases.....	109
Figure 146. Manage Labels	110
Figure 147. See a Case history in Bonita User Experience	110
Figure 148. Dashboard shows the status of User’s workload.....	111
Figure 149. Change Dashboard graph.....	111
Figure 150. Change Profile data	112
Figure 151. Start a new Case of a Process	112
Figure 152. Admin menu in Control Panel	113
Figure 153. See the List of all Processes	113

Figure 154. Manage cases in Cases list	114
Figure 155. See Process diagram	114
Figure 156. Change general settings in Bonita User Experience	115
Figure 157. Change Label and Star settings in Bonita User Experience.....	115
Figure 158. Specify the URL for your custom web application	116
Figure 159. Change how names and numbers of Cases are presented.....	116
Figure 160. Reporting shows status of Cases.....	117
Figure 161. Define Users	117
Figure 162. Define User Roles.....	118
Figure 163. Create a new Group	118
Figure 164. Define User metadata	119
Figure 165. Apply a filter to list a subset.....	119

Bonita Open Solution User Guide

Welcome to Bonita Open Solution (BOS). If you don't already have the software, you can download it from the [BonitaSoft web site](#).

This User Guide is intended to help you navigate and use **Bonita Open Solution** Version 5. You can download this document and its updates from the [BonitaSoft web site](#).

Parts 1 to 4 are generally organized in the way that one would employ the software to design, connect, deploy and use it for Business Process Management.

Part 1 gives a basic overview and general information.

Part 2 describes how to use **Bonita Studio**, the graphic process design interface of Bonita Open Solution. This section contains information about how to use Bonita Open Solution to design a process, connect it to external information systems, and run (deploy) it.

Part 3 describes how to create and customize end-user interfaces (**Forms**), which are generated at the Steps you've defined in the process you've designed, via stand-alone Web-based applications.

This is the part of Bonita Open Solution that runs externally and can be widely distributed, accessible by your end users via a web portal or standalone web page, designed by you and integrated into your company or organization's web site or intranet.

Part 4 describes how to use **Bonita User Experience** to develop and test the management of steps in cases that are created each time a process is initiated in its real-world deployment, and the management of the cases and processes behind the execution. This is the Bonita Open Solution management portal, which can be used:

- by the end users who can take action in process steps
- by both end users and administrators to begin new cases of a process
- by the administrator to follow and/or intervene in processes as they are actually executed Step-by-Step, by the end users, and as automated

Part 5 gives hardware, software and general configuration requirements.

Part 6 describes the installation of Bonita Open Solution in Standard (standalone) and Enterprise (client-server) configurations.

Part 7 describes how to configure Bonita Open Solution for a production environment.

Part 8 addresses trouble-shooting Bonita Open Solution.

Part 1. Bonita Open Solution Overview

1.1 Description

Bonita Open Solution comprises three integrated modules:

- Bonita Studio
- Bonita Form Builder (form destined for end user Web application)
- Bonita User Experience

Plus the Bonita Execution Engine underneath it all.

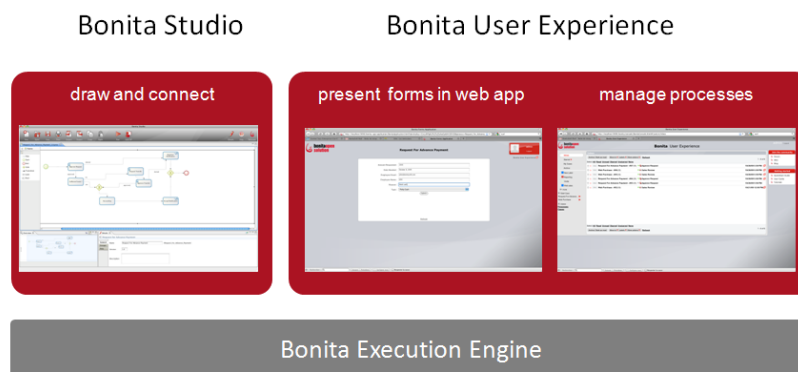


Figure 1. Bonita Open Solution

1.1.1 Bonita Studio

The graphic interface of Bonita Studio allows you to draw processes directly on the Bonita Studio Whiteboard using notation consistent with standard Business Process Modeling Notation (BPMN: www.bpmn.org).

Behind the graphic interface is the execution engine that connects processes to your existing systems and deploys/runs the Process. Thus, this graphic interface is connected directly to the execution engine.

When you design a process in Bonita Studio, you can:

- Assign corresponding flows to parallel Lanes in a Pool
- Assign corresponding Processes to parallel Pools in a Diagram
- Assign a Step type (e.g. human /manual, automatic, or subprocess, among other options)
- Indicate whether a Step is initiated once, or multiple times, during a process flow
- Define the Actors to be assigned to a Human Step, including dynamically by connecting to an external personnel database or list
- Define multiple paths using AND, XOR, or Inclusive logic gates to split and join them
- Define data variables (global and Step-specific)
- Create Messages to hand off data between pools (processes)
- Add Timers and Deadlines to delay or trigger steps as a function of time
- Add Connectors to external systems
- Define and customize Forms for the end user interface

When the Process is designed and defined, deploy (run) it with the Run button. Once it is deployed, users can initiate new cases and the Process can be

- run over and over again indefinitely
- inactivated;
- reactivated; and / or
- archived.

1.1.2 Bonita Forms (Customizable)

At each Process Step that takes an input (i.e., Human Step), Bonita Open Solution automatically creates a default Bonita Form with fields based on the Data Variables you have defined.

When the Process is run in development mode, a basic, un-customized Bonita Form for each Step is presented in the default Bonita Web application.

You can modify the end user interface in the following ways:

- Customize the fields and contents of an individual Form at the Step level (in what order the fields are presented, the field labels, field characteristics, and so on);
- Further customize the layout of an individual Form by modifying or changing the html template for the individual form, and/or modifying or changing the Global Page Template; and / or
- Change the Process template (Web application) in which all Forms appear.

1.1.3 Bonita User Experience

Bonita User Experience (User XP) provides an email-like interface for managing Steps, Cases, and Processes.

In this document we'll look at three ways to interact with the User Experience.

- As the Process Developer (e.g. the person who can design, deploy, modify Processes)
- As the Process Administrator (e.g. the person responsible for managing a set of deployed Processes)
- As an End User (e.g., an employee within the company with responsibility to act in order to complete the Process case, or an external client or customer)

The User Experience for End Users who have tasks to perform in a Process permits them to see what tasks are waiting for them to complete, provides a Form for their data entry, and shows the case histories of Cases in which they have been involved.

The User Experience assigned to the administrator permits the Admin to monitor and interact with Processes deployed by the developer.

The User Experience for the developer permits him/her to see and test all of these functions while developing the Process.

1.2 Licenses

The open source and public license used by Bonita Open Solution, GNU General Public License V2, is in the files which have been downloaded with Bonita Open Solution V5. You can find it in **BOS-5.2 ->license.txt**.

1.3 How to download, install, and launch Bonita Open Solution 5.3

Download Bonita Open Solution. When the zipped folder has been downloaded, extract all files. To launch Bonita Open Solution and begin designing a process:

- open the unzipped **BOS-5** folder,
- open the **Studio** folder, and
- launch the BonitaStudio application file for your operating system.

For software and hardware requirements and how to configure Bonita Open Solution for a server, see [Part 5, Bonita Open Solution Hardware, Software and Configuration Requirements](#).

Part 2. How to Use Bonita Studio

2.1 Overview

For a quick overview of Bonita Studio and a basic getting-started tutorial, see the Bonita Open Solution QuickStart Guide at <http://www.bonitasoft.org/wiki/doku.php?id=quickstartguide>.

2.1.1 Welcome to Bonita Studio

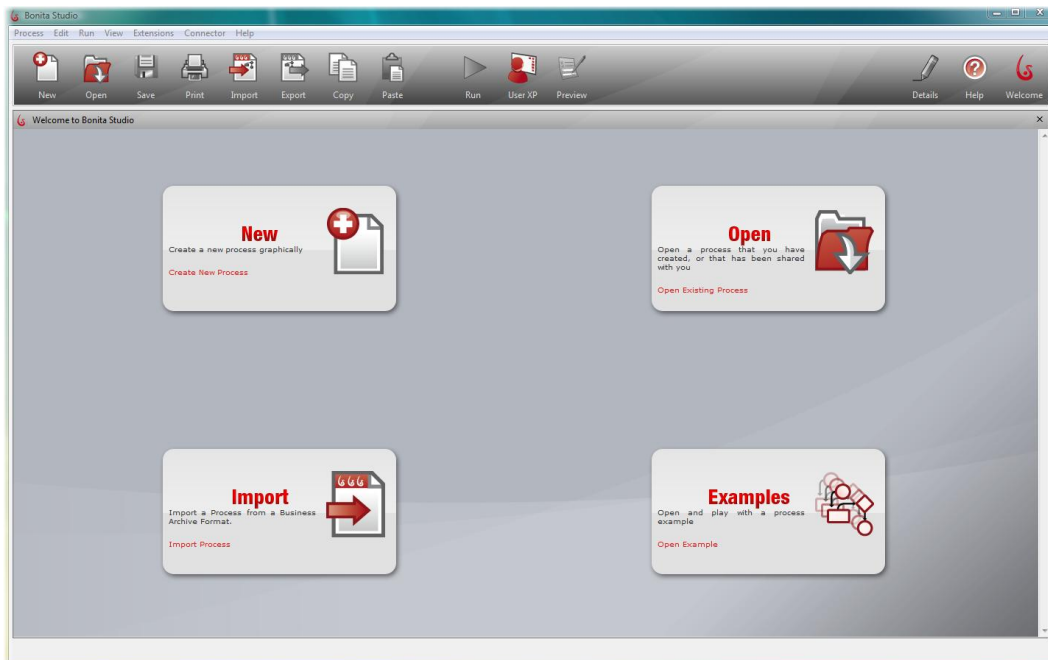


Figure 2. Welcome to Bonita Studio

When you enter Bonita Studio, the Whiteboard is ready to begin:

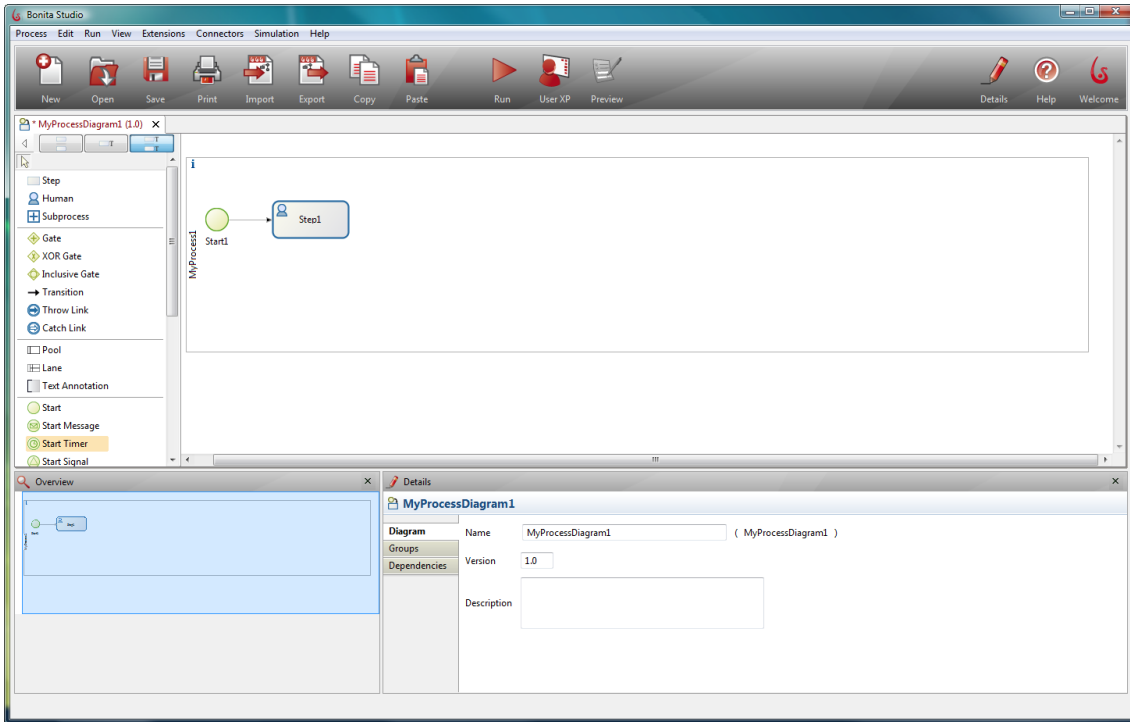


Figure 3. The Whiteboard ready to design a Process

2.1.2 How to define or change Bonita Studio preferences

To change Bonita Studio preferences, go to the menu bar and select **Edit -> Preferences -> Bonita -> Studio**.

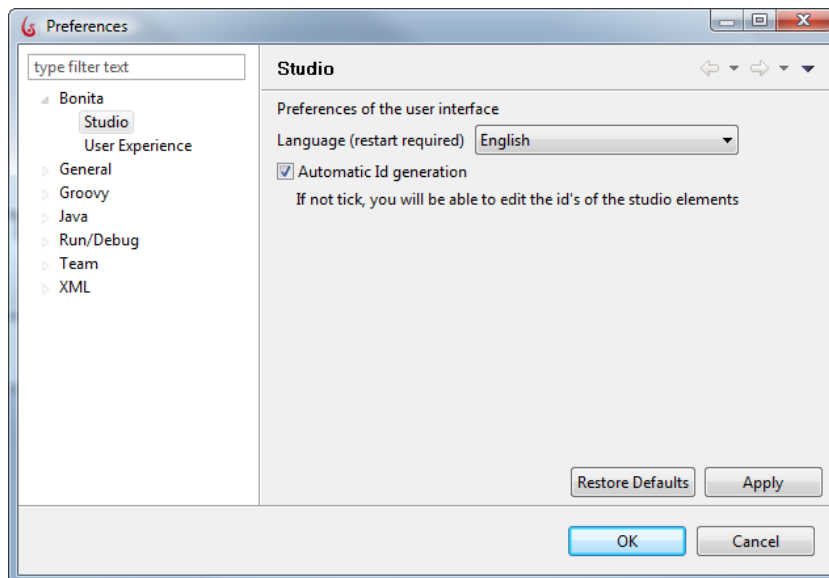


Figure 4. Change Bonita Studio preferences


2.1.2.1 Change language

Choose the language in which Studio is presented. You will need to close and relaunch Bonita Studio after making a change.

2.1.2.2 Change to non-Latin alphabet for Bonita Studio labels

Automatic ID generation is selected by default. Uncheck **Automatic ID generation** if you want to use a non-Latin alphabet. This allows you to manually set the element identifiers in Bonita Studio in an ASCII compliant alphabet, and then use another alphabet for the labels.

2.2 How to design a process

Note that throughout Bonita Studio, there are markers  that indicate the presence of helpful tool tips.

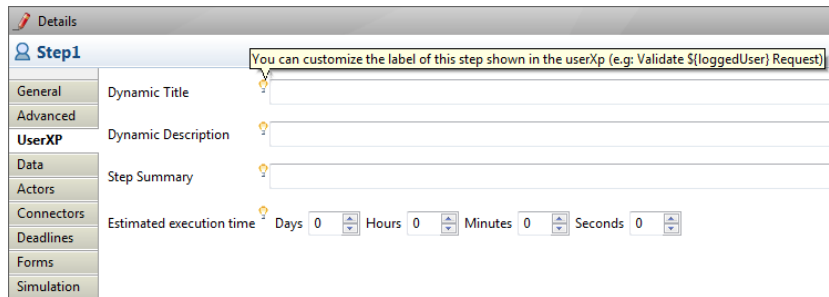


Figure 5. Tooltips provide reminders when defining Details

2.2.1 Design the Process Steps and Transitions graphically

When you select **New**, Bonita Studio opens with a **Start** and the first **Step**. These are presented in a single **Pool**.

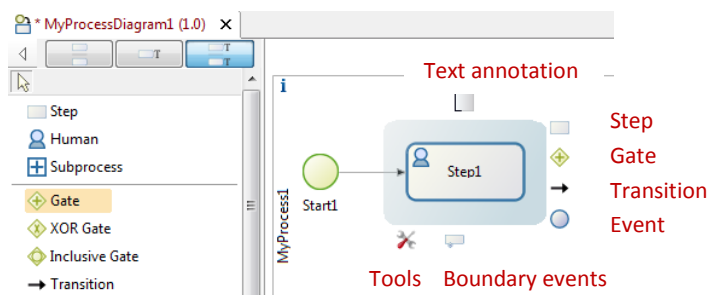


Figure 6. The Context Palette appears to the right of a highlighted element

Whenever you click on a Step, a **Context Palette** appears with 4 icons to choose the next element:

- Step
- Gate
- Transition
- Event

Events include:

- Start
- End
- Start and End Messages
- Intermediate Throw and Catch Messages
- Start and Catch Timers
- Start and End Signals
- Intermediate Throw and Catch Signals
- Intermediate Throw and Catch Links
- Error



Figure 7. Events

The Context Palette below the Step also has a Tools icon that allows you to change the Step to **Automatic**, **Human**, Subprocess, Receive, Throw, Script, or Service.



Figure 8. Tools Icon

The Context Palette below the Step also has a boundary events icon that allows you to add an Error, Message, Timer, or Signal.



Figure 9. Tools Icon

There is a marker above the Step to add a **Text Annotation**.

You can click and drag on any of these elements from this Context Palette, or from the **Palette** in the left hand column of Bonita Studio.

2.2.1.1 Use the Palette to add elements to the Whiteboard

Using the Palette elements, create your top-level process graphically. You can use any one of three versions of the Palette, located to the left of the Whiteboard. All elements are accessible from any of the three. Click and drag the element you want onto the Whiteboard.

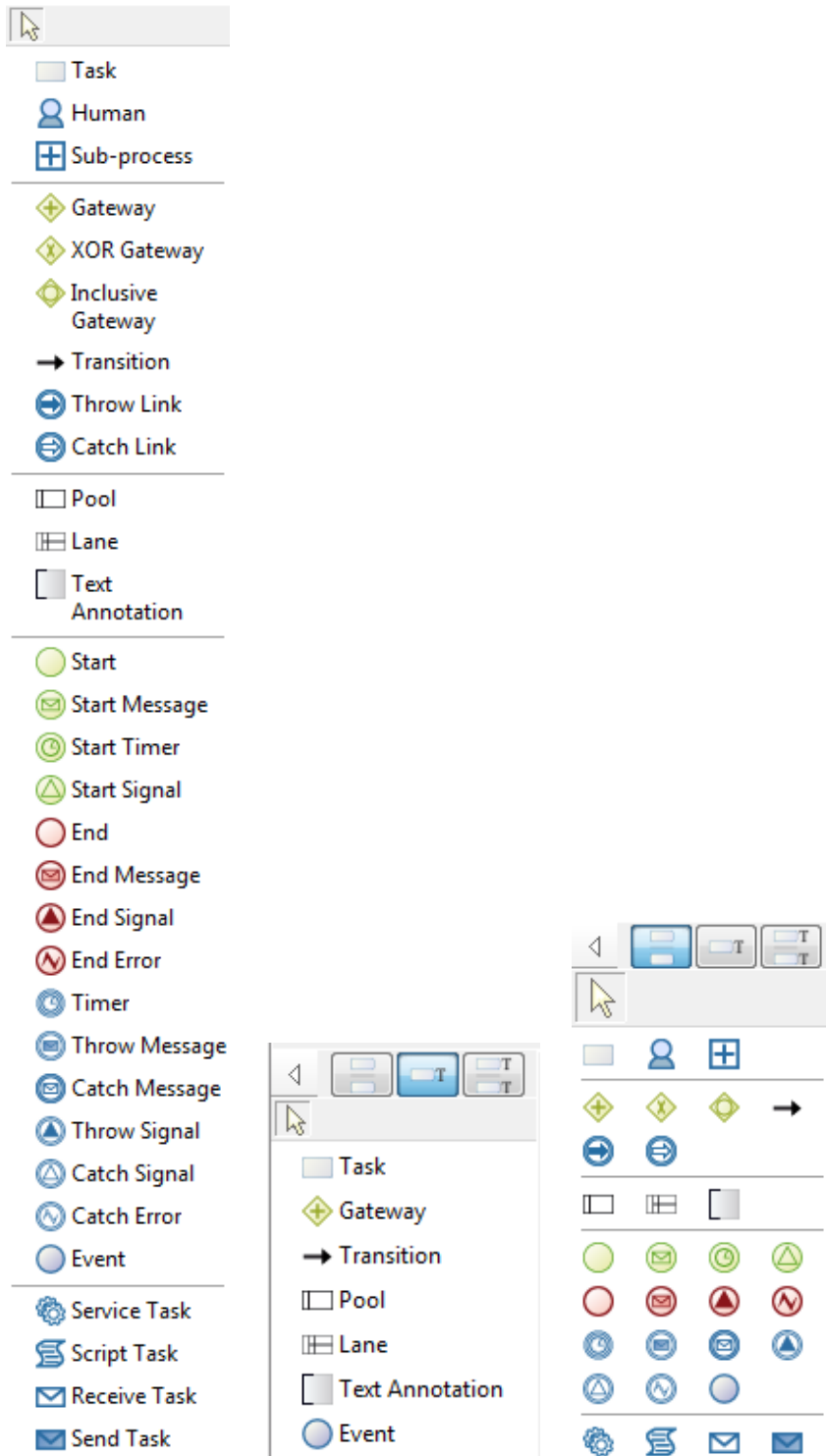


Figure 10. Three views of Bonita Studio Palette

In addition to the elements described above, the Palette also contains elements to create new **Lanes** and **Pools**.

You can add multiple Lanes or multiple Pools by clicking and dragging these elements from the Palette. Use the arrow as indicated to move the entire Lane or the Pool up or down.

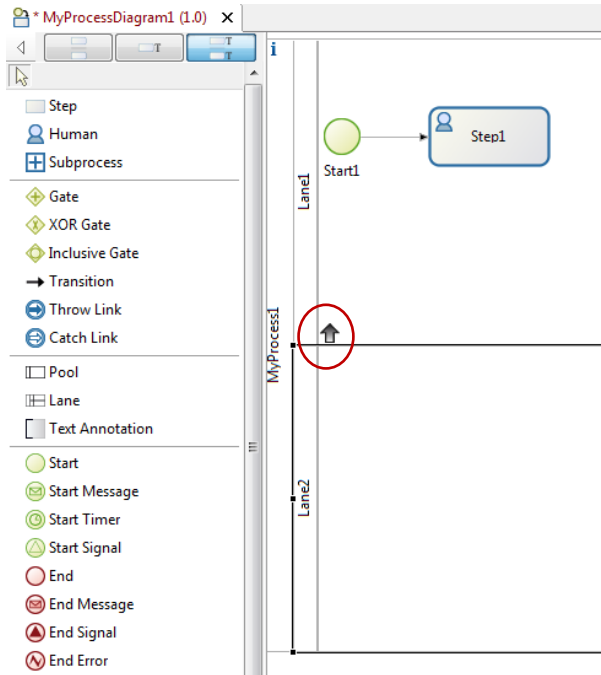
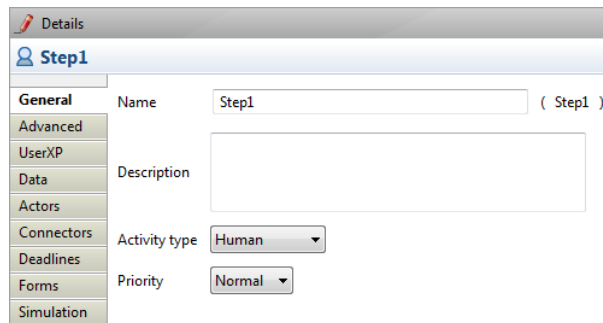


Figure 11. Move Lanes or Pools up or down on the Whiteboard

2.2.2 Define Details for a Step

When you have created a Step, you can define its **Details**.

For each **Step**, the **Details** panel offers the following options:



The screenshot shows the 'Details' panel for a step named 'Step1'. The panel has a sidebar with tabs: General, Advanced, UserXP, Data, Actors, Connectors, Deadlines, Forms, and Simulation. The 'General' tab is selected. It contains the following fields:

- Name:** Step1 (Step1)
- Description:** (Empty text area)
- Activity type:** Human (dropdown menu)
- Priority:** Normal (dropdown menu)

Figure 12. Define Details for a Step

In **General**, you can enter a fixed **Name** and **Description** of the Step. (If you want to apply a dynamic label and/or description to this Step when it appears in Bonita User Experience or an external Web application, go to **UserXP**.)

NOTE: Steps are given identifiers (ID) which you see in parenthesis next to the name you've entered. For example, "An Important Step!!" becomes "An_Important_Step__". This is the identifier that must be used in Groovy script expressions .

2.2.2.1 Activity types

Steps can be Human (manual), Automatic, or Subprocess (for a Step which represents another process as a subprocess). There are also options for Sending and Receiving, Scripts and Service tasks.

Here you can also select the **Priority** for a Human Step: **Normal**, **High**, or **Urgent**.

- Subprocess Step

To call a subprocess, see [How to Call a SubProcess](#).

- Script Task

A Script Task is an Automatic Step with a script that executes when the Step is reached in the Process. Define the script to be executed (and when it is to be executed) by implementing a Connector. See [How to Configure a Connector](#).)

- Service Step

A Service Step is an Automatic Step that calls a Web service when the Step is reached in the Process. Define the Web service to be called (and when it is to be called) by implementing a Connector. See [How to Configure a Connector](#).)

2.2.2.2 Other General Details for a Step

Use **Advanced** to:

- create the capability for multiple instances for the Step (see [How to create multiple instances of the same Step](#))
- Loop the Step (See [How to Loop a Step](#))
- change the Step from synchronous to asynchronous (see [How to define Transactions](#)).

Use **UserXP** to apply a [dynamic title](#), [description](#) and/or [summary](#) for the Step as it will appear in Bonita User Experience. Here you can also specify an estimated execution time for this Step – this will be used in User XP to determine when a Step is overdue, or at risk of being overdue.

Use **Data** to set variables. See [How to define Data variables](#).

Use **Actors** to assign a task to a User (for Human Steps only). See [How to Assign a Step to an Actor](#).

Use **Connectors** to connect Steps in the Process to external information systems. (See [How to Configure a Connector](#).)

Use **Deadlines** to specify duration or a deadline date by which a Step must be completed. When the duration has completed or the deadline date/time reached, a Connector action is triggered, so you'll specify that too. (See [How to Configure a Connector](#).)

Use **Forms** to define web-based end-user applications. See [How to create and customize Forms for end users](#).

Use **Simulation** to define parameters to be used when running a Simulation of the Process. For information about how to configure and run a simulation, see [How to configure and run a simulation](#).

2.2.3 Define Details for an Event

For each **Event**, the **Details** panel offers the following options under **General**:

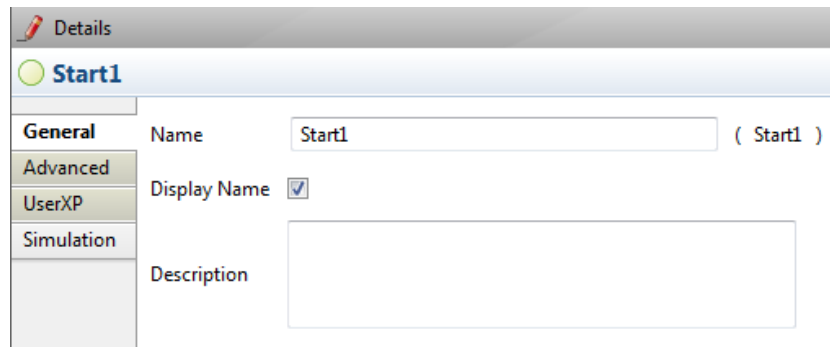


Figure 13. Define Details for an Event

Events are named `EventX` by default (X increments 1 each time an event is added). You can rename an Event by changing the entry in the Name field, and choose to display its name on the Whiteboard or not.

Advanced allows you assign an Event to an asynchronous sequence. See [How to define Transactions](#).

Use **UserXP** to create a dynamic title, description and/or Summary for the Event as it will appear in Bonita User Experience, in the same way as you [Define General Details for a Step](#).

Use **Simulation** to define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

2.2.4 Define Details for a Gate

There are 3 types of Gates available:

- AND
 - all inputs must arrive before Process passes through Gate
 - if there are multiple outputs, all will fire (unless Conditions are defined on Transitions)
- XOR
 - the first input to arrive will allow the Process to pass through the Gate
 - If there are multiple outputs, only single one will fire
- Inclusive (XOR join, AND split)
 - the first input to arrive will allow the Process to pass through the Gate
 - if there are multiple outputs, all will fire (unless Conditions are defined on Transitions)

For more information about defining Conditions on Transitions, see [Define Details for a Transition](#).

For each **Gate**, the **Details** panel offers the following options under **General**:

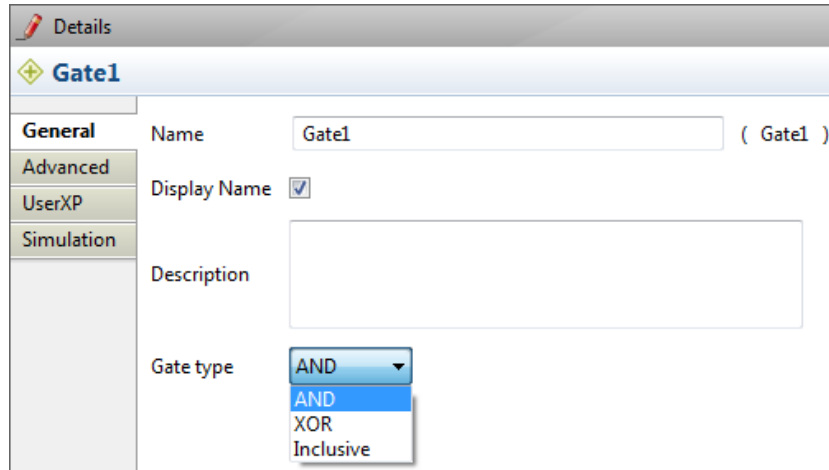


Figure 14. Define Details for a Gate

Gates are named GateX by default (X increments 1 each time a Gate is added). You can rename a Gate, and choose to display its name on the Whiteboard or not.

Advanced allows you to assign a Gate to an asynchronous sequence. See [How to define Transactions](#).

Use **UserXP** to apply a dynamic title, description and/or Summary for the Gate as it will appear in Bonita User Experience, in the same way as you [Define General Details for a Step](#).

Use **Simulation** to define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

2.2.5 Define Details for a Transition

For each **Transition**, the **Details** panel offers the following options under **General**:



Figure 15. Define Details for a Transition

- **Name and Description**

You can display the name you choose for the Transition (e.g. “If not available”), the Condition represented by the Transition (e.g. “Car_available==false” or “!Car_available”), or display no label at all (select the Name and Delete it).

- **Default flow:** check this option to select this path “by default” when all other conditions evaluate to “false.”
- **Condition:** use this field to enter an expression in Groovy script language; when the condition of this expression is met, the Process follows this path. The check box in the **Details** panel offers the option to show either **Condition** or **Name** in the Process on the Whiteboard.
 - **Edit expression:** select this option to bring up the Groovy editor. With this editor, you can create both Condition and Rule elements.
- **Automatic layout:** if this box is checked, Bonita Studio will automatically choose and fix the exit and entry points for the Transition arrow. You can change the arrangement of arrows manually by unchecking this.

Use **Simulation** to define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

2.2.6 Define a Text Annotation

You can add labels attached to Whiteboard elements by clicking and dragging the **Text Annotation** icon that appears in the Context Palette for the element.

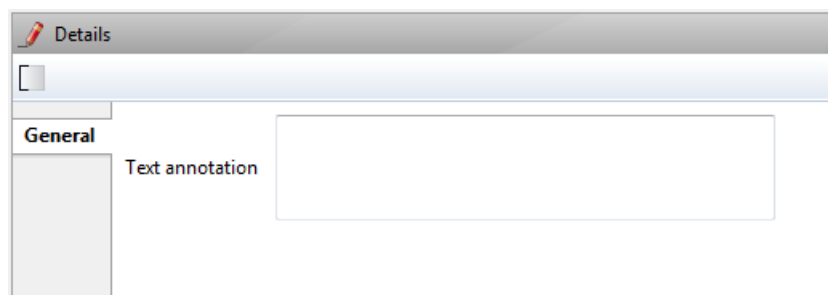


Figure 16. Define a Text Annotation

Enter the text in **Details -> General**. You can click and drag the Text Annotation on the Whiteboard; it will remain attached by a thread to its element.

2.2.7 Define a Boundary Event

See [How to define an Error event](#), [Create a catch message](#), [How to set Timers](#), and [How to define a Signal Event](#).

2.2.8 Define Details for a Diagram

Define the details of a Diagram by clicking on the Whiteboard outside all Pools to go to **Diagram -> Details**:

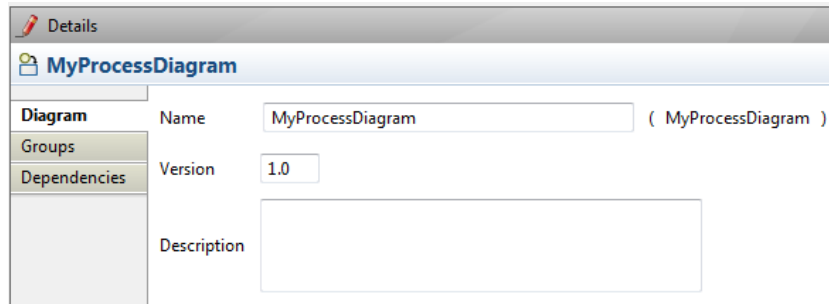


Figure 17. Define Details for a Process Diagram

- **Diagram: Name, Version, and Description:** Use these fields to add information about the overall Process or the part of a Process described in a Pool. You can save different versions of the Process by changing the Version number before you save.
- **Groups:** Here you can *Create, Remove, or Edit* Groups of Actors to whom Steps can be assigned. See [Assign an Actor to a Step](#).
- **Dependencies:** Use this to select resources to be embedded with the process (connectors, libraries, scripts). You can also import *.jar files.

2.2.9 Define Details for a Pool

A Pool is equivalent to a Process. There can be multiple Pools (Processes) in a single Process Diagram. Define the Pool details: go to **Pool -> Details**:

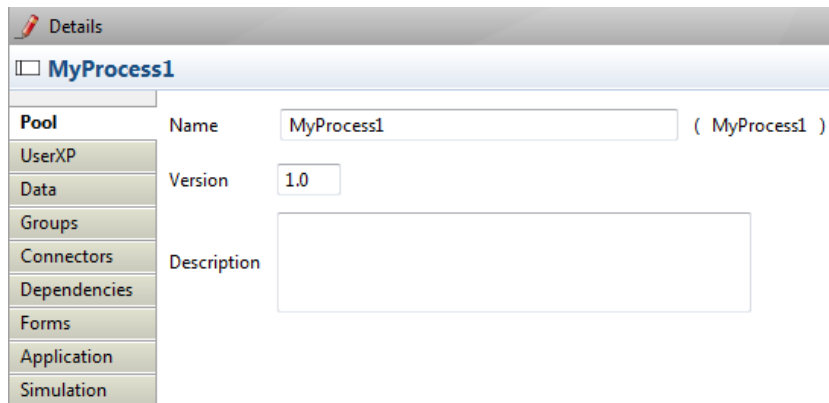


Figure 18. Define Details for a Pool

- **Pool: Name, Version, and Description:** Use these fields to add information about a Pool. You can save different versions of the Pool by changing the Version number. When you Run or Export a process, you can select which version to deploy.
- **UserXP:** Use this to *add or delete* the fixed (unchangeable) Category names that will be available to the User in User Experience for [managing Cases of the Process](#).
- **Data:** Use this to assign global variables for the entire Process contained in this Pool. See [How to define Data variables](#).
- **Groups:** Here you can *Create, Remove, or Edit* Groups of Actors to whom Steps can be assigned. See [Assign an Actor to a Step](#).

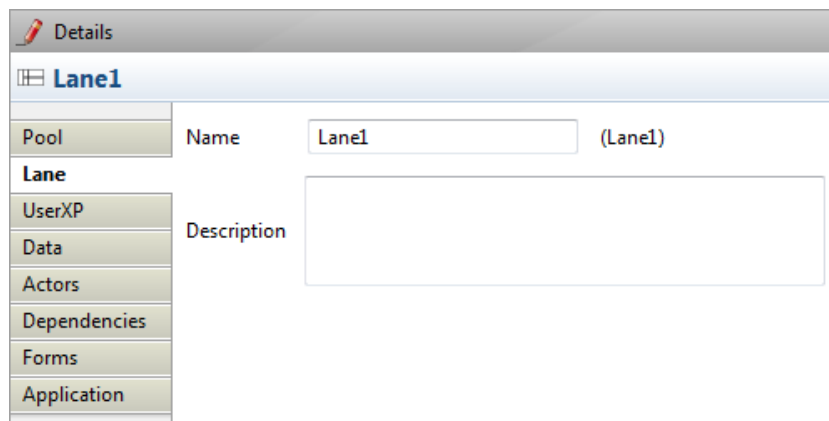
- **Connectors:** Use this to design connections between a Process and external systems. See [How to Configure Connectors](#).
- **Dependencies:** Use this to select resources to be embedded with the process (connectors, libraries, scripts). These can then be exported with the *.bar file. (Note that internally defined applications such as html templates are added automatically to the *.war file).

You can also import *.jar files here.

- **Forms:** Use this to create a **Form** to accept or present data through the end User Interface. See [How to create a Form for a Step](#).
- **Application:** Use this to manage your Web applications, and imported resources associated with them such as html templates. See [How to customize templates for Web applications](#) and [How to customize the Welcome page, Log-in page, Error message and Confirmation messages](#). **Process -> Export application** allows you to export these html templates and associated resources. See [Export a Process](#).
- **Simulation:** Use this to add data that is used only for a Simulation; it is not applicable to the Process as it is deployed. See [How to Configure and Run a Simulation](#).

2.2.10 Define Details for a Lane

The only significant differences for the Details of a Lane are in the **Lane** tab:



The screenshot shows a web interface for defining a Lane. At the top, there is a 'Details' tab. Below it, a sidebar lists various configuration options: Pool, Lane (selected), UserXP, Data, Actors, Dependencies, Forms, and Application. The main area shows the 'Name' field with the value 'Lane1' and '(Lane1)' next to it. Below the name field is a large empty text area for the 'Description'.

Figure 19. Define Details for a Lane

Lane: Name and Description: Use these fields to add information about the Lane.

Actors allows to define a single Actor for all Steps in this Lane. If you define another Actor for a Step and select **Override actors defined in Lane**, the Step Actor definition will override the Lane Actor definition.

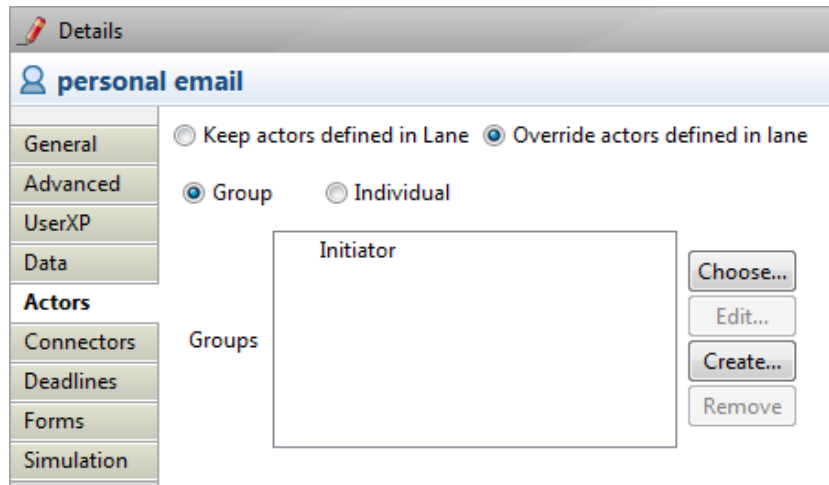


Figure 20. If an Actor is defined for all Steps in a Lane, you can override this for individual Steps

User XP, Data, Dependencies, Forms and Applications are used in the same way as for a Pool. See [Define Details for a Pool](#).

2.3 How to Assign an Actor to a Step (Role Mapping)

An **Actor** is the individual or group of individuals who are called to complete a Step. All Human Steps need to have an Actor assigned. You can assign a Process Initiator, a group (at runtime or statically), an individual selected or filtered from a group (at runtime or statically), or an individual (selected at runtime or statically). This is done using role mapping.

Click the step, then go to **Details -> Actors**.

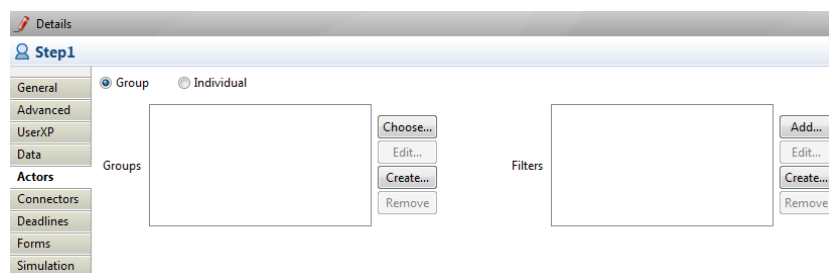


Figure 21. Assign an Actor to a Step

2.3.1 Assign the Process Initiator to a Step (dynamic)

The Initiator is the person who starts a process. There are two ways to identify the Initiator of a Process. Both use Bonita Open Solution role mappers.

- 1) In **Details -> Actors -> Group**, select **Choose**. When the **Assign Actors** window appears, select **Initiator** and **Finish**.

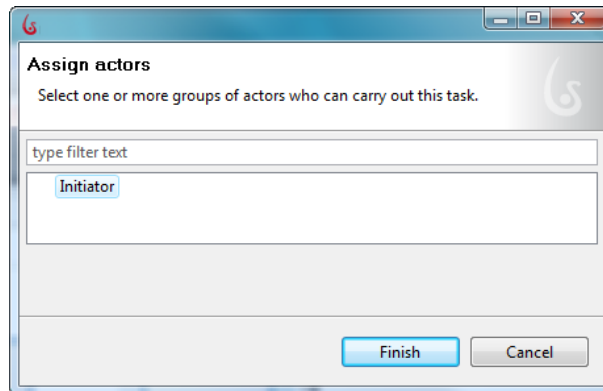


Figure 22. Assign the Process Initiator to a Step

2) In Details -> Actors -> Group -> Create -> Bonita, select Process Initiator.

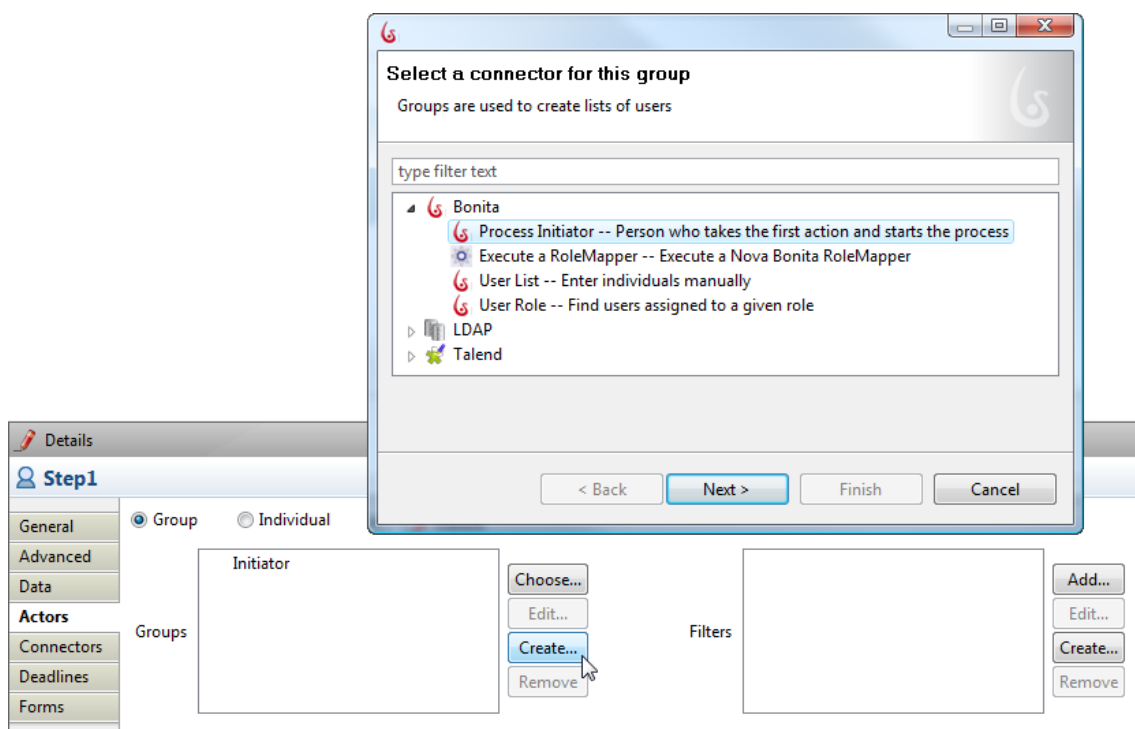


Figure 23. Assign the Process Initiator to a Step via a Role Mapper

When the **Set the name of this group** window appears, enter a **Name** and **Description** for the Process Initiator. This name can now be used in variable expressions.

2.3.2 Assign a group from an LDAP directory to a Step (dynamic)

Select **Group -> Create**. When the **Select a connector for this group** role mapper window appears, select **LDAP**.

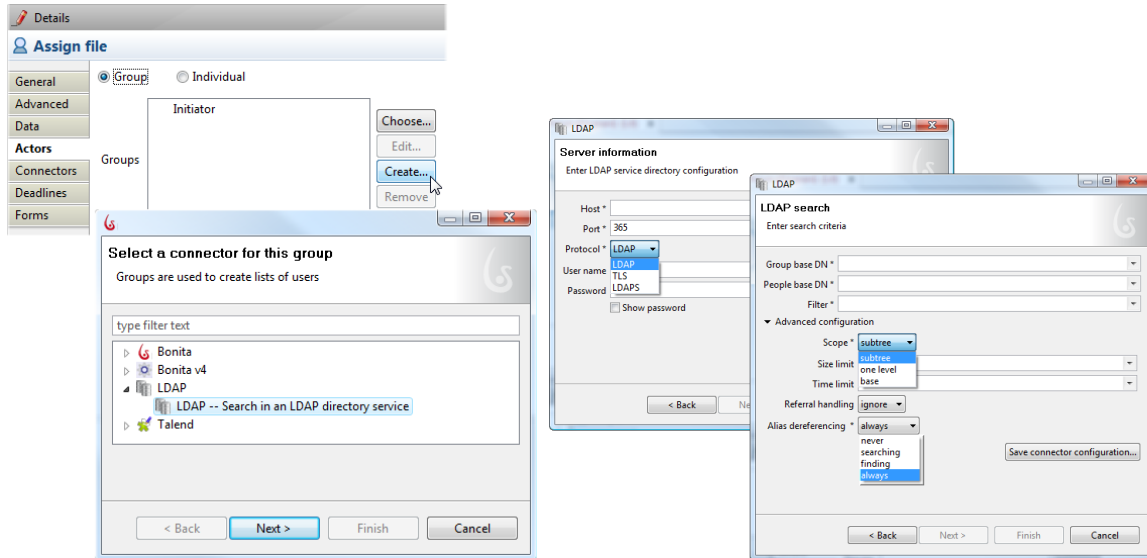


Figure 24. Assign a group from an LDAP directory to a Step

2.3.2.1 Inputs

Enter the **Name** and **Description** of the Group.

LDAP		
Category: Database		
Connector description: search in an LDAP directory and return a list (with specified attributes)		
Input	Description	Type
Host	IP address or name of server hosting LDAP directory	string
Port	LDAP directory port number	number
Protocol	choose LDAP, LDAPS, or TLS	select
User	LDAP User name	string
Password	LDAP user password	string
Group base DN	The DN that contains group records	String
People base DN	DN at which to start search	String
Filter	specify a subset in the LDAP directory set (e.g. Multiple Random – to select N candidates from specified groups Unique Random – to randomly select a single candidate)	string
Scope	<ul style="list-style-type: none"> • subtree (the entire subtree starting at the base DN) • onelevel (entries immediately below the base DN) 	select

	• base (search just the named entry)	
Size limit	maximum number of entries to return	number
Time limit (in seconds)	maximum time to allow search to run	number
Referral handling	Follow or Ignore referrals	select
Alias dereferencing	Always: Always dereference aliases. Searching: Dereferences aliases only after name resolution Never: Never dereference aliases. Finding: Dereferences aliases only during name resolution.	select

When you have saved the Group, you can access it for other Steps in this Process.

2.3.3 Assign one or more groups to a Step

A **Group** of users can be created from an external directory, or can be defined internally as a List of users. Filters can be added to select a subset of members from a defined Group.

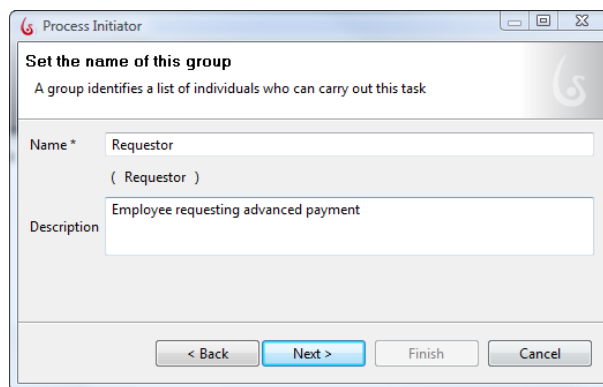


Figure 25. Assign a group to a Step

2.3.3.1 Assign a group from an itemized list (dynamic)

First, define the variable that will contain the list of names. See [How to define Data variables.](#)

To use the variable you have created as an input for the Actor assignment, click on the Step.

Select **Group -> Create**. When the **Select a connector for this group** role mapper window appears, select **Bonita -> User List**.

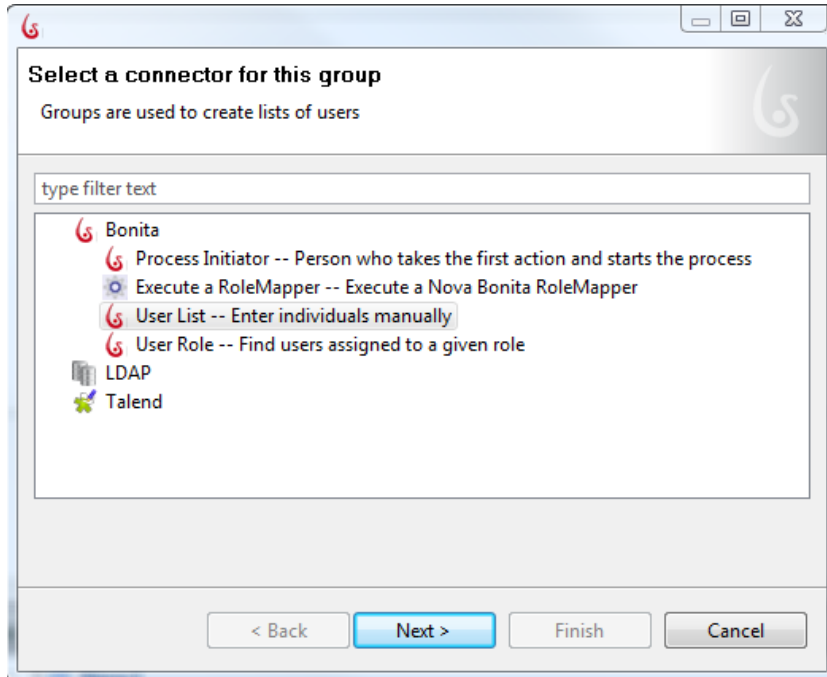


Figure 26. Assign a manually define Group to a Step

Enter the **Name** and **Description** of the Group.

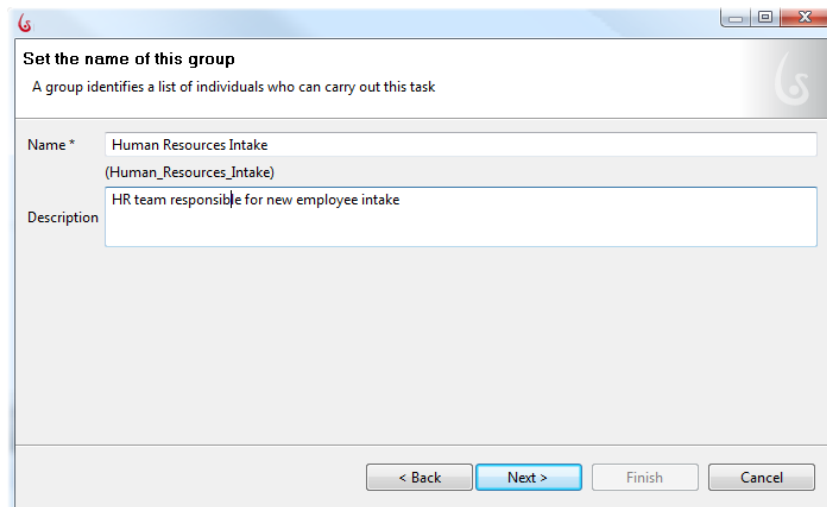
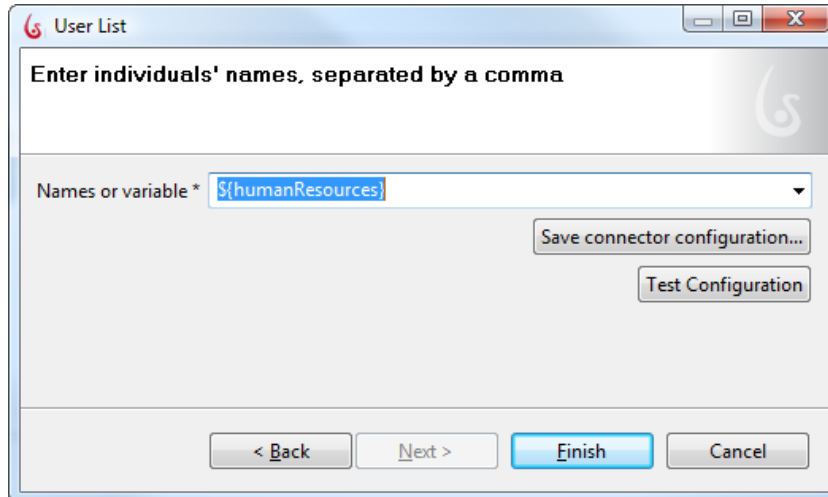


Figure 27. Enter the Name and Description of the Group

In **Users**, select the variable you have created with the list of names.



2.3.3.2 Assign a manually defined Group (static)

Select **Group** -> **Create**. When the **Select a connector for this group** role mapper window appears, select **Bonita** -> **User List**. Enter the **Name** and **Description** of the Group.

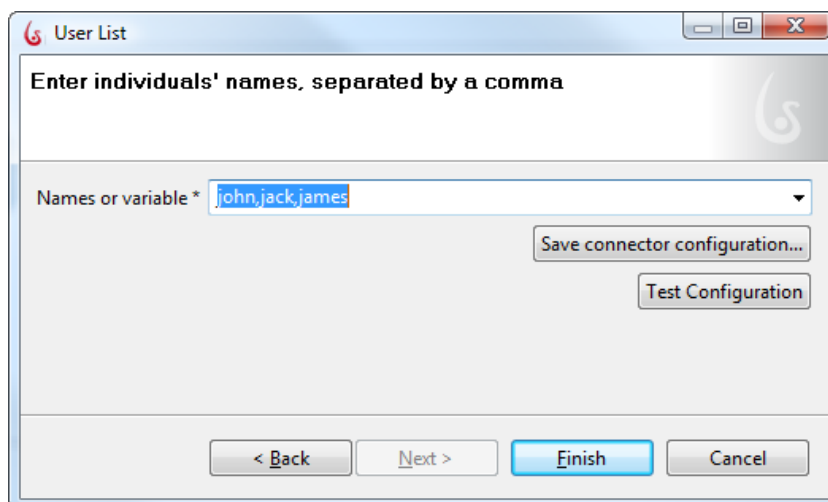


Figure 28. Enter the individuals' names

In **Users**, enter individuals' names in the form john, jack, james (comma, no space).

When you have saved the Group, you can access it for other Steps in this Process.

2.3.4 Filter a Group

When you have a list of individuals, either from a list defined as a variable or an LDAP data file, you can **Add** a predefined filter or **Create** your own to further narrow the list of possible Actors.

2.3.4.1 Select random candidate(s) from a Group list with a predefined Filter

To select random candidates from a User list, go to **Group** -> **(Filters) Add filter** -> **Bonita**.

Select **Multiple Random** to randomly select a specified number of candidates from the group list.

Enter the **Name** and **Description** of the Filter.

Enter the **Number** of candidates to randomly select from the group list.

Select **Unique Random** to randomly select a single candidate from the group list.

Enter the **Name** and **Description** of the Filter.

2.3.4.2 Create a new Filter

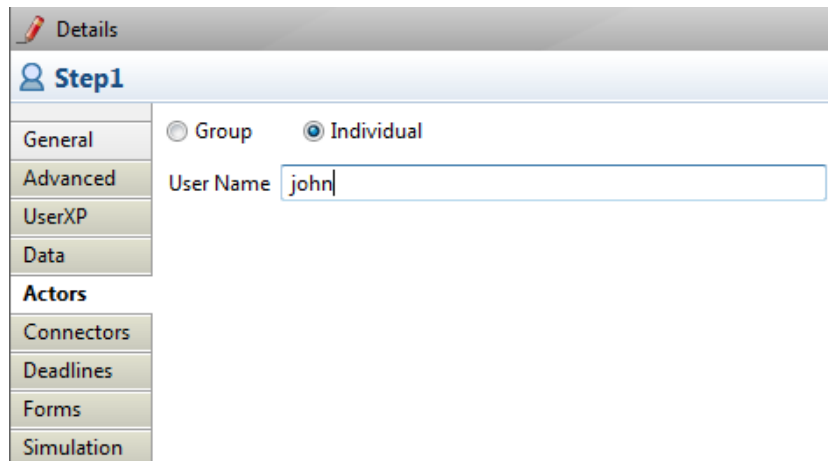
To create a filter, Select **Group -> (Filters) Create**.

The **Set the description of your filter** wizard will appear. This is a connector - see [How to Configure Connectors](#) for details of how to complete this wizard.

2.3.5 Assign an individual Actor (static)

In **Details -> Actors**, select **individual** and enter the **User name**. To assign an individual in this way, the User must be predefined. See [Define Users and User Roles](#).

During development, you can assign pre-defined example Users here also. Use john, james, or jack (lower case).



The screenshot shows a configuration window titled 'Details' for 'Step1'. On the left is a navigation menu with options: General, Advanced, UserXP, Data, Actors, Connectors, Deadlines, Forms, and Simulation. The 'Actors' section is currently selected. Within this section, there are two radio buttons: 'Group' (unselected) and 'Individual' (selected). Below the radio buttons is a text input field labeled 'User Name' containing the text 'john'.

Figure 29. Assign an individual Actor

2.3.5.1 Assign a previous Actor to a Step (dynamic)

To assign the same Actor – one who was assigned to complete a previous Step-- to a Step, Select **Group -> (Filters) Add filter -> Bonita -> Actor continuity**.

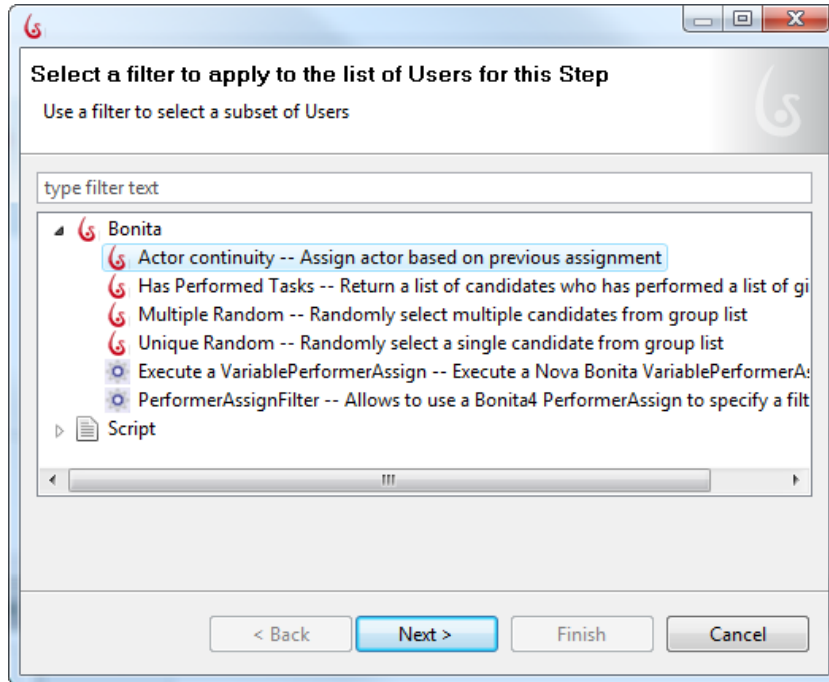


Figure 30. Assign an Actor based on a previous assignment

Enter the **Name** and **Description** of the Filter.

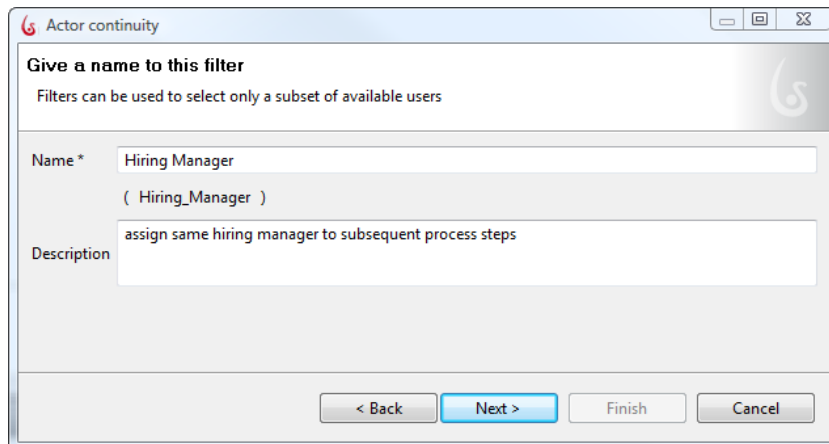


Figure 31. Name the filter

Reference step name: Enter or select the name of the Step whose Actor will be the same for this step.



Figure 32. Identify the previous Step in which the Actor is assigned

2.3.5.2 Assign a list of previous Actors to a Step (dynamic)

To assign a groups Actors who have performed previous Steps, Select **Group -> (Filters) Add filter -> Bonita -> Has Performed Tasks**.

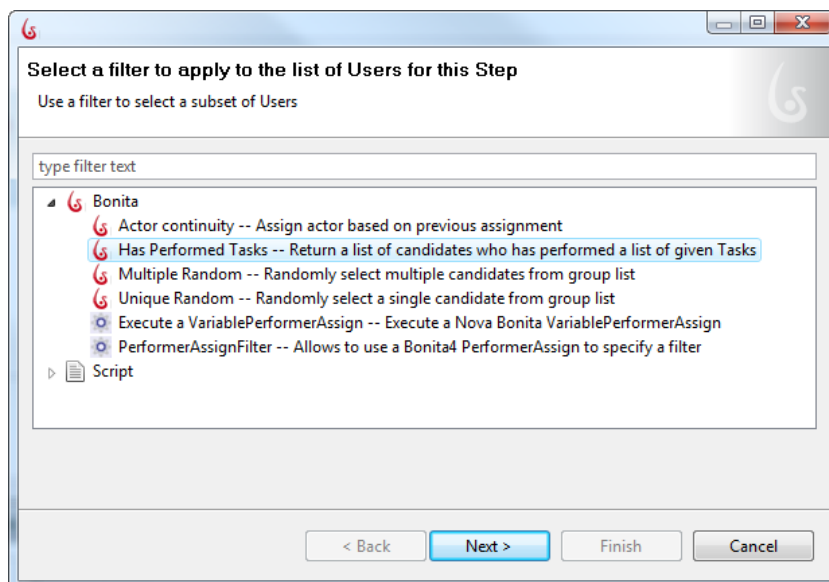


Figure 33. Assign an list of candidates based on a previous assignment

Enter the **Name** and **Description** of the Filter.

Enter the list of tasks from which to select previous Actors.

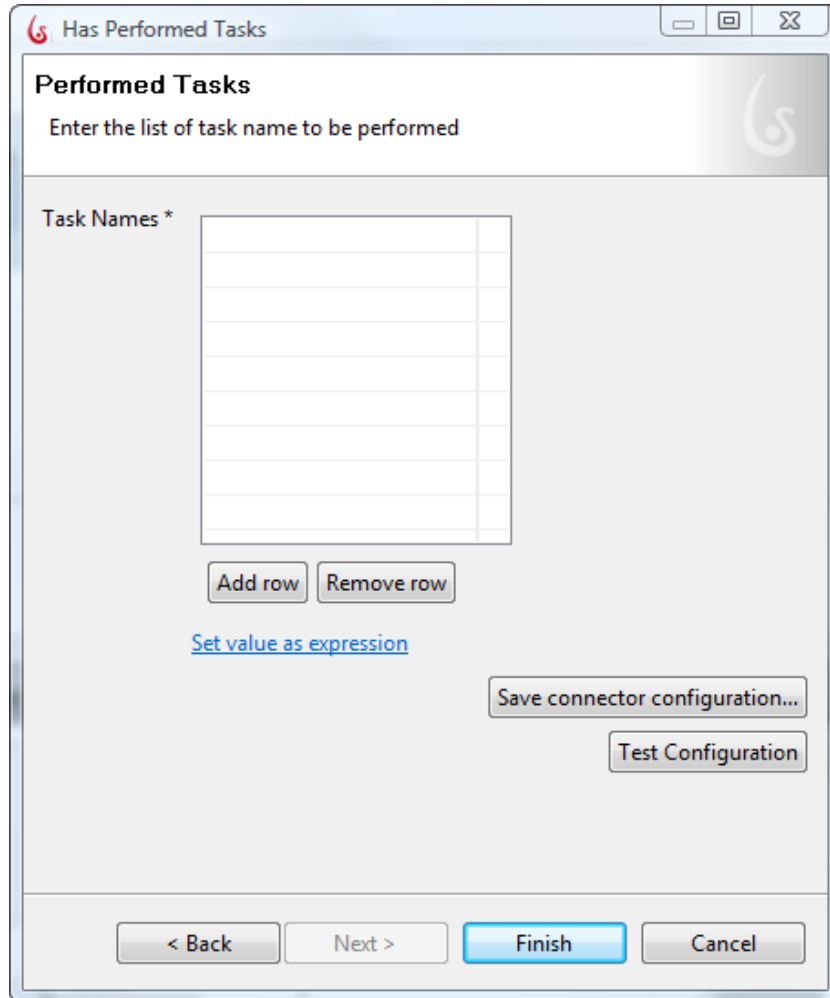


Figure 34. Identify previous Steps from which to choose Actors

2.3.6 Edit / Remove Group

When a Group is created, it is saved in the Details at the Process level.

You can **Edit** or **Remove** a Group from a Step.

To Edit or Remove a Group, select **Step** -> **Actors** -> **Group** and select the Group. Then select **Edit** to make changes or **Remove** to remove it from this Step.

To delete a Group entirely, click the Process (or Pool) to bring up the **Details** panel and select **Groups**. Select the Group you want to delete from Bonita Open Solution and select **Remove**.

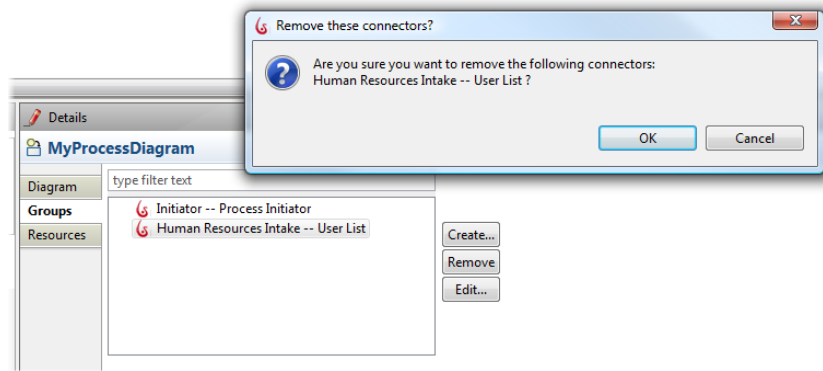


Figure 35. Delete a Group

2.3.7 Use example users / Administrator default for development

When you first open Bonita Open Solution, by default you are automatically logged in as the Administrator (`admin`), with the password `bpm`. You can change this configuration in **Edit -> Preferences -> Bonita User Experience**.

There are also 3 predefined non-administrator “example” users that you can play with as a developer.

You can define any of the following as Actors in Human Steps by assigning one of them as an individual, or you can create a **Group** with one or more of them:

```
admin
john
jack
james
```

The User Experience inbox will open for the admin by default. To interact as John, Jack, or James as the Process runs, you can log in as any of these Actors:

- Go to your Bonita User Experience inbox.
- Logout as admin.
- The **Welcome to Bonita User Experience** window will appear.



Figure 36. Bonita User Experience welcome page (default)

Enter Username: john or jack or james (note lowercase)
Enter Password: bpm

When you are logged in as “John,” you will see John’s inbox and the tasks assigned to John. John, Jack, and James have also been given the ability to start Process Cases.

2.4 How to define Data variables

2.4.1 Overview of Data variables

Data variables can be defined globally (for the Pool) or locally (for an individual Step.) Global variables are available to all Steps in the Process. Local variables can be used only within the Step where they are defined.

To define a Global variable, click on the Whiteboard to see the Details associated with the Pool or Lane. To define a local variable, click on a Step.

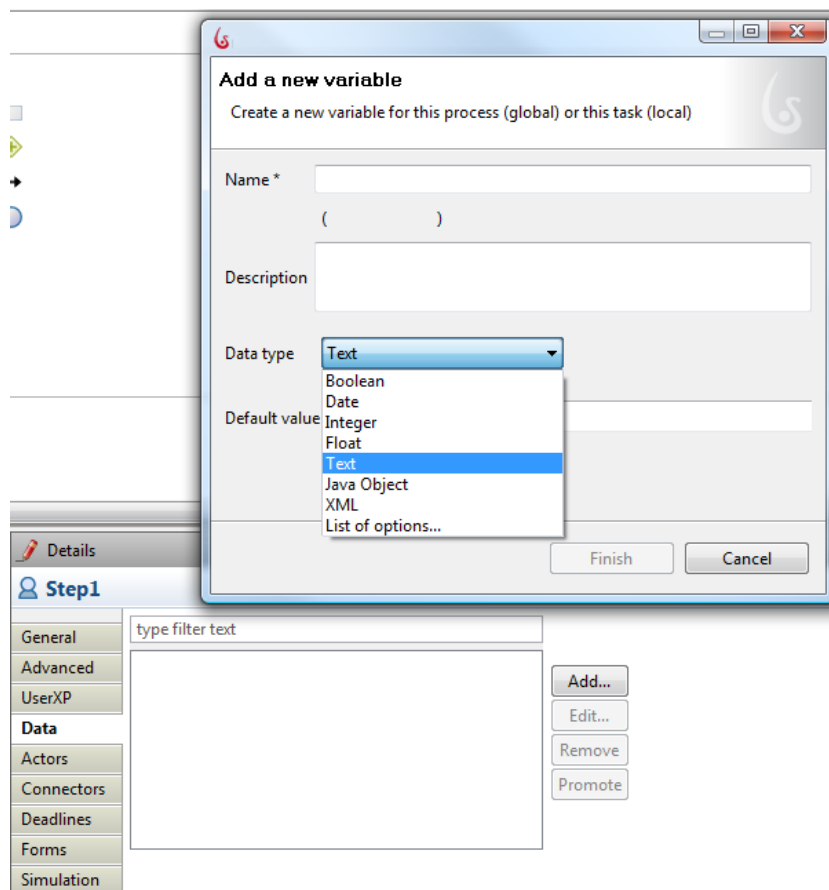


Figure 37. Add a Data variable

Go to **Details** -> **Data** -> **Add**.

Enter the **Name** and **Description** of the variable.

Select a **Data type**:

- Boolean
- Date
- Integer
- Float (floating decimal)
- Text
- Attachment (global variables only)
- Java object (see [Add User-Defined Data in the form of Java files](#))
- XML
- List of options (a set of items from which a user can choose one or more)

For all data types except **List of options**, you have the option of specifying a **Default value** or an expression.

Edit Expression... brings up the Groovy script editor. See [Write expressions using the Groovy script editor](#).

To define an XML variable:

Enter the **Name** and **Description** of the Variable. Set the XML namespace and XML element for this variable. **Add** an XML schema, if desired, by uploading a file.

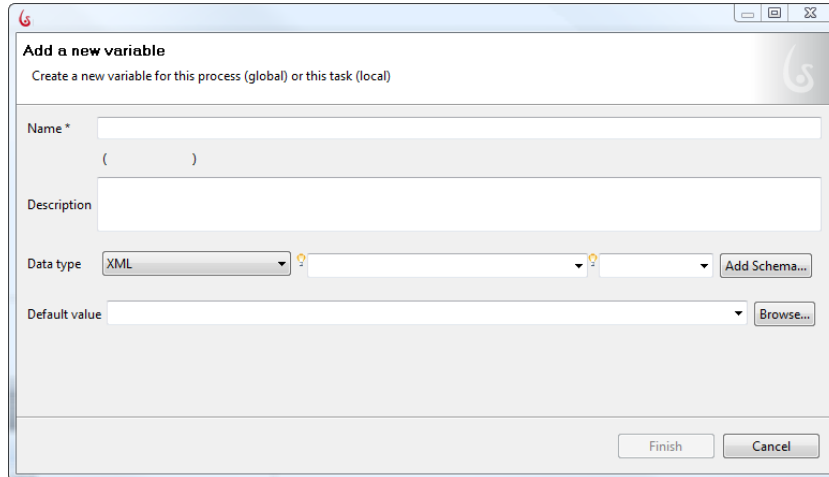


Figure 38. Add an XML variable

2.4.2 Write expressions using the Groovy script editor

For more information on writing scripts in Groovy, see <http://groovy.codehaus.org/>.

The Groovy editor is available throughout Bonita Open Solution, to apply scripts in the form of expressions. It is presented as `Edit expression`.

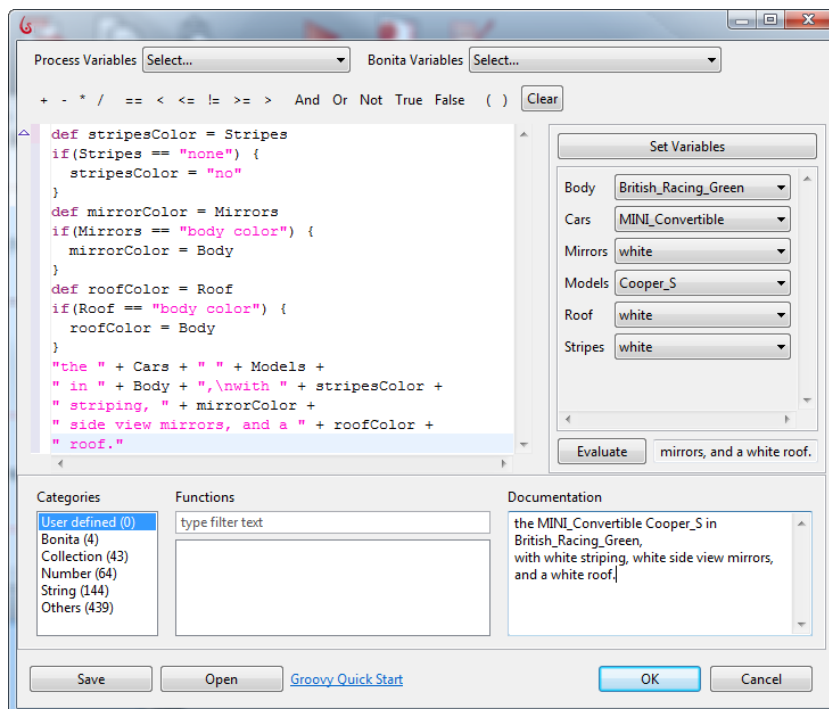


Figure 39. Groovy editor

If you need to apply operations to the output of a Connector, for example, before it is stored as the value of a variable, you can use Groovy to create the expression.

The Groovy editor presents a field for entering an expression, surrounded by tools and shortcuts. For example, all available user-defined data variables are offered in a drop-down menu in the list at the top.

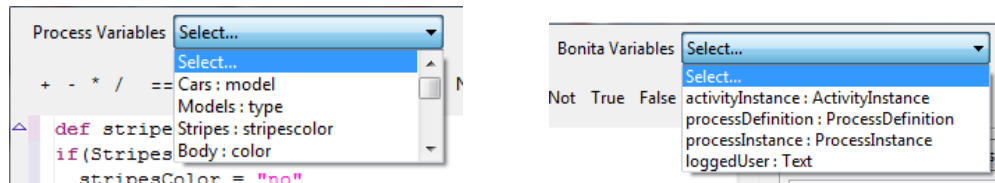


Figure 40. Select defined variables to use in Groovy expressions

A selection of common operations is also offered across the top.

A set of Groovy expressions is available in Categories; click on the script highlighted in Functions to load it into the editor.

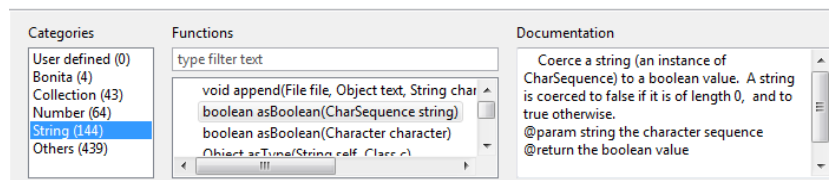


Figure 41. Find predefined Groovy functions

When the expression is valid, use **Set Variables** to enter test data, and click **Evaluate** to see the test data presented in the resulting expression.

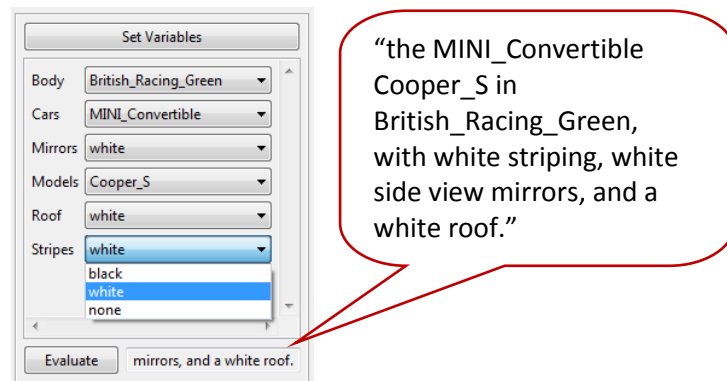


Figure 42. Evaluate to see result of variables in Groovy expression

2.4.3 Promote a local variable to become a global variable

A variable defined locally in a Step can be elevated to the Process level. Select the Step variable and **Promote**.

2.4.4 Use Java Objects (*.jar) as Variables

Add a common Java object: go to **Details -> Data -> Add**. Select **Java object** and Browse. Select the *.jar you want to define as a variable – use the string filter to bring up lists of available Java objects.

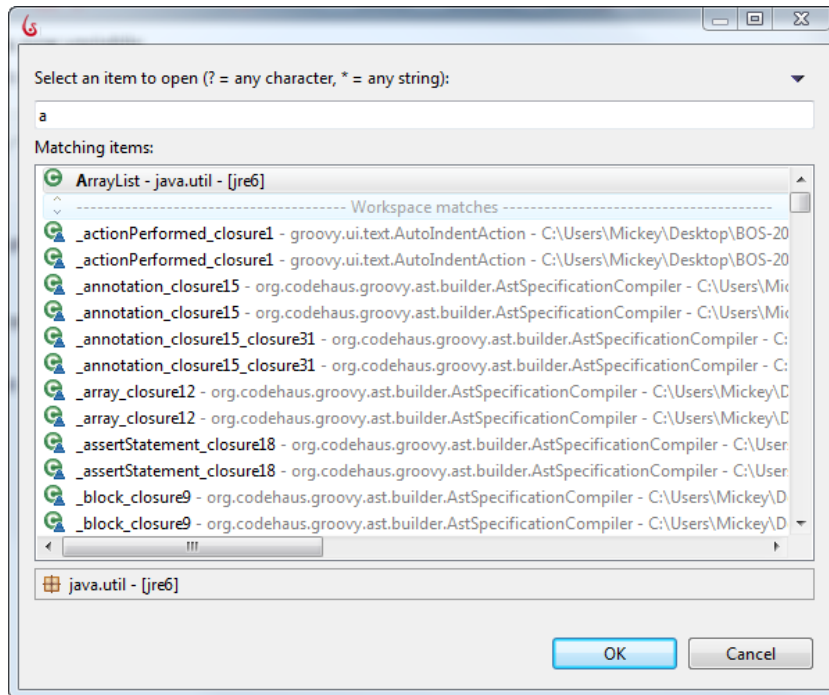


Figure 43. Browse to find Java Objects to define a Data type

If you want to further associate this with an expression, scroll through the **Default** value list to **Edit expression** and bring up the Groovy editor to create one.

To add your own complex data to your Process, first import your *.jar file(s). Go to the Menu bar and select **Extensions -> Add/Remove** and select **Add jar**.

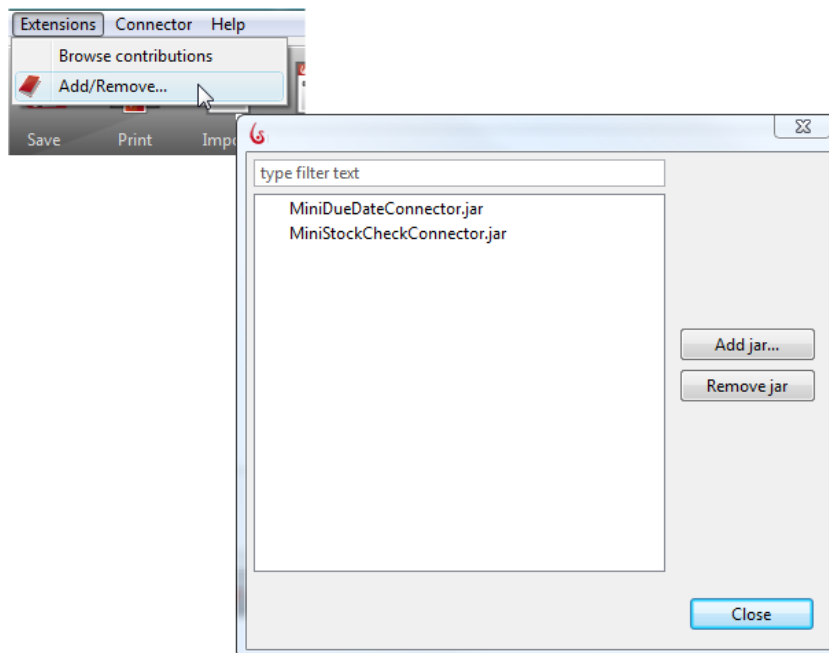


Figure 44. Add *.jar files from Menu Bar

Or, go to **Details -> Dependencies -> Add JAR**.

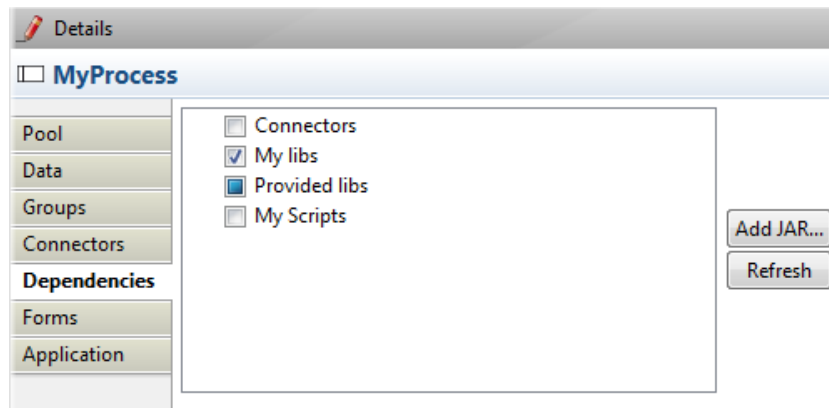


Figure 45. Add *.jar files from within Dependencies

Browse to where your files are and upload them.

To use the imported *.jar as a variable, click on the Process, Pool, or Step and go to **Details -> Data -> Add**. Select **Java object** and Browse. Your imported *.jar(s) will appear in the list of available java files.

2.5 How to call a subprocess

Any Process can be called by another Process via a Subprocess Step.

You can create a Process that calls one or more **Subprocesses** as it is run. To create a Subprocess Step: click the tools icon to the bottom left to change to a Subprocess, or Go to **Step -> Details -> General -> Activity type** and select **Subprocess**.

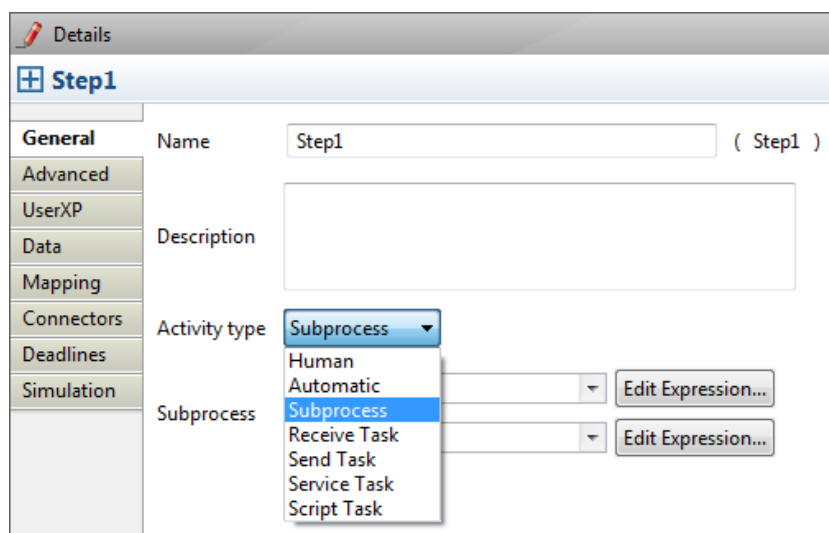
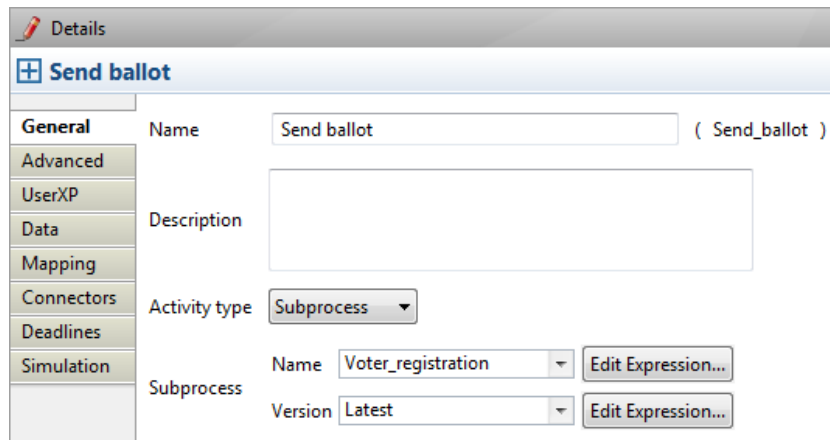


Figure 46. Create a subprocess Step

Subprocesses are built in the same way as Processes. **Open** a New Process whiteboard, create your Subprocess, and save it. Or, create a new Pool in the same Process Diagram and create a subprocess there.

When the Subprocess is completed, go to the parent Process. Select the Step that represents the Subprocess.

Go to **Step -> Details -> General -> Subprocess.**



The screenshot shows the 'Details' window for a process named 'Send ballot'. The 'General' tab is selected. The 'Name' field contains 'Send ballot' with '(Send_ballot)' next to it. The 'Description' field is empty. The 'Activity type' is set to 'Subprocess'. Under the 'Subprocess' section, the 'Name' is 'Voter_registration' and the 'Version' is 'Latest'. Both have 'Edit Expression...' buttons next to them.

Figure 47. Associate the Subprocess with its parent Process

- Browse the Processes in your Workspace and select the Subprocess you've created.
- Save your Process. The parent Process will now run with the Subprocess.

You can double-click on a Subprocess Step to bring up the linked Subprocess.

2.5.1 Map data between parent process and subprocess

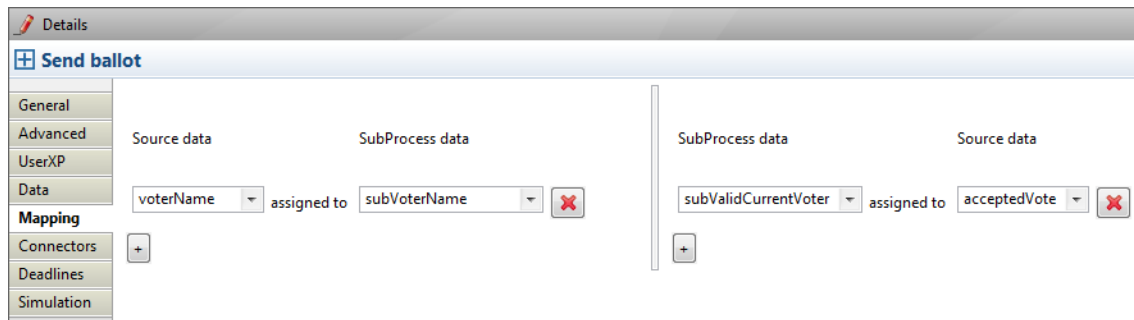
Subprocess data mapping allows you to transfer data and attachments from the parent Process directly to a variable in the Subprocess. You can use this function to:

- hand off data from the parent to the subprocess
- hand off data from the subprocess to the parent

To map data between the parent and subprocess, first identify the global variables to be linked (mapped) in both the parent and the Subprocesses (see [How to define Data variables](#)).

In the parent process, go to the Subprocess Step. Be sure that the Subprocess has been identified in **General**.

Select **Mapping**.



The screenshot shows the 'Details' window for the 'Send ballot' process, with the 'Mapping' tab selected. It displays two mapping tables. The first table maps 'Source data' to 'SubProcess data', showing 'voterName' assigned to 'subVoterName'. The second table maps 'SubProcess data' to 'Source data', showing 'subValidCurrentVoter' assigned to 'acceptedVote'. Both mappings have a red 'X' icon to the right, indicating they are active or being edited.

Figure 48. Map data between Process and Subprocess

- In the left-hand column select the **Source** data (variable defined in the parent Process or variable defined at this Step) to map to the **Subprocess** data (variable defined in subprocess).
- In the right-hand column select the **Subprocess** data (variable defined in the subprocess or variable defined at this Step) to map to the source data (variable defined in parent Process).
- You can select **Source** data from the drop-down list. Enter the subprocess variable name(s) directly.
- Delete empty fields by clicking on the red X.

To see an example of how to define a Subprocess, see the video [How to Call a Subprocess with Bonita Open Solution](#).

2.6 How to Loop a Step

Looping allows the process to execute an activity or subprocess repeatedly, either until a condition (defined as a Groovy expression) has been met, or until a specified number of loops has completed. The loop has either a pre-test or post-test condition associated with it; this is evaluated either at the beginning or at end of the loop to determine whether it should continue.

Click on the Step to be looped and go to **Details -> Advanced**. Select the option **is Looped**.

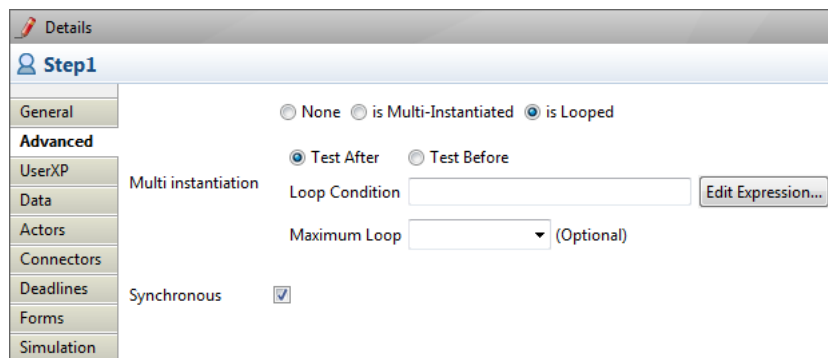


Figure 49. Loop a Step

Select the test condition “test after” or “test before.”

Define the Loop condition using a Groovy expression.

You can manually limit the number of loops performed by entering a number in **Maximum Loop**.



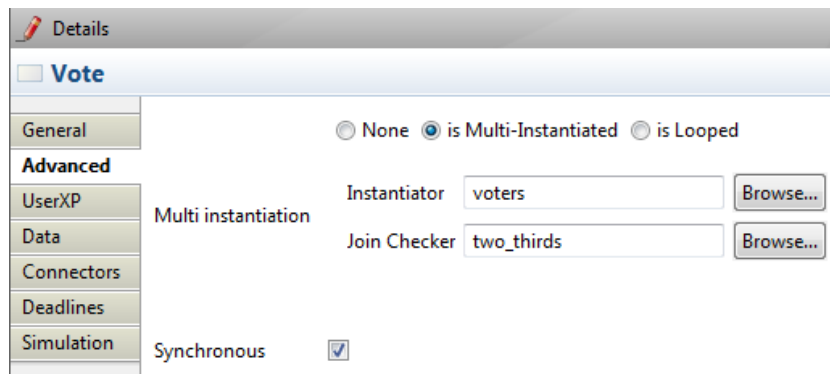
Figure 50. A looped Step

2.7 How to execute multiple instances of the same Step (multi-instantiation)

Multi instantiation (multiple instances) executes the same Step multiple times in parallel.

It requires an **Instantiator** and a **Join Checker** which are both defined in the same way as Connectors.

Click on the Step to be multi instantiated and go to **Details -> Advanced**.



- Select **is Multi-Instantiated**

2.7.1 Configure an Instantiator

An instantiator allows you to specify variable values for data as needed for each instance you want to create for a multi-instantiated Step.

For each instance that will be created for a multi-instantiated Step, the instantiator will be called to initialize its data.

To Multi instantiate a Step, *Browse* to select the **Instantiator** to be applied in the multiple instances.

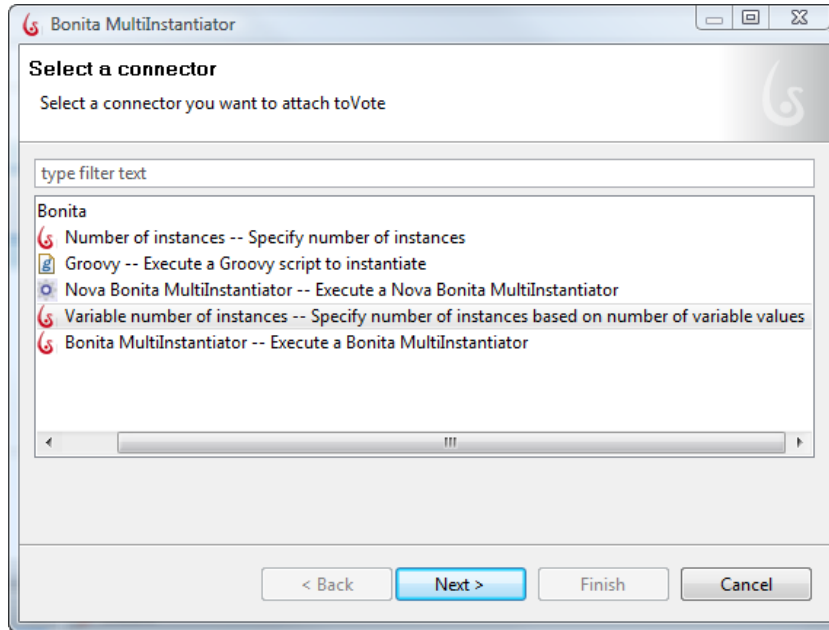


Figure 51. Define an Instantiator

As the Instantiator is like a Connector, it needs to be named and configured according to the type of multiple instances you want to generate. For example, for **Variable number of instances** as shown above, a set of values must be defined.

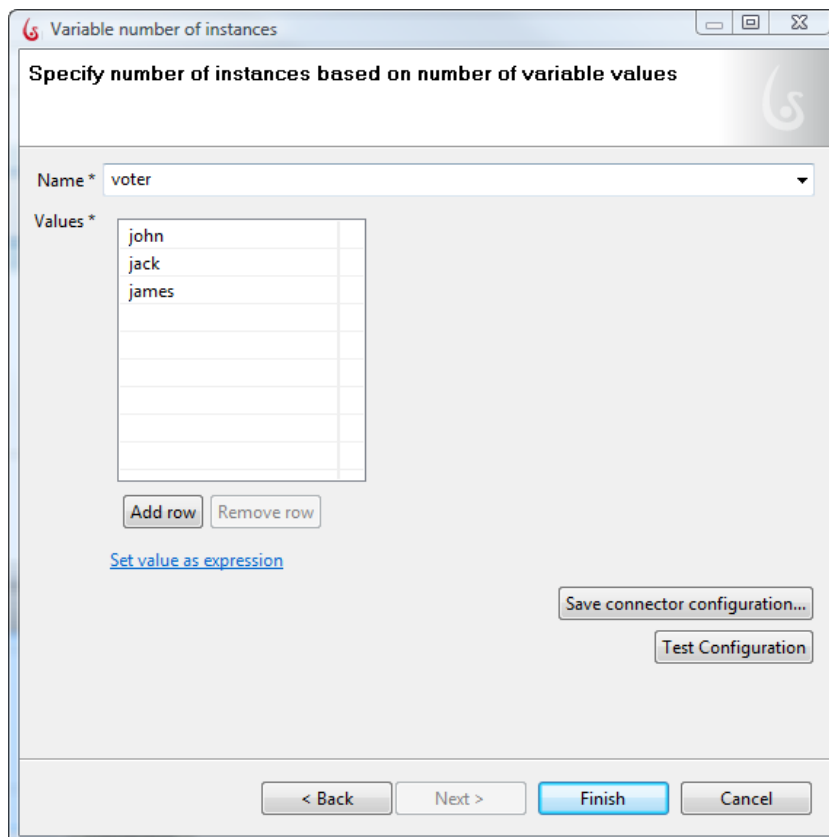


Figure 52. Configure Instantiator

In the above example, the same task will be instantiated three times: the first instance will set the voter task with John as the “Voter.” The second “Voter” will be Jack, and so on.

2.7.2 Configure a Join Checker

A multi-Instantiated Step will generate multiple parallel paths that need to be joined before the Process continues. To define the conditions of the join, configure the Join Checker.

Browse to select the **Join Checker** that will evaluate when to pass the Process along.

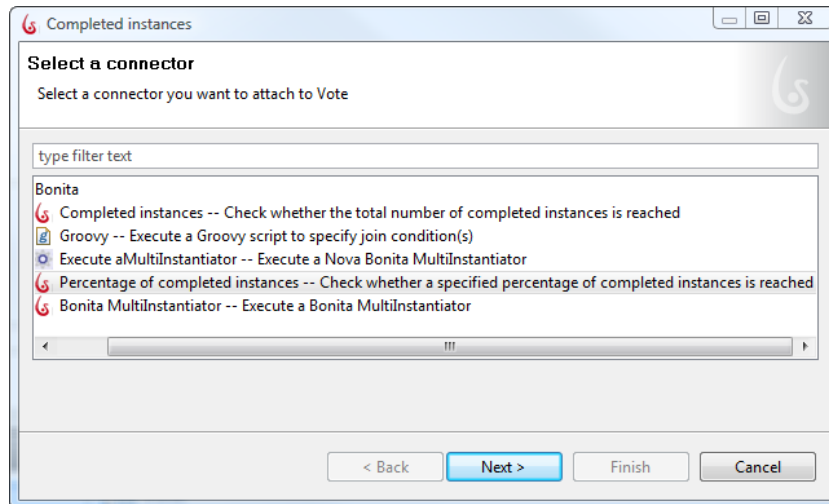


Figure 53. Define a Join Checker

As the Join Checker is a Connector, it needs to be named and configured according to the type of join you want to evaluate. For example, for **Percentage of completed instances** as shown above, a percentage of inputs received must be defined.

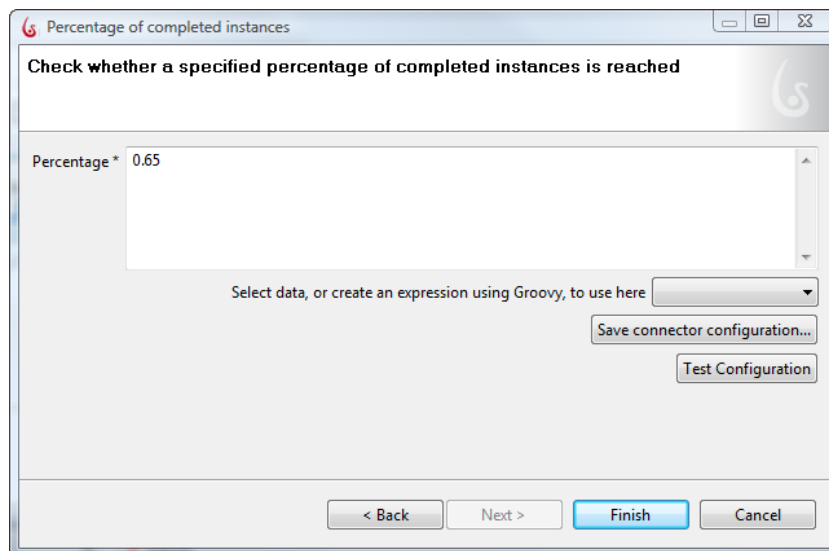


Figure 54. Configure the Join Checker

In the above example, when 65% of the instantiated Steps are completed, the Process continues.

2.7.3 Define a new Multi-Instantiation

2.8 How to set Timers

Timers are used to pause a Process until the specified duration or date has passed. A Timer can be used to trigger a Connector, or it can simply allow the Process to continue when the specified duration or date has elapsed.

From the Palette, drag the **Timer** icon or **Event** icon onto the Whiteboard, or use the Context Palette from a Step to create an Event. Then choose the Timer icon to create a Timer.

There are three types of Timers:

- Start Timer (green)
- Intermediate Timer (blue)
- Boundary Timer (attached to a Step)

2.8.1 Create a Start Timer

A Start Timer is a Start event and can be used to initiate a Process.

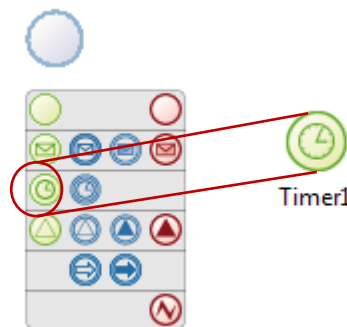


Figure 56. Create a Start Timer

To define a Start Timer, go to **Timer -> Details**:

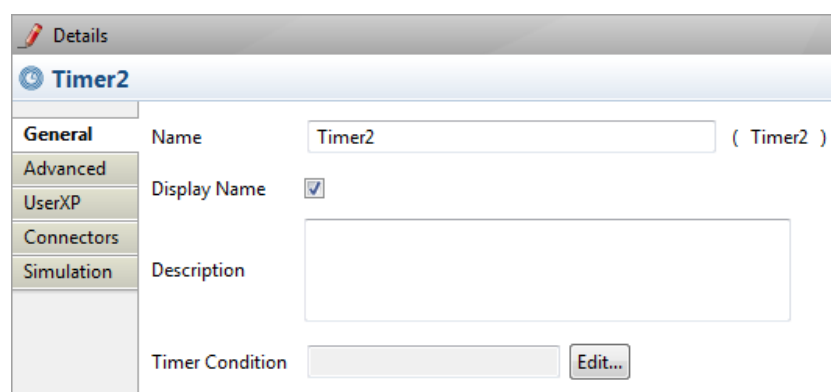


Figure 57. Set a Start Timer

- Enter **Name** and **Description**. You can choose to display the Timer name on the Process Diagram.
- **Timer Condition**: select **Edit** and then **Duration, Date** or **Variable**
- **Advanced**: untick the synchronous checkbox if you want to associate the Timer Event with a Transaction

- **UserXP:** apply a [dynamic title](#), and/or [description](#) for the Timer as it will appear in Bonita User Experience.
- **Connectors:** you can assign a Connector to launch when Timer completes. See [How to Configure Connectors](#)
- **Simulation:** define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

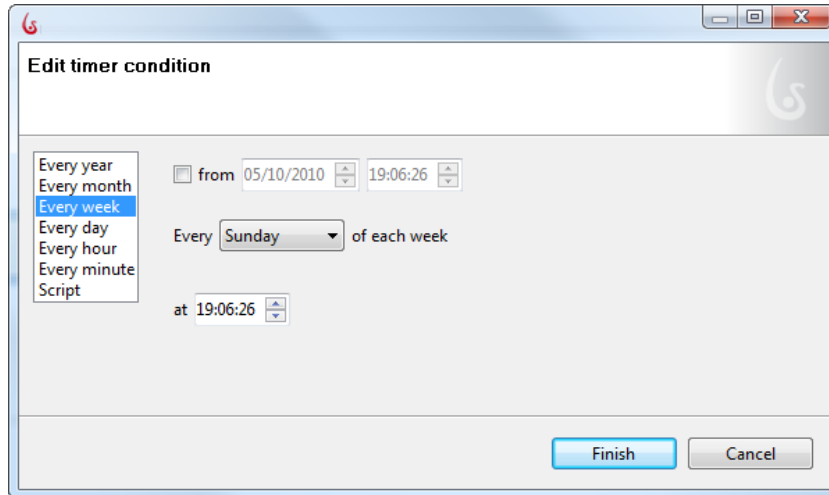


Figure 58. Configure a Start Timer

2.8.1 Create an Intermediate Timer

An Intermediate Timer can only be inserted into a Process after the Start.

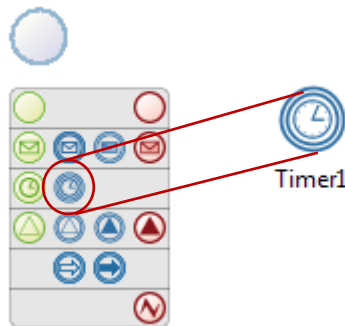


Figure 59. Create an Intermediate Timer

To define a Start Timer condition, go to **Timer -> Details -> General**. The other parameters for an Intermediate timer are the same as for a Start Timer.

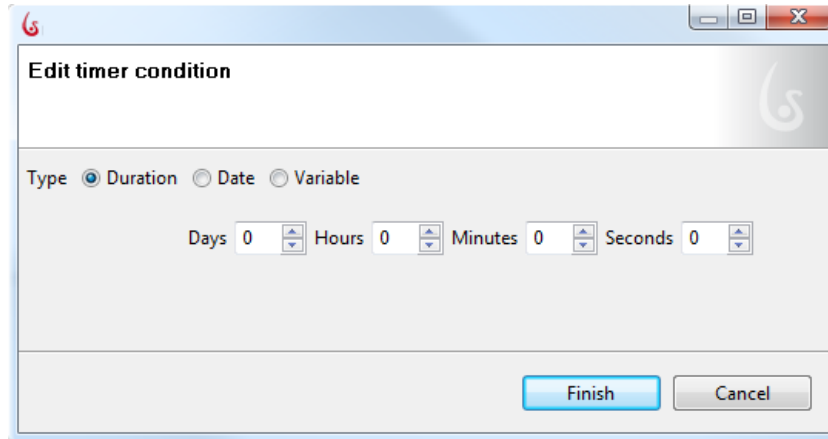


Figure 60. Configure an Intermediate Timer

- **Duration:** select the length of time to elapse
- **Date:** Select the date and time for the Timer to terminate
- **Variable:** Select a predefined variable that will return a date or an integer (See [How to define a Data variable](#))

2.8.1 Create a Boundary Timer

A Boundary Timer is attached to a Step, and will pause the execution of the activity according to the conditions defined.

From the Palette, drag the **Timer** icon directly onto the Step, or use the Context Palette from a Step to create a Boundary Event. Then choose the Timer icon to create a Boundary Timer.

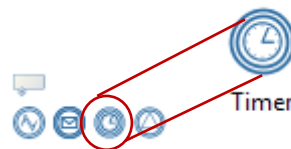


Figure 61. Choose a Boundary Timer

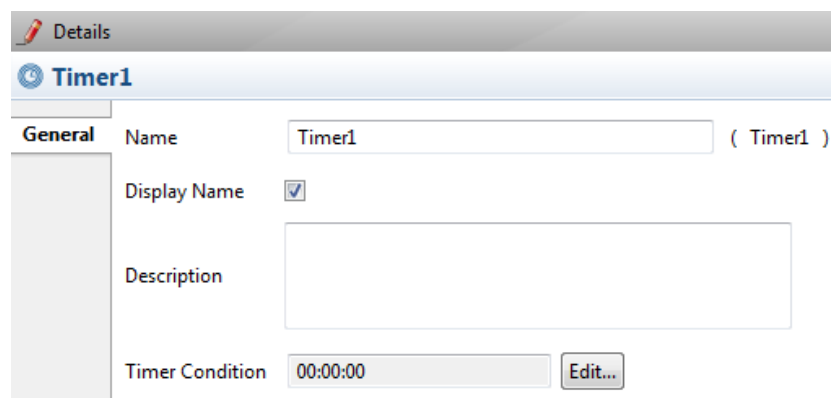


Figure 62. Set a Boundary Timer

- Enter **Name** and **Description**. You can choose to display the Timer name on the Process Diagram.
- **Timer Condition:** select **Edit** and then **Duration, Date** or **Variable**

Configure a Boundary Timer in the same way as for an [Intermediate Timer](#).

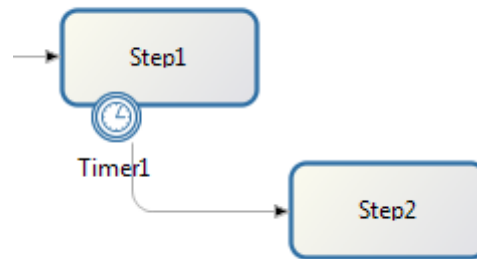


Figure 63. A Boundary Timer requires an alternate path for Process to take if Step does not complete

2.9 How to Send and Receive Intermediate Links within a Pool

Link events are used to create point-to-point connections within the same Pool. They can be used as connectors to link two different parts of the same Process.

For an example of how to use messaging, see the video [Using Links in Process Diagrams](#).

Links are connected from the Throw Link. Catch Links will automatically show the connections specified in the corresponding Throw Links you've identified.

From the Palette, drag the **Link** icon or **Event** icon onto the Whiteboard, or use the Context Palette from a Step to create an Event. Then choose the appropriate Link icon.

2.9.1 Create a Throw Link

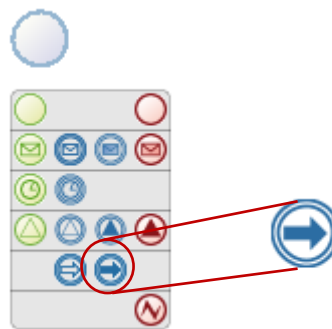
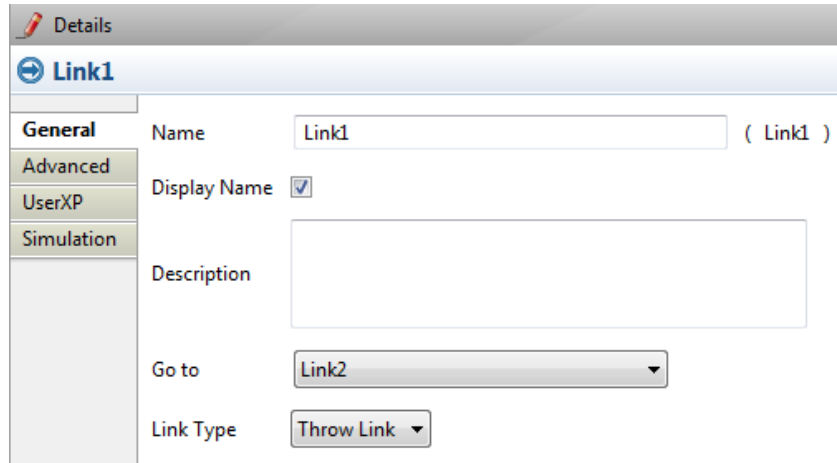


Figure 64. Select a Throw Link

Go to **Link** -> **Details**:



The screenshot shows the 'Details' window for 'Link1'. The 'General' tab is active. The 'Name' field contains 'Link1'. The 'Display Name' checkbox is checked. The 'Description' field is empty. The 'Go to' dropdown menu is set to 'Link2'. The 'Link Type' dropdown menu is set to 'Throw Link'.

Figure 65. Create a Throw Link

- Enter message **Name** and **Description**
- **Go to**: select the name created for the corresponding Catch Link
- **Link type**: select **Throw Link**

2.9.2 Create a Catch Link

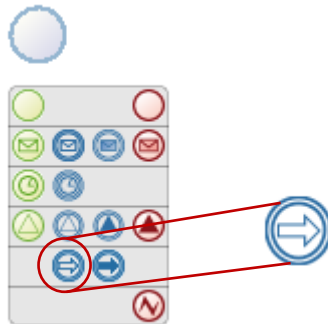
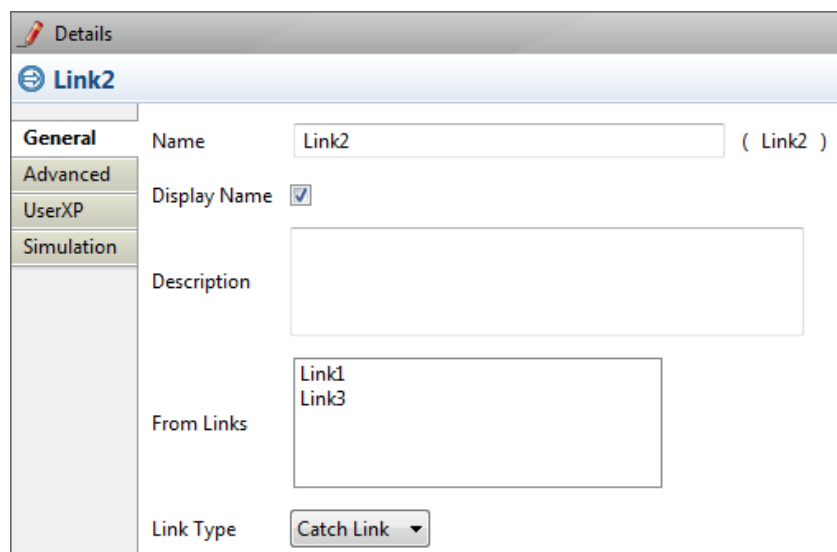


Figure 66. Select a Catch Link

Go to Link -> Details:



The screenshot shows the 'Details' window for 'Link2'. The 'General' tab is active. The 'Name' field contains 'Link2'. The 'Display Name' checkbox is checked. The 'Description' field is empty. The 'From Links' field contains 'Link1' and 'Link3'. The 'Link Type' dropdown menu is set to 'Catch Link'.

Figure 67. Create a Catch Link

- Enter message **Name** and **Description**
- **From Links**: the corresponding Throw Links you have already defined and linked here will show
- **Link type**: select **Catch Link**

2.10 How to send and receive messages across Pools

For examples of how to use messaging, see the videos [How to Send a Message Between Diagrams](#) and [How to Send and Receive a Message Between Processes with Bonita Open Solution](#).

Messages carry information from one Pool to another.

- **Throw Message**: sends a message
- **Catch Message**: receives a message
- **Start Message**: catches a message and starts a Process
- **End Message**: ends the Process and sends a message
- **Boundary Message**: a Catch Message which receives a message and starts the Step to which it is attached

From the Palette, drag the **Message** icon or **Event** icon onto the Whiteboard, or use the Context Palette from a Step to create an Event. Then choose the Message icon.

2.10.1 Create a Throw Message

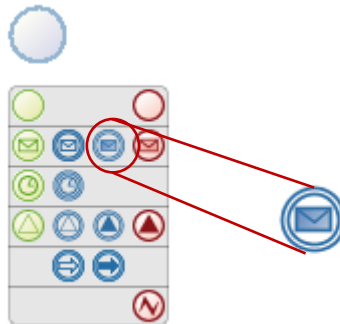
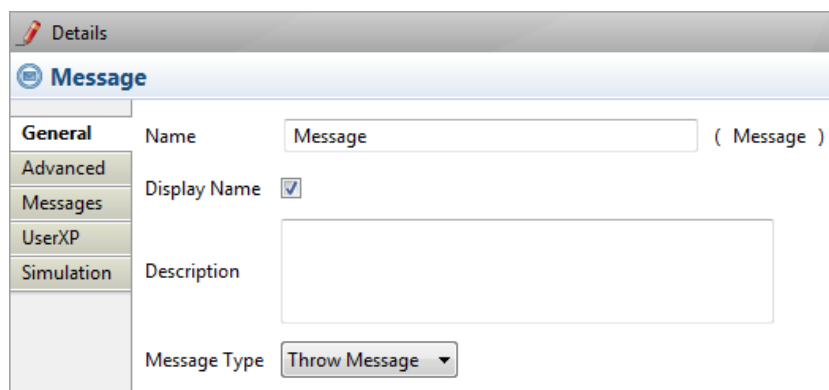


Figure 68. Select a Throw Message

Or go to **Message -> Details**:



The screenshot shows the 'Message Details' configuration window. It has a sidebar with tabs for 'General', 'Advanced', 'Messages', 'UserXP', and 'Simulation'. The 'General' tab is selected. The 'Name' field contains 'Message' with '(Message)' next to it. The 'Display Name' checkbox is checked. The 'Description' field is empty. The 'Message Type' dropdown menu is set to 'Throw Message'.

Figure 69. Create a Throw Message

- Enter **Name** and **Description**. You can choose to display the Message name on the Process Diagram.
- **Message Type**: select **Throw Message**
- **Advanced**: untick the synchronous checkbox if you want to associate the Message Event with a Transaction
- **UserXP**: apply a [dynamic title](#) and/or [description](#) for the Message as it will appear in Bonita User Experience.
- **Simulation**: define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

Go to **Messages** -> **Add** to create the Message content:

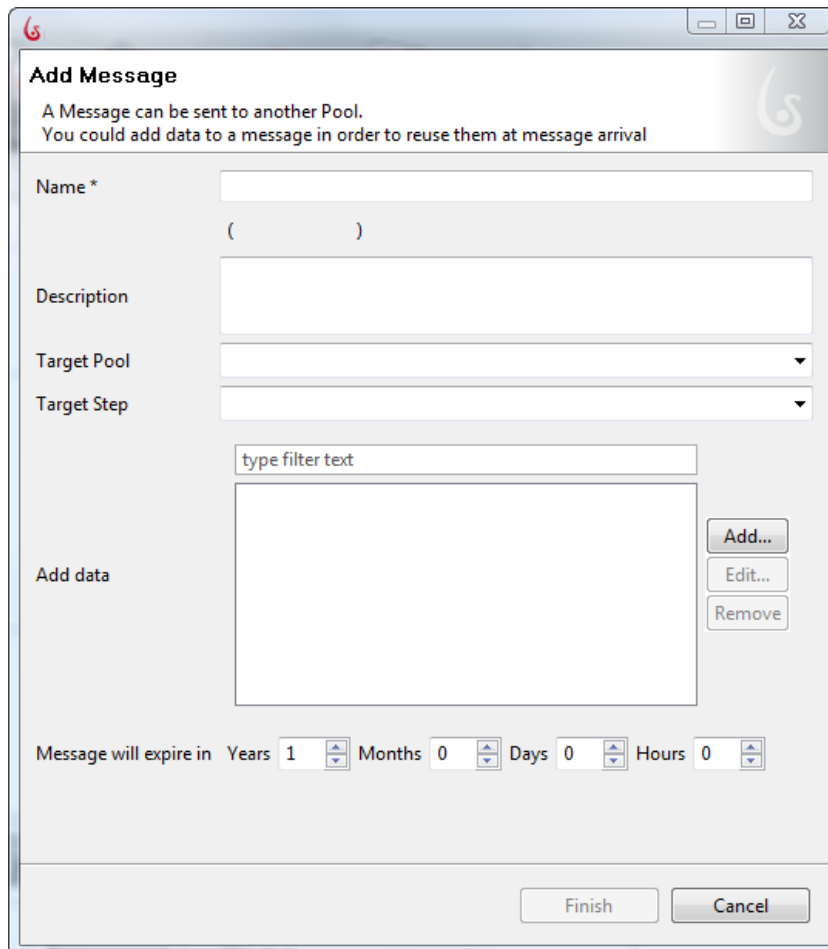


Figure 70. Add Throw Message content

- **Name** and **description**
- **Target Pool**: select the Pool you want to Catch the Message
- **Target Step**: select the Message event you want to Catch the Message
- **Add data**: here you can define the information to be carried in the Message, including data or text
- **Message will expire in**: you can specify a duration for this message to remain active

The Target Pool and Step are optional. However, if you do not specify these, the message will be broadcast to all Pools and all Steps. This can affect the performance of your system.

To define the message contents, select **Add** in **Add data**.

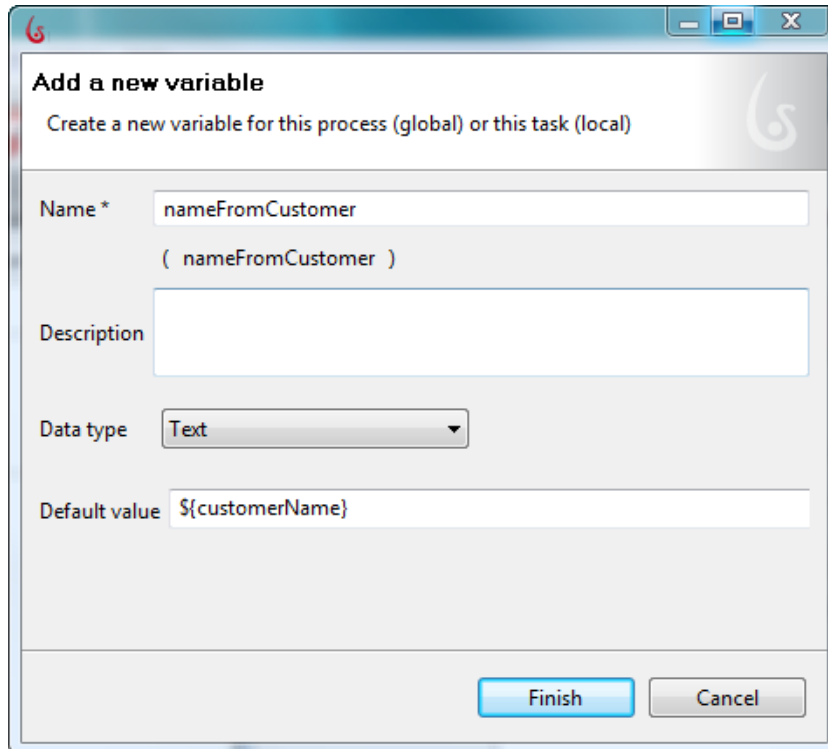


Figure 71. Define each data value to be carried by Message

Here you can define a variable to take a value from the Process of origin. Click **Finish**, and continue to add one variable for each datum you want to send.

2.10.2 Create a Catch Message

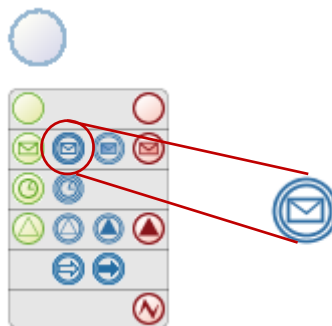
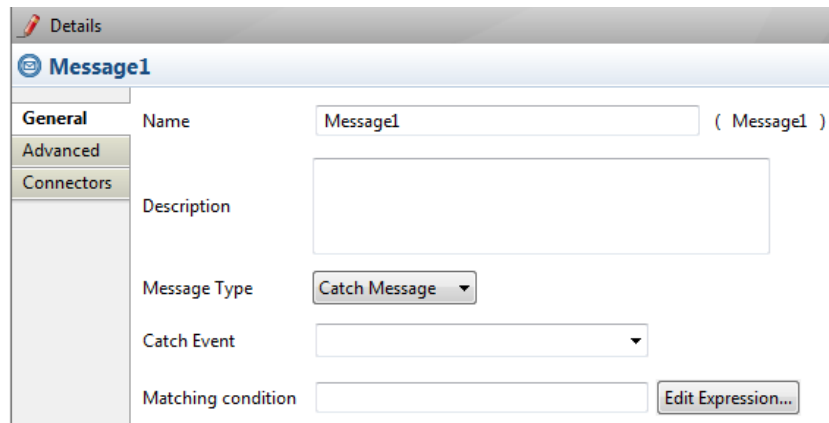


Figure 72. Select a Catch Message

Go to **Message -> Details:**



The screenshot shows the 'Details' window for a message named 'Message1'. The 'General' tab is selected. The 'Name' field contains 'Message1'. The 'Description' field is empty. The 'Message Type' dropdown is set to 'Catch Message'. The 'Catch Event' dropdown is empty. The 'Matching condition' field is empty, with an 'Edit Expression...' button next to it. The left sidebar shows 'General', 'Advanced', and 'Connectors' tabs.

Figure 73. Create a Catch Message

- Enter **Name** and **Description**.
- **Message Type:** select **Catch Message**
- **Matching condition:** create a Groovy expression to uniquely match data passed to the Catch event. For example, an expression `name != john` insures that if an event catches a name that is not "john," the event is completed and the process continues. If the catch event is not completed, the event continues to wait for another Message.
- **Advanced:** untick the synchronous checkbox if you want to associate the Message Event with a Transaction
- **UserXP:** apply a [dynamic title](#) and/or [description](#) for the Message as it will appear in Bonita User Experience.
- **Connectors:** you can use this to initiate a Connector when the message is received, or set a variable to receive the message contents.
- **Simulation:** define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

If you have not already set a variable to receive the Message, go to **Message -> Details -> Connectors -> Add**. Choose **Bonita -> Set Variable a Process or step variable**.

When the **Name the connector** window appears:

- Enter **Name** and **Description**
- **If connector fails:** choose action to take if Message fails to arrive
- Select **Next** to continue
- Enter the **Name** of the catch variable
- Select or enter the value to take from the received Message

2.10.3 Create an End Message

An End Message terminates a Process in a Pool and sends a message to another Pool.

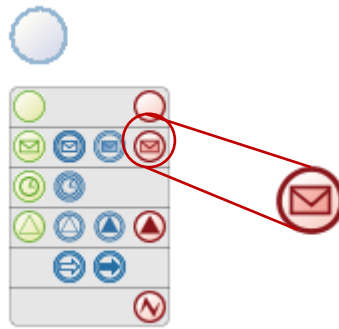


Figure 74. Select an End Message

See [Create a Throw message](#).

2.10.4 Create a Start Message

A Start Message begins a Process in a Pool on receipt of a message from another Pool.

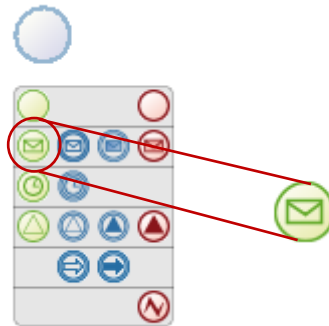


Figure 75. Select a Start Message

See [Create a Catch Message](#).

2.10.5 Create a Boundary Message

A **Boundary Message** is attached to a Step, and pauses the Step until the receipt of a message from another Pool.

From the Palette, drag the **Boundary Message** icon directly onto the Step, or use the Context Palette from a Step to create a Boundary Event. Then choose the Message icon.

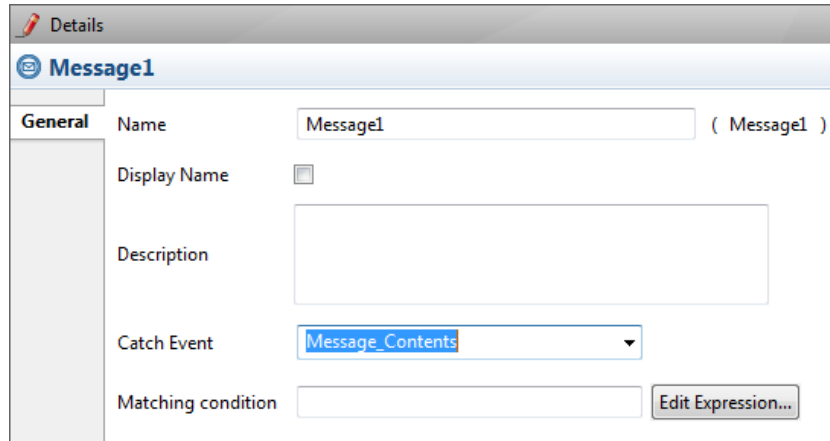


Figure 76. Create a Boundary Message

- Enter **Name** and **Description**. You can choose to display the Message name on the Process Diagram.
- **Catch event**: select the Message to catch from the sending Pool
- **Matching condition**: create a Groovy expression to uniquely match data passed to the Catch event.

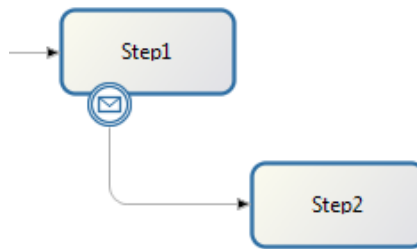


Figure 77. A Boundary Message requires an alternate path for Process to take if Step does not complete

2.11 How to define an Error Event

Error events carry information about exceptions.

- **End Error**: terminates the Process and sends an error code (message)
- **Catch Error**: this is a boundary event that catches an error code (message) and aborts the Step to which it is attached

2.11.1 Create an End Error

From the Palette, drag the **End Error** icon or **Event** icon onto the Whiteboard, or use the Context Palette from a Step to create an Event. Then choose the End Error icon.

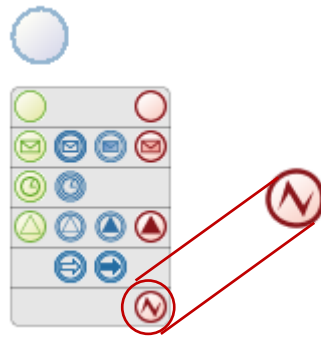


Figure 78. Select an End Error

Figure 79. Create a Throw Error

- Enter **Name** and **Description**. You can choose to display the Error name on the Process Diagram.
- **Error code**: define error code/message to be sent
- **Advanced**: untick the synchronous checkbox if you want to associate the Error Event with a Transaction
- **UserXP**: apply a [dynamic title](#) and/or [description](#) for the Error as it will appear in Bonita User Experience.
- **Simulation**: define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

2.11.2 Create a Boundary Error

A Boundary Error catches an error code (message), cancels the Step to which it is attached, and activates an exception path.

From the Palette, drag the Catch Error icon directly onto a Step, or use the Context Palette from a Step to create a Boundary Event. Then choose the Error icon.

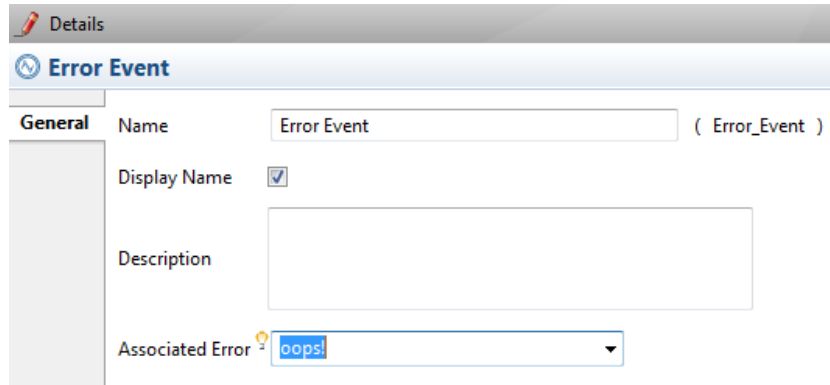


Figure 80. Create a Boundary Error

- Enter **Name** and **Description**. You can choose to display the Error name on the Process Diagram.
- **Associated error**: select the error code that will activate this event; or, if no error code is given, this error will catch any error not specified for a particular target

2.12 How to define a Signal Event

Signals broadcast or receive information from anywhere in a Process.

- **Throw Signal**: broadcasts a Signal
- **Catch Signal**: receives a Signal
- **Start Signal**: catches a Signal and starts a Process
- **End Signal**: ends the Process and sends a Signal
- **Boundary Signal**: a Catch Signal receives a Signal and starts the Step to which it is attached

From the Palette, drag the **Signal** icon or **Event** icon onto the Whiteboard, or use the Context Palette from a Step to create an Event. Then choose the Message icon.

2.12.1 Create a Throw Signal

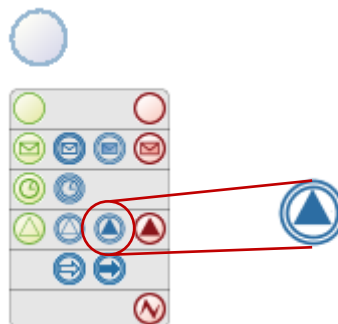


Figure 81. Select a Throw Signal

Go to **Signal -> Details**:

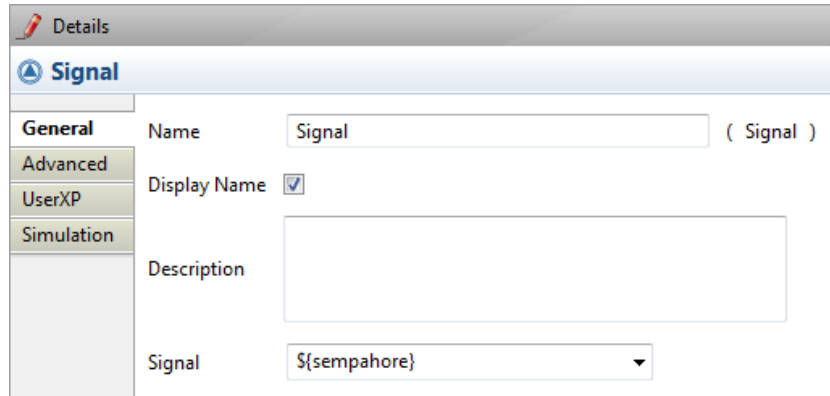


Figure 82. Create a Throw Message

- Enter **Name** and **Description**. You can choose to display the Signal name on the Process Diagram.
- **Signal**: select the signal code to be broadcast
- **Advanced**: untick the synchronous checkbox if you want to associate the Signal Event with a Transaction
- **UserXP**: apply a [dynamic title](#) and/or [description](#) for the Signal as it will appear in Bonita User Experience.
- **Simulation**: define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

2.12.2 Create a Catch Signal

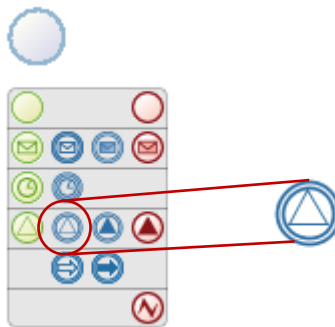


Figure 83. Select a Catch Signal

Go to **Signal** -> **Details**:

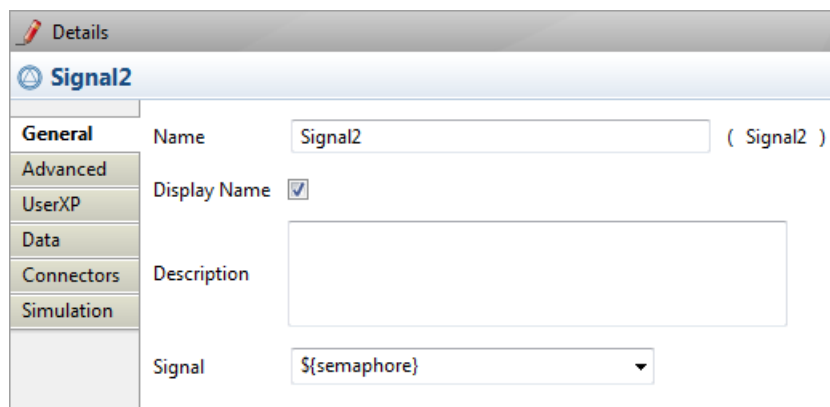


Figure 84. Create a Catch Signal

- Enter **Name** and **Description**. You can choose to display the Signal name on the Process Diagram.
- **Signal**: select the signal code to be broadcast
- **Advanced**: untick the synchronous checkbox if you want to associate the Signal Event with a Transaction
- **UserXP**: apply a [dynamic title](#) and/or [description](#) for the Signal as it will appear in Bonita User Experience.
- **Connectors**: you can use this to initiate a Connector when the signal is received
- **Simulation**: define parameters to be used when running a Simulation of the Process. See [How to Configure and Run a Simulation](#).

2.12.3 Create an End Signal

An End Signal terminates a Process and broadcasts a signal.

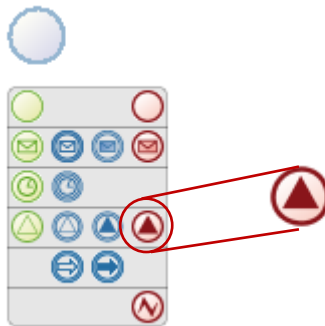


Figure 85. Select an End Signal

See [Create a Throw Signal](#).

2.12.4 Create a Start Signal

A Start Signal begins a Process on receipt of a broadcast Signal.

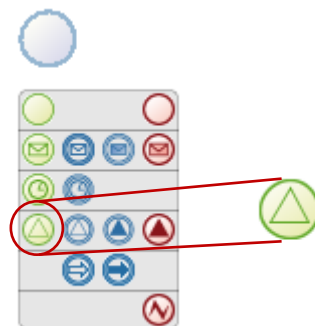


Figure 86. Select a Start Signal

See [Create a Catch Signal](#).

2.12.5 Create a Boundary Signal

A **Boundary Signal** is attached to a Step, and pauses the Step until the receipt of a broadcast Signal.

From the Palette, drag the **Boundary Signal** icon directly onto the Step, or use the Context Palette from a Step to create a Boundary Event. Then choose the Signal icon.

Figure 87. Create a Boundary Signal

- Enter **Name** and **Description**. You can choose to display the Message name on the Process Diagram.
- **Signal**: select the signal code to be received

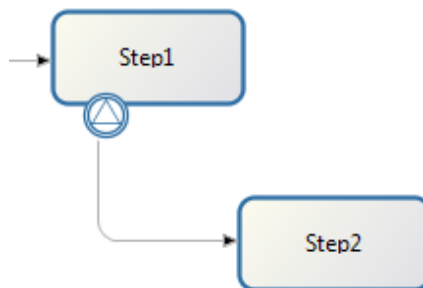


Figure 88. A Boundary Signal requires an alternate path for Process to take if step does not complete

2.13 How to define Transactions

A transaction is a group of activities that must all complete successfully before the Process can continue. A series of activities are defined as a Transaction in Bonita Open Solution if they are sequential or part of the same execution pattern and are synchronous.

Events are synchronous by default in Bonita Open Solution. To mark the end of one Transaction and identify the beginning of another, the event in the Process immediately following the Transaction should be “de-synchronized.” This signals the disconnection of this event from the Transactional group.

When the Bonita Execution Engine receives an order from a client application (Bonita User Experience or third party application) it uses the client Java thread. During the execution, if an activity is marked as Asynchronous, the engine will give the control back to the client and open a new Java thread to continue with the execution of following activities.

To indicate de-synchronization of any event, click on the Event and go to **Details -> Advanced** and un-check the selection for **Synchronous**.

2.14 How to configure Connectors

For information about how to configure Connectors included in Bonita Open Solution and to create your own Connectors, see the **BOS Connectors Reference Guide**.

2.15 How to define Categories

Categories are labels that can be applied by the Process designer to individual Processes.

To define a Category, select a Pool or Process and go to **Details -> UserXP**.

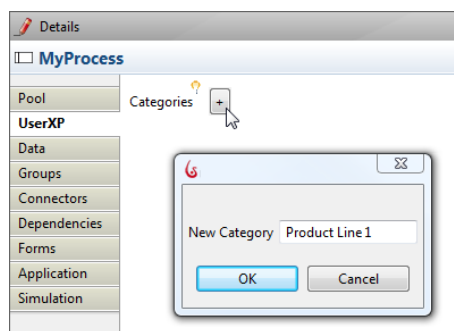



Figure 89. Define a Category for a Process

Click on the  icon to bring up the window to name the Category.

Each Category label or set of labels designed this way is applied to each case of the Process when it is run. The Category (ies) will show up in all User's User Experience inboxes whenever this Process appears.

2.16 Install (import) an extension shared in the BonitaSoft Community Contributions

To import a Contribution into Bonita Open Solution:

- **Menu Bar: Extensions -> Browse contributions.**

This will connect to the Contributions section of www.Bonitasoft.org. From here you can select a shared extension and **Install** it. (You will need to be registered as a BonitaSoft Community user and log in to the Community forum from Bonita Studio).

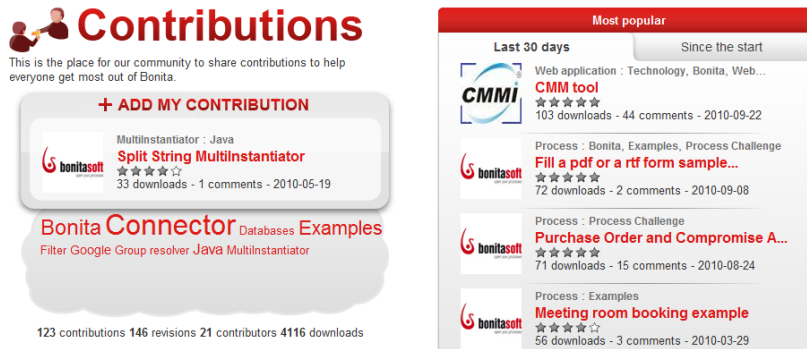


Figure 90. Select an extension from the Contributions page



Figure 91. Import an extension

To close the Contributions page and return to the Whiteboard, click the X in the upper right corner of the BonitaSoft.org window.

When you install an extension, be sure to confirm if it requires that you also install any dependencies. See [Add *.jar files](#).

2.17 How to manage Processes in Bonita Studio

2.17.1 Find Process files

Bonita Open Solution stores your Processes as *.proc files in your Workspace directory, in My Processes. This directory is located by default in **BOS-5.X\studio\workspace\local\local_My Processes**.

The management tasks for Processes are located under **Process** in the Task bar.

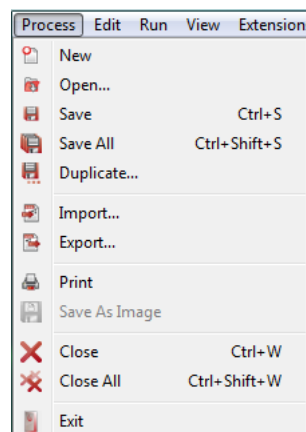


Figure 92. Process management tasks are located in the Task bar

2.17.2 Create a New Process or Open an existing Process

Start a **New** Process or **Open** a Process

- By selecting **Process -> New** or **Process -> Open** in the Menu bar;
- By clicking on the **New** or **Open** icon on the Task bar; or
- By clicking on **New** or **Open** from the Bonita Studio welcome page.

2.17.3 Save a Process / Save all Processes

Bonita Open Solution saves Processes as *.proc files.

Save a Process:

- By selecting **Process -> Save** in the Menu bar
- By clicking on the **Save** icon on the Task bar

To **Save all** Processes, select **Process -> Save All**.

2.17.4 Duplicate a Process

Create a copy of an existing Process by selecting **Process -> Duplicate** from the menu bar. You can **Name** the duplicate and give it a **Version** number.

2.17.5 Import a Process

You can **Import** processes in the following formats:

- *.proc (Bonita Open Solution),
- *.bar (Bonita Open Solution),
- *.xpd1 (Bonita 4),
- *.bpnm (BPMN 2.0), and

processdefinition.xml (jBPM3)

Select **Process -> Import** from the Menu bar, or the **Import** icon from the Task bar.

To Import a file, browse to the file location and upload.

2.17.6 Export a Process

You can **Export** a zipped package which includes:

- The Process Diagram (Process BAR)
- any custom Web applications you have created (Application WAR) - resources, forms, and templates and everything needed to render the process as defined in Bonita Studio
- the Bonita Execution Engine (Runtime) - *.ear files, configuration and resource library directories

The library directories can be exported as

- Light – this generates a lightweight WAR with its libraries available in an external folder (“lib”)

- Embedded – this generates a WAR with embedded libraries
- Client WAR – this generates a WAR ready for JEE with no “bonita-server” libraries

More information is given in the README text file included in the export.

Select **Process -> Export** from the Menu bar, or the **Export** icon from the Task bar.

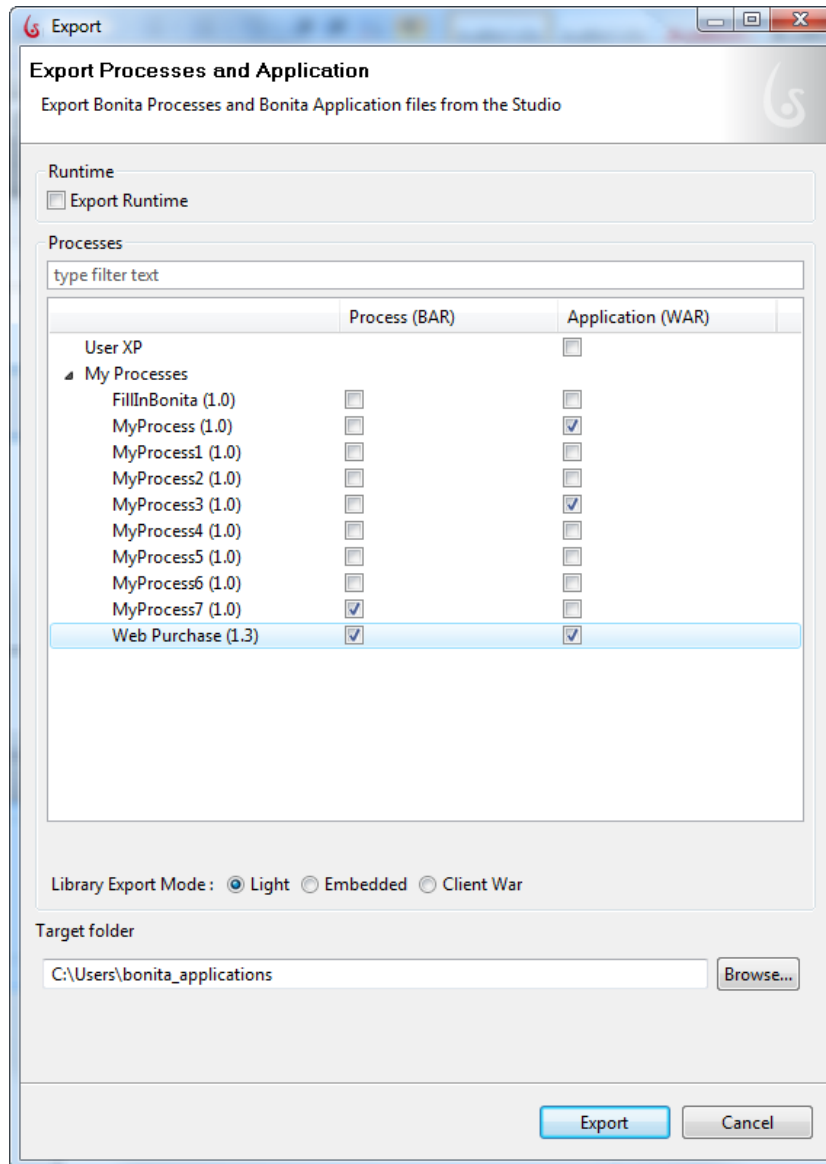


Figure 93. Export Application

This function will export all libraries for your web applications. You can choose to save them in a WEB-INF/lib folder (default) co-located with the process file, or in a common folder in the zipped archive.

Should you need **Export** Processes in a *.proc file, you will find it in **Workspace \ local\ local_MyProcesses.**)

2.17.7 Print a Process

To **Print** a hard copy of a Process Diagram, select **Process -> Print**.

2.17.8 Close a Process / Close all

To **Close** a single Process, select **Process -> Close** or click the X on its tab.

To **Close** all open Processes, select **Process -> Close All** from the Menu bar.

2.17.9 Delete a Process

Each time you open a new Process in Bonita Studio, it is automatically saved. To clear out unwanted Processes in the Existing Processes list, **Open** a Process.

When the **Open an Existing Process** window appears, select the Process you want to delete and **Delete**.

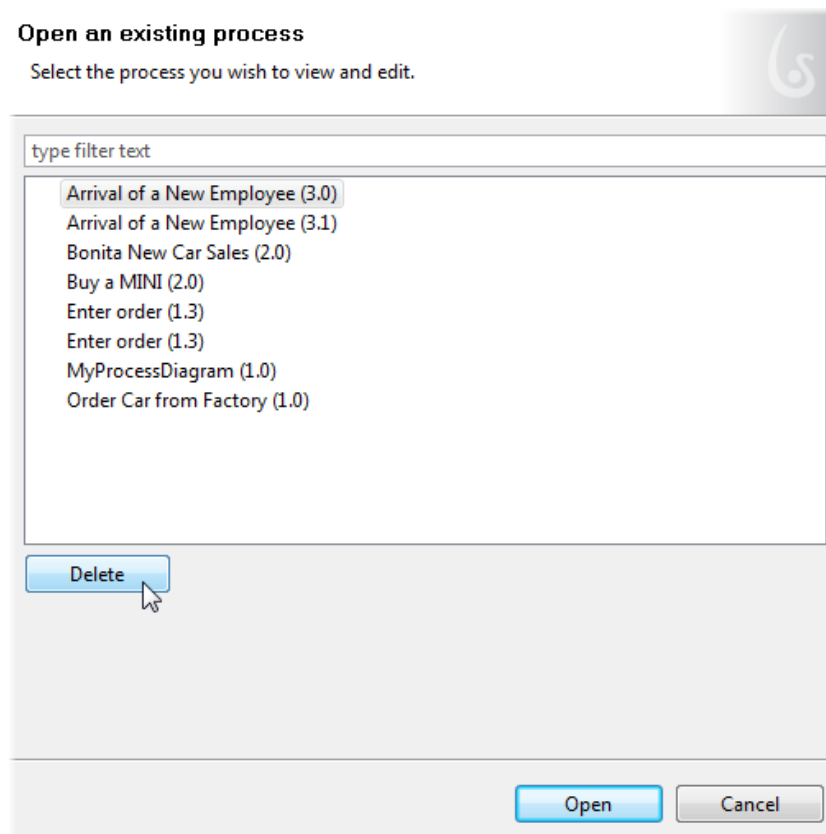


Figure 94. Delete a Process

2.17.10 Rename a Process Diagram

To change the name of a top-level Process Diagram, go to **Diagram -> Details -> General**. Here you can change the **Name**, the **Version**, and the **Description** of an opened Process.

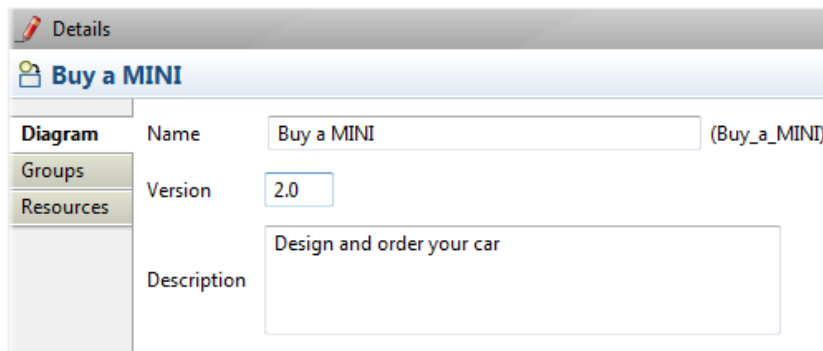


Figure 95. Rename a Process Diagram

NOTE: Be sure you are in the Details of the Process Diagram level. Changing the name of a Pool or a Lane will not change the name of the top-level Process Diagram.

2.18 How to run a Process

Running a process deploys it, that is, the process becomes “live” and cases can be implemented. For development, the Process is run in default Bonita Open Solution web applications.

Run a Process after it is built in Bonita Studio:

- Select **Run -> Run** from the Menu bar

- Click on the **Run** icon  in the Task bar

This will generate a default web application.

What happens, essentially, behind the RUN button:

- The process and resources are packaged in a *.bar file (Bonita process archive)
- A specific Web application is created
- The Bonita Execution Engine and a lightweight relational database are used by the application
- A web container is started and the *.war file deployed
- A new browser window opens and shows the initial application page

A dialog may appear to inform you that Bonita Open Solution is initializing the User Experience.

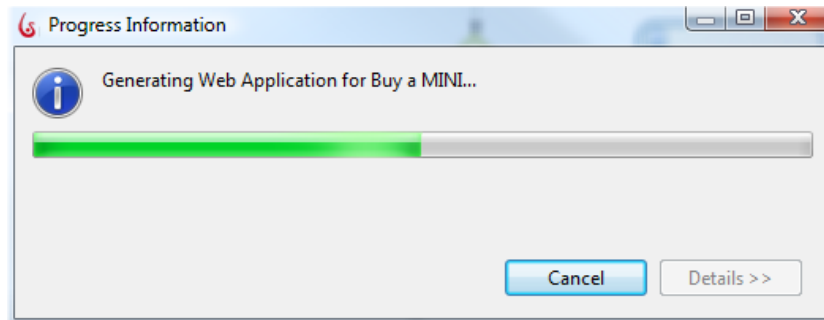


Figure 96. Bonita Open Solution initializing the User Experience

When it has completed, a form appears in the default web application, and you are automatically logged in as admin. If appropriate, enter data in its fields and click **Submit** to start the Process.

When presented with the message: “The information has been submitted. To continue, open your inbox...”:

Click on **Bonita User Experience** to open the inbox.

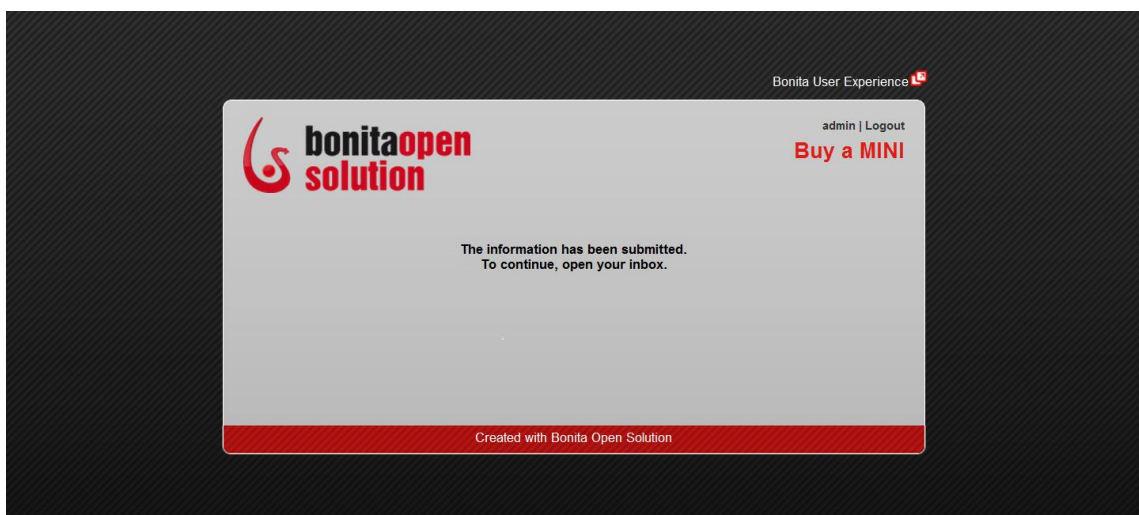


Figure 97. Open the Bonita User Experience inbox in the default web application

The Process will show up in the admin inbox. As you run this first Case, you can enter data in the application fields Step by Step.

This will complete the first Case of the Process.

When a Process has been **Run**, it can be implemented repeatedly, each time as a new Case. See [Part 4, How to Use Bonita User Experience](#).

2.19 How to configure and run a Simulation

For information about how to configure and run a Simulation in Bonita Open, see the **BOS Simulation Reference Guide**.

Part 3. How to create and customize forms for end users

3.1 Overview

You can customize the interface presented to your end users by modifying html, and/or creating Web applications and linking them to Bonita Open Solution.

By default, at each Process Step that requires data input, Bonita Open Solution automatically creates a basic, un-customized Bonita Form based on the Data Variables you have defined.

When the Process is run with no customization, a simple default Bonita Form for each Step is presented in a default Bonita Web application.

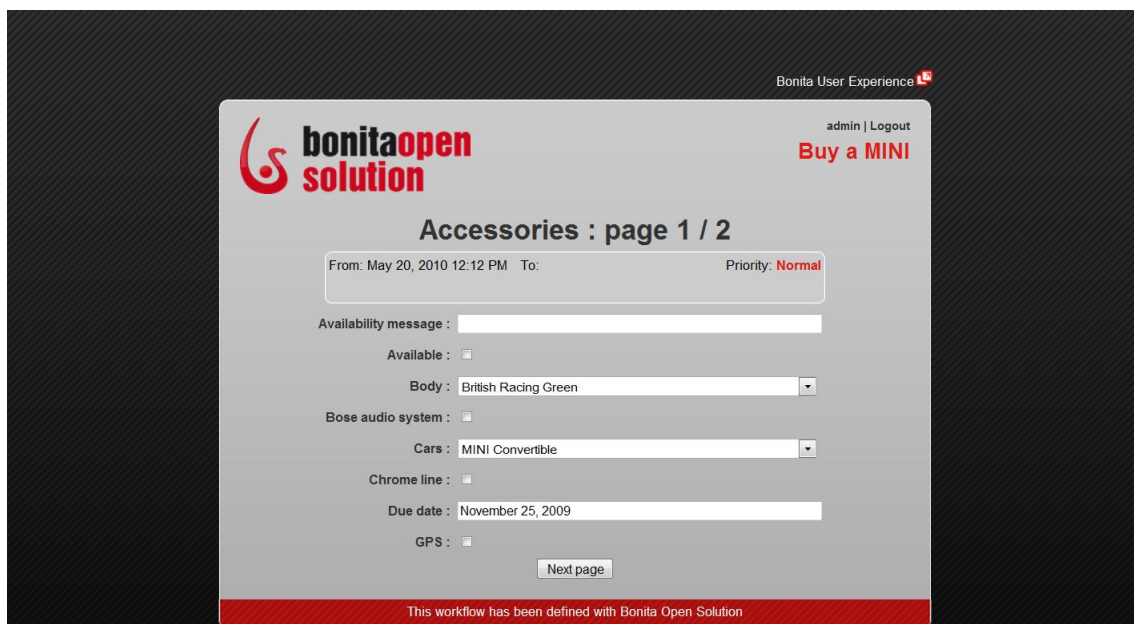


Figure 98. Step Form with no customization
(basic Bonita Open Solution form in default Web application templates)

You can customize the end user interface in several ways. For Forms, you can:

- Change the Process template (Web application) in which all Forms appear (See [Customize the Process Template](#)).
- Further customize the layout of an individual Form by modifying or changing the html template for the individual form, and/or modifying or changing the Global Page Template (See [Customize a Step level template](#)).
- Customize the fields and contents of an individual Form at the Step level (in what order the fields are presented, the field labels, field characteristics, and so on). This is done with the Form Builder in Bonita Studio. (See [Design a Step Form](#)).

You can [customize confirmation messages](#) presented to the user. There can be multiple confirmation messages in a single Process.

And - you can also customize the following user interfaces at the process level:

- [Customize Welcome page](#)
- [Customize Log in page](#)
- [Customize error messages](#)

3.2 How to create a Form for a Step

You can design 3 types of Forms:

- **Entry** allows data entry by Step Actor
- **View** shows data already entered; can not be modified
- **Overview** shows all these Forms with read-only contents in the User Experience

You can create Forms to summarize the state of the Process at the Process level in the Forms tab.

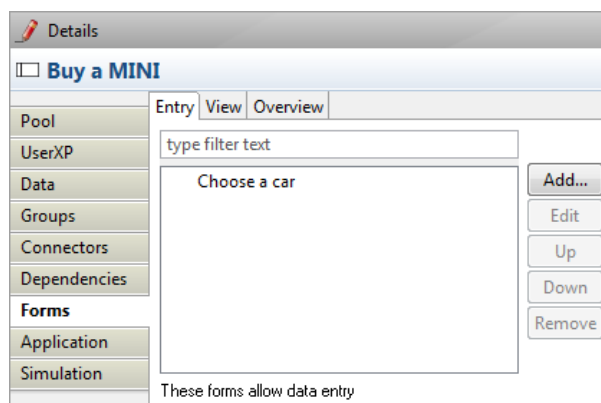


Figure 99. Three types of Forms in a Process

To design these Forms, go to select a Human Step, and go to **Details -> Forms**, and select either **Entry** or **View**.

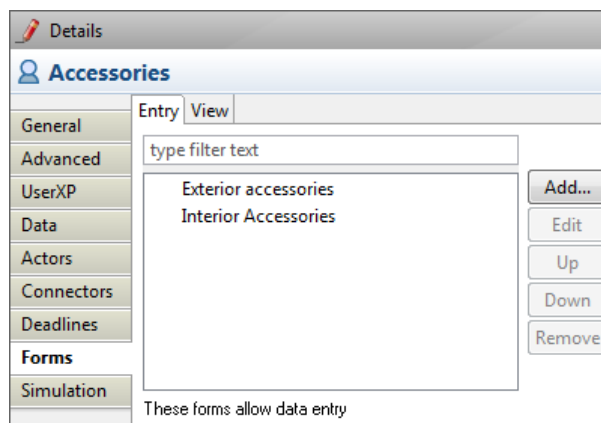


Figure 100. Create an Entry or View form for a Step

3.2.1 Design a Step Form

Forms for data entry and/or presenting information are available for Human (manual) Steps. To complete a Form, you will need to have global and local variables defined. (see [How to Define Data Variables](#)).

To create a new custom form, select a Human Step, and go to **Details -> Forms -> Add**.

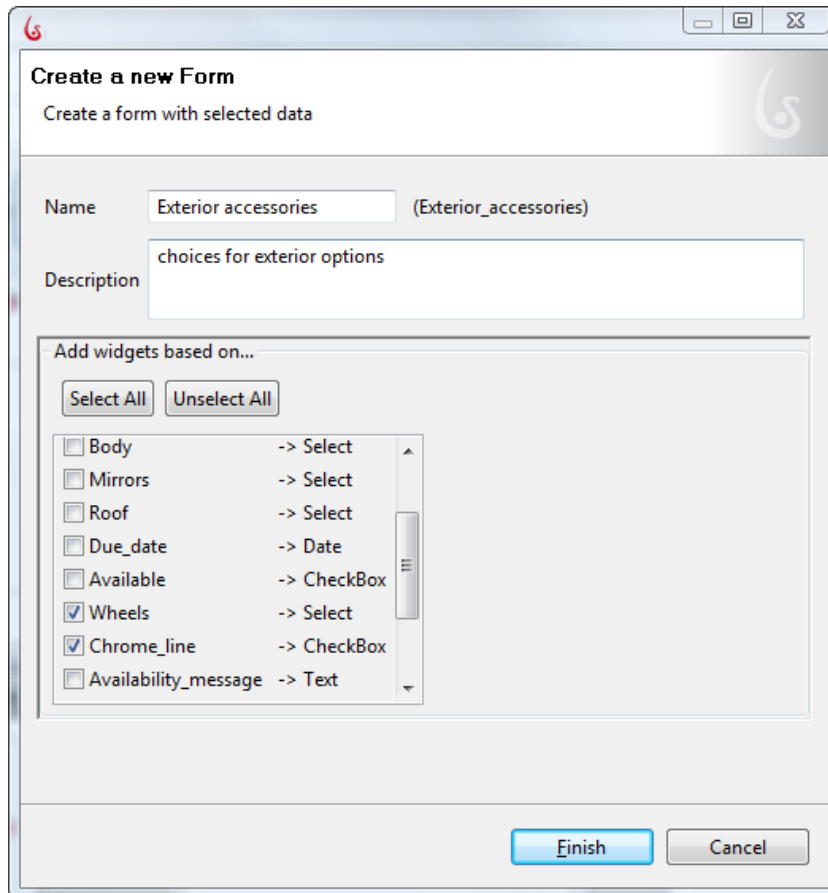


Figure 101. Create a New Form

The **Create a New Form** wizard will ask you to enter a **Name** and **Description** for the Form. In this window you'll also see an initial list of the Data variables available for this Step. You can Select or de-select variables here. You can also add and remove variables in the next steps of creating the Form.

Select some, all, or none of the variables presented and click **Finish** to continue.

The next window presented, the **Form Builder**, will allow you to manipulate data fields in the form of cells in a grid. If you have selected some or all variables, they are presented here as a series of cells in a single vertical column. Form Builder adds a **Submit** button at the bottom of the column.

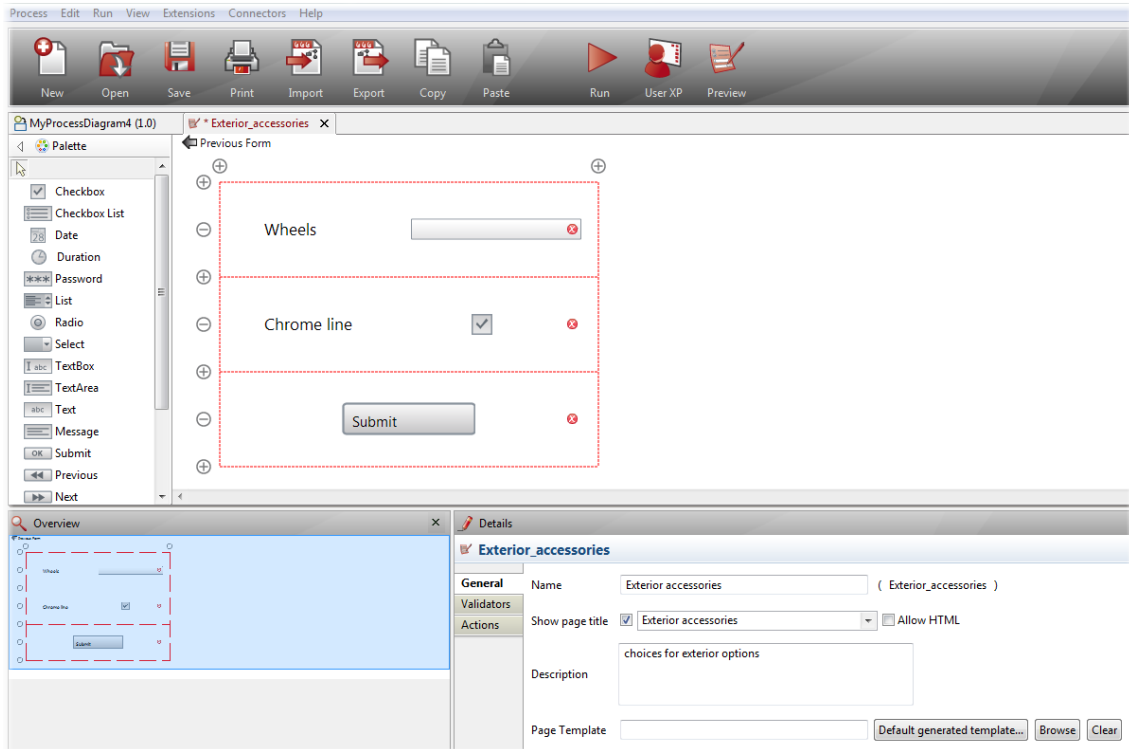


Figure 102. Begin with the Form Builder

To add data fields that are not included here, you can add cells (see below) and then click and drag the field widget from the Palette. See [Define or redefine the contents of a field in a Form.](#)

3.2.1.1. Add / remove rows of cells in a Form

Add or remove rows using the + and – icons.

- + along the left side, upper adds a row above
- + along the left side, lower adds a row below
- - along the left side, center removes the row (and its contents).

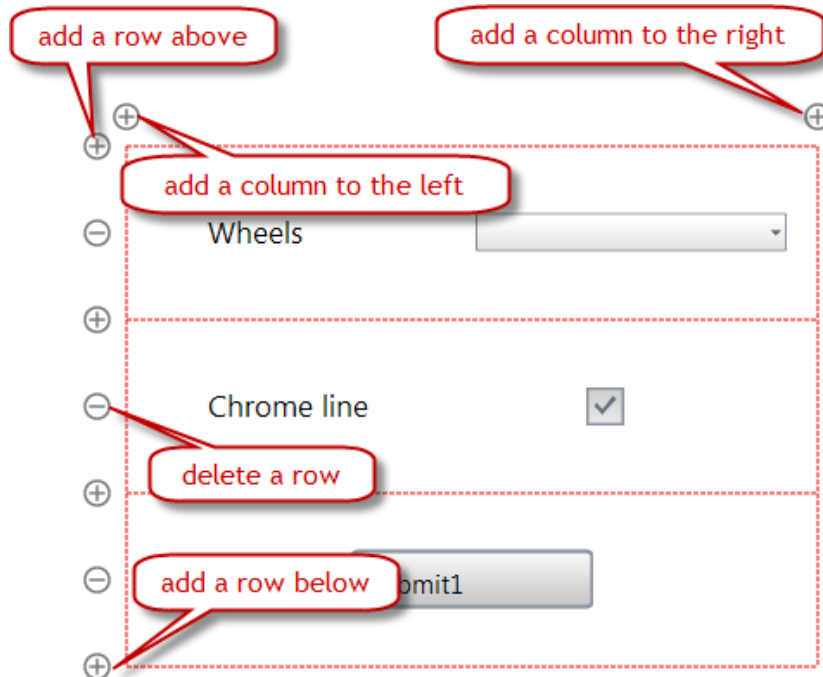


Figure 103. Add or remove rows and columns in the grid

Similarly, add or remove columns using the + and – icons

- + at the left adds a column to the right
- + at the right adds a column to the left
- - in the center removes the column (and its contents)

3.2.1.2 Move cells

Click and drag the data fields to move them around in the Form Builder.

3.2.1.3 Merge cells

To merge a data field cell across adjacent cells, click on the cell and then on the arrow in the direction you want to expand it.

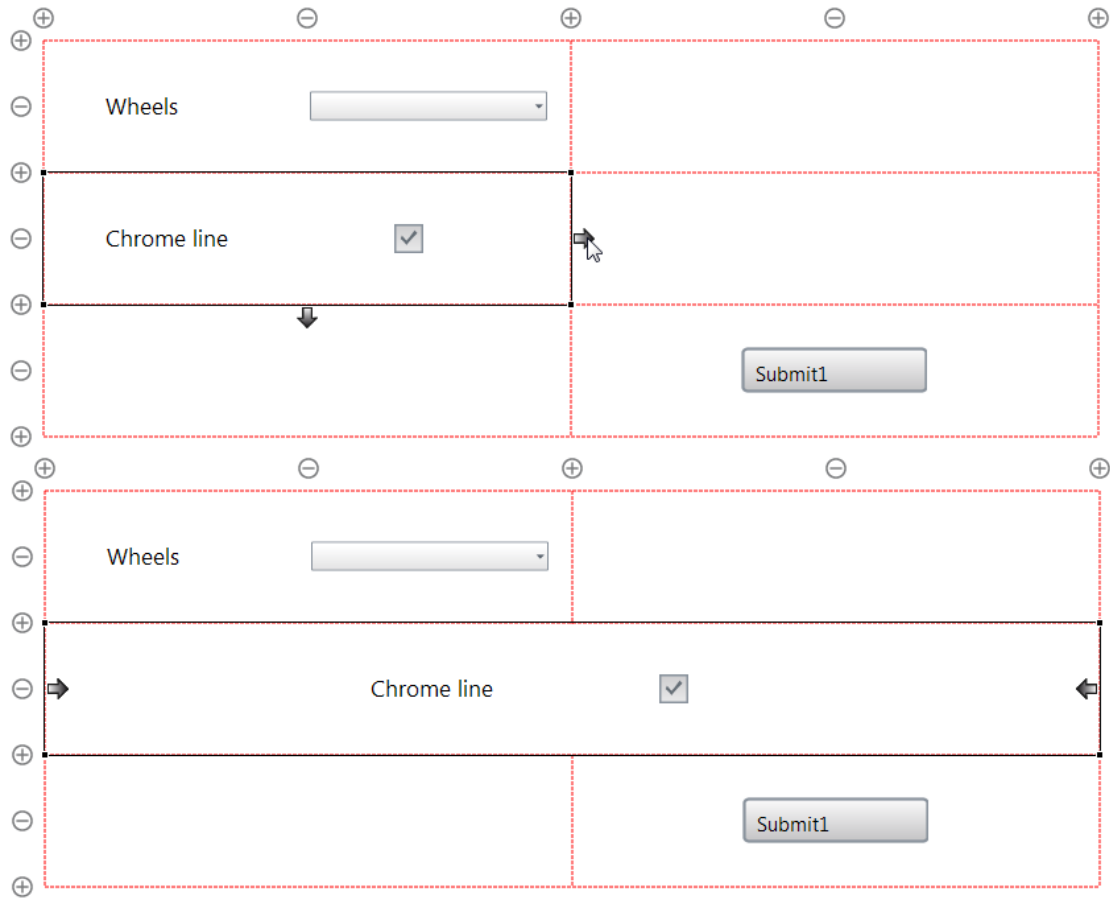


Figure 104. Click and drag to move a cell, click the arrow to merge cells

3.2.1.4 Define and arrange data fields in a Form

Go to **Details -> General** to change the **Name** and **Label** (to be displayed). You can enter a label, use a Groovy expression, or choose not to label a data field. You can also choose to use html for a page name or a data field label. (For security reasons, html is not used by default in forms; you can change forms to html and implement your security precautions if you wish.)

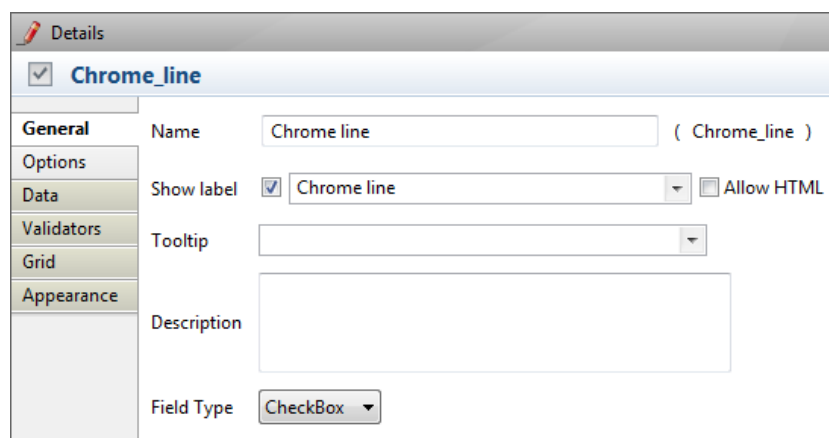


Figure 105. Add General Details to a form field

Add a **Tooltip** here if you want one (enter it directly, or use a Groovy expression). You can enter text, or create a Groovy expression containing defined data variables.

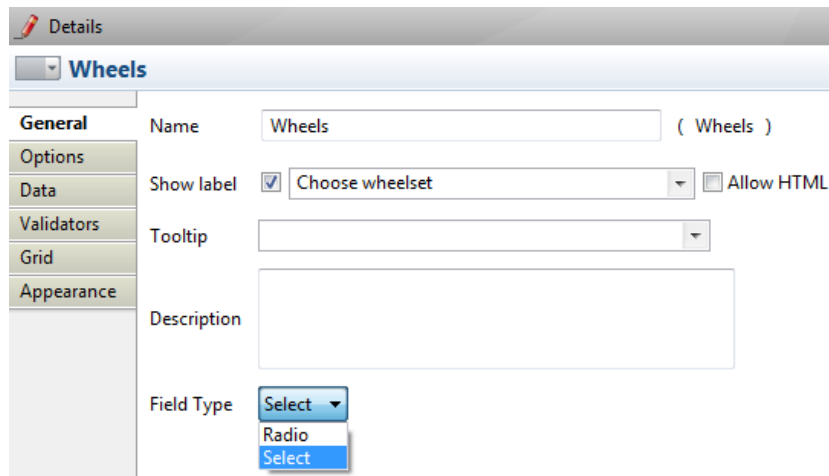
Enter a **Description** of the data field.

The **Field type** associated with field depends on what you have defined. There are “live” fields in which the end user will select a *Boolean yes/no, or enter text or numbers*:

- **Check Box**
- **Date**
- **Duration** enter days/hours/minutes/seconds for a timer event
- **Password**
- **Text Box** single line of text
- **Text Area** multiple lines of text
- **Rich Text Area** user entering data can format the way it appears
- **Table** See below
- **Editable Table** See below

You can use html in some of these fields. Where html is permitted, there is a checkbox to tick.

There are also “live” fields for which the user will select from a *predefined list of options*:



The screenshot shows a configuration window for a data field named "Wheels". The "Field Type" dropdown menu is open, showing three options: "Select", "Radio", and "Select". The "Description" field is empty. Other visible fields include "Name" (Wheels), "Show label" (checked), "Choose wheelset" (dropdown), "Allow HTML" (checkbox), and "Tooltip" (dropdown).

Figure 106. Change data field type

- **CheckBox List** allows multiple selections from a list of items
- **List** allows multiple selections
- **Radio** allows a single selection
- **Select** one item from a drop-down list

You can also add a variable type by selecting it from the Palette on the left side, and dragging it into an empty cell in your Form Builder.

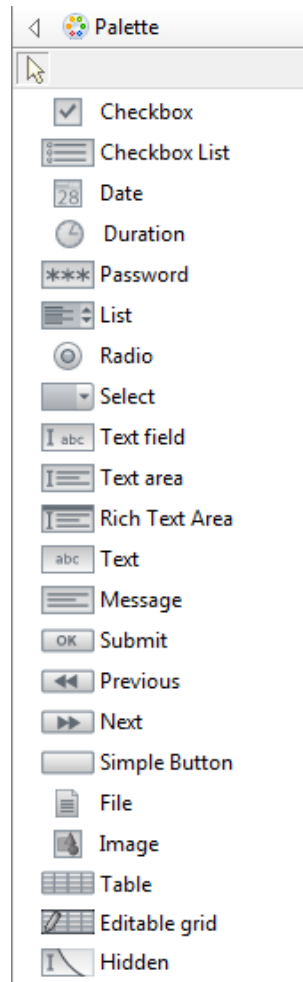


Figure 107. Form Builder Palette

In addition to the fields presented in **Field Type**, there are also fields in the Palette for

- **Text** use to carry over actual data entered earlier in the process, for all data types (can display the result of a calculation or expression); not a live (interactive) field
- **Message** use to present textual information, for example, instructions, that are not interactive

You can add a Table (predefined or editable by the User) to a Form.

- **Table** You can define a table graphically in **Details -> Data**.

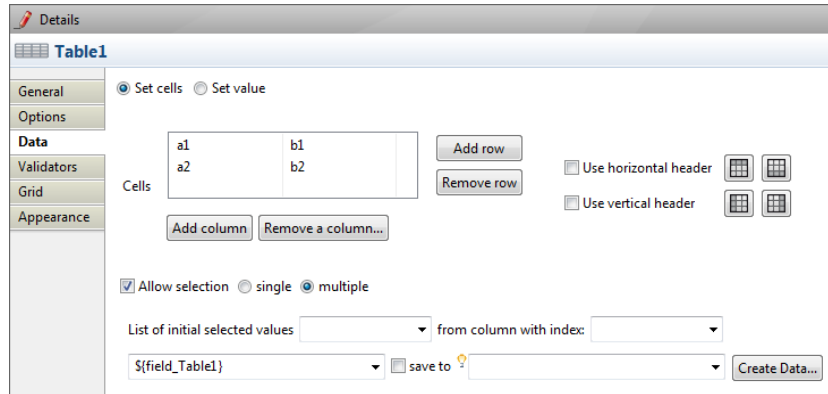


Figure 108. Define a table

You can select one or multiple rows from a Table and save this selection as a variable in form of a list of strings (for example, in the form `$(item1, item2,item3,item4)}` or `List<String>`) by clicking **Allow selection** and either **single** or **multiple**. The items of the list are those from the indicated **Column with index**.

You can define the appearance characteristics for a Table in **Details -> Appearance**, including CSS attributes for the entire table (for example, borders), for individual cells, and for table headers.

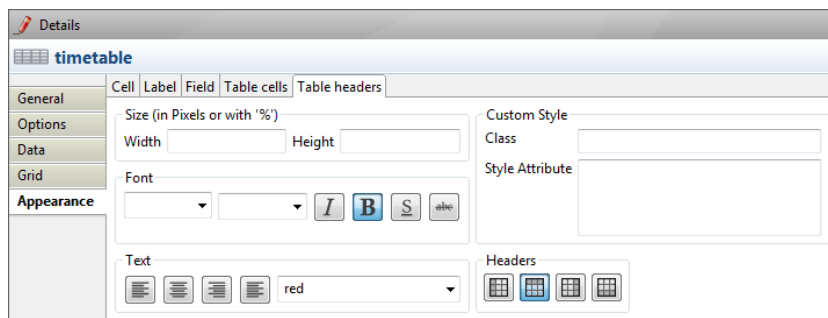


Figure 109. Configure the appearance of a table

- **Editable Grid** An editable Grid is interactive, allowing the User to both enter data into it and manipulate (within limits as defined).

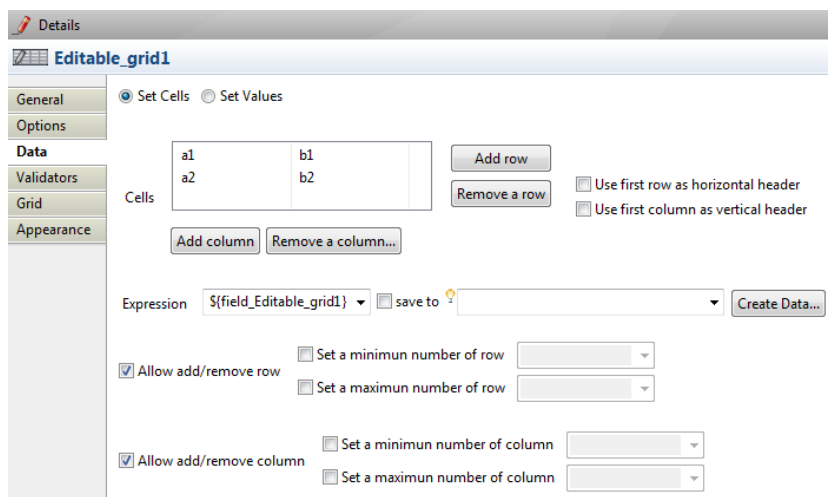


Figure 110. Define an Editable Grid

You can save the contents of an Editable Grid as a variable in form of a list of lists of strings (for example, in the form $\{\{[item1, item2],[item3,item4]\}\}$ or `List<List<String>>`).

You can define the appearance characteristics for an Editable Grid in **Details** -> **Appearance**.

The Form Builder includes 3 **button** types:

- **Submit** moves the Process to the next Step
- **Previous** back up to a previous page in a multi-page form in the same Step
- **Next** move forward to the next page in a multi-page form in the same Step (See [Create multiple page Forms](#)).
- **Simple Button** use this button to call an html script (enter the script in the widget's Options)

You can also add the following to a Form:

- **File** allows attached **File** (attachment) upload (and download)
- **Image** display an image
- **Hidden** use for scripts and data passing not seen by user

If the field must be completed in the form, make is mandatory by selecting **Is mandatory** in **Details** -> **Options**.



Preview the form as you design it by clicking on the **Preview** icon in the Task bar.

This will show you the Form in the default Bonita Form and Web applications.

3.2.1.5 Define or redefine the contents of a field in a Form

Click (to highlight) the field. Go to **Details** -> **Data**. Each type of data field will present options here.

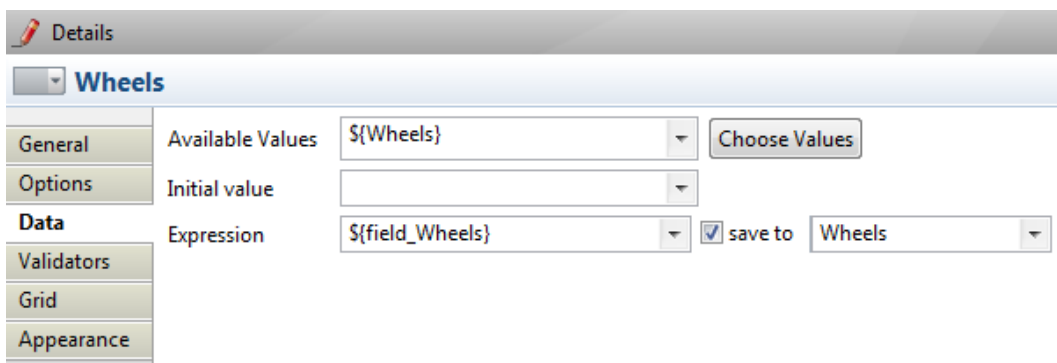


Figure 111. Example: Data definitions for a List field

- **Available values** accepts a List of strings or a Map containing the label and value
- **Initial value** allows you to choose a default or starting value. For **Text** and **Message** fields (these have fixed values), the initial value becomes a constant; you can use input from a variable or enter an expression here.

- **Expression** allows you to either take the value filled in by the User, or to use it in a Groovy expression.
- **Save to** will then be set with the output of the expression (in a Data variable, or a Groovy expression pointing to a Data variable).

For Date fields, you can define the **Display format** here.

Uploaded files can be designated **download only**. If the uploaded file is an image, you can choose to display a preview of it in the Form.

Images can be uploaded and displayed in the final form. There are several ways to upload images:

- Browse to where the image file is located and upload it directly to the form; or
- Upload the image to a Data field as an attachment; then call this attachment in **Step** -> **Forms** -> **Image** [widget in form] -> **Data**, or
- Enter the URL where the image is located.

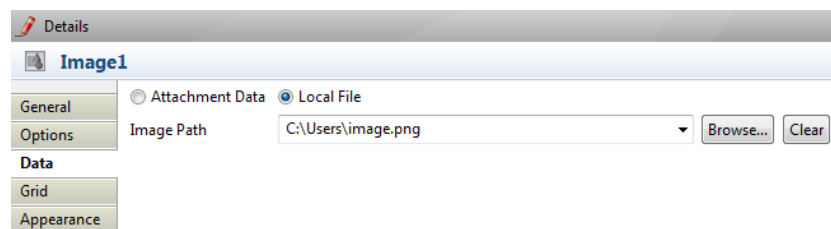


Figure 112. Use an uploaded attachment as an image

NOTE: In order for a data field to be displayed at the completion of this Step, **Initial Value** and **Save To** must be set with the same variable.

3.2.1.6 Restrict the type of data a field will accept

The **Default validator** checks the type of input against the field defined for the entry (e.g., if the field is an integer, it will not accept a character string).

To validate input more specifically (e.g., that a character string is in the form of a date, or an email address), add a customized validator.

To add a custom validator, select a cell with a data field, then **Details** - > **Validators** -> **Add**.

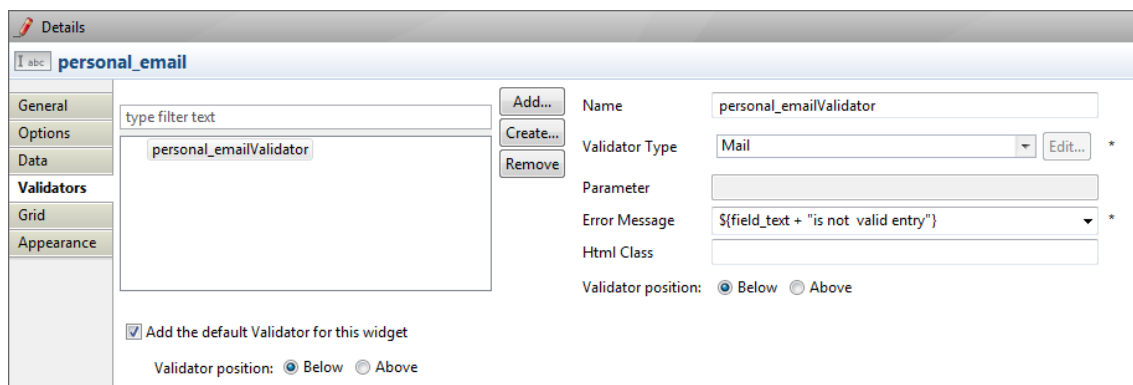


Figure 113. Add a Validator to a field

Name the **Validator**. Choose a **Validator type** appropriate for the field from the dropdown list.

If you want to use a **Regex validator** (Regular Expression), enter the Regular Expression in **Parameter**.

Enter the **Error message** to appear to the user if the field is determined invalid. You can create a dynamic message using **Edit expression**. (Note that this field is mandatory).

You can define your own **html** class for the validator here.

You can also go to **Details -> Validators -> Create** to define your own Validator here by defining a **Java class**, **package** and **Display Name**.

Both the default and the custom validator(s) can each be positioned above or below the entry field in the Form.

3.2.1.7 Resize columns and rows

Select a data field cell, then **Details -> Grid**.

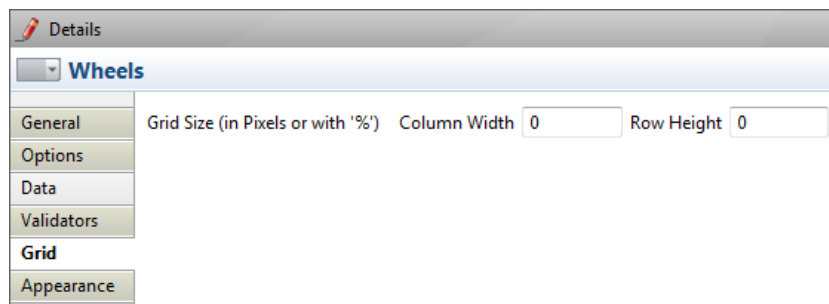


Figure 114. Change the size of a column or row

Choose a cell in a Column and enter a column width and /or row height in pixels or in percentage (of the total width or height occupied by the completed form). Note that the total width and height of the grid is constrained by the dimensions of the form it is presented in.

3.2.1.8 Change the appearance of a cell, label, or field

Go to **Details -> Appearance** to:

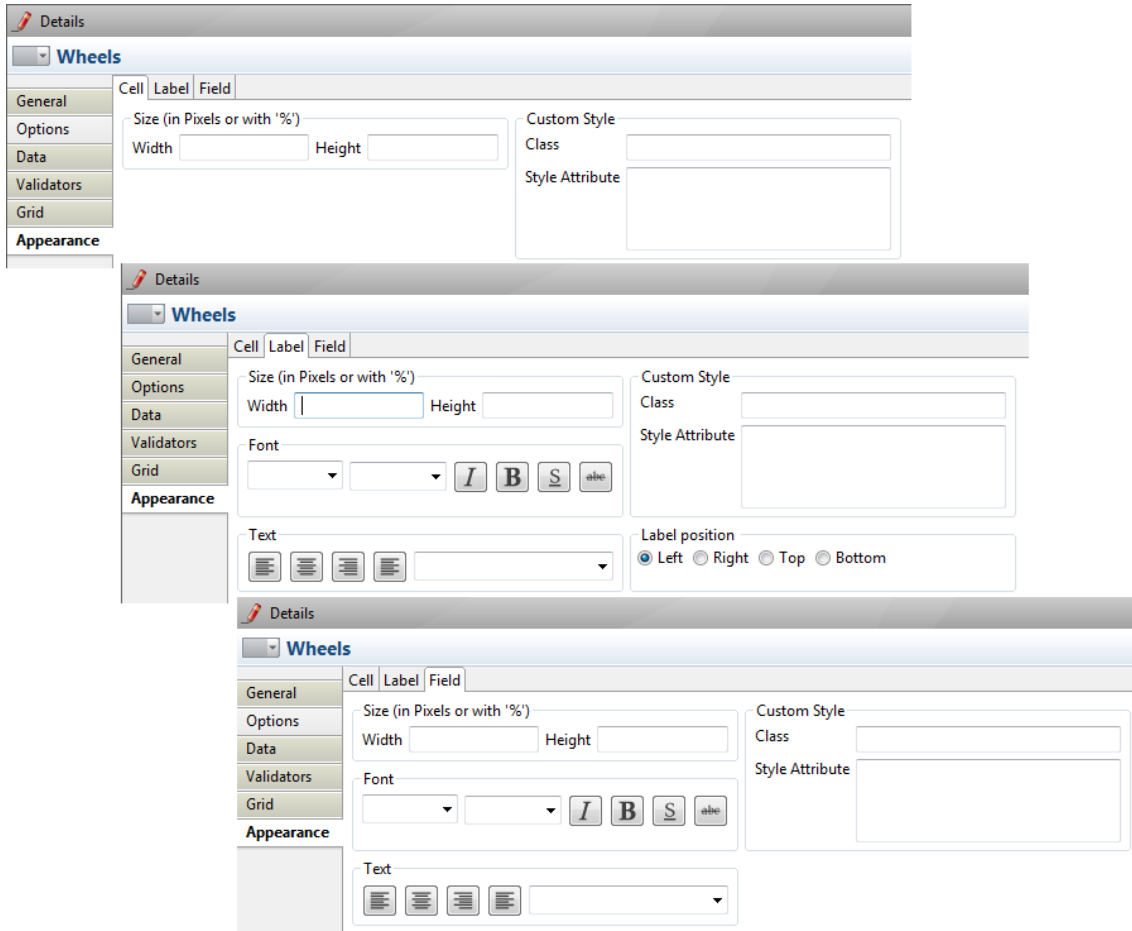


Figure 115. Change the appearance of a call, a label, or a field

- Resize a **Cell** (inside of specified Grid)
- Change **Label** font, size, placement, and so on
- Change **Field** font, size, placement, length, and so on

You can also upload your own **Custom Style** in a CSS file for all or each of these.

3.2.2 Create multiple page Forms

When you add more Forms to a Step, they are handled as multi-page Forms. Add each Form in the same way you create a new Form.

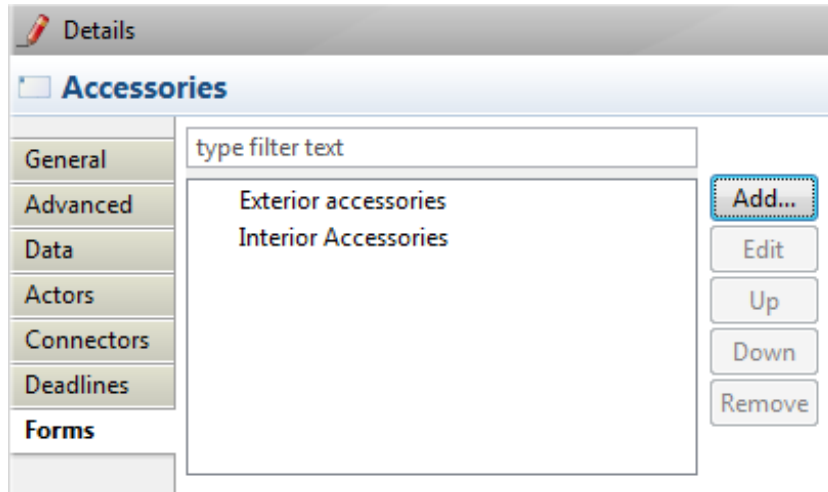


Figure 116. Several forms defined in the same step create a multipage form

Each new form will add a **Previous** button.

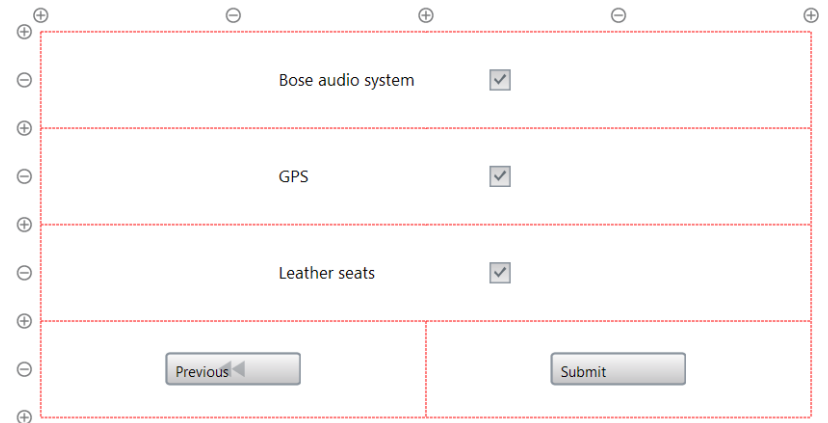


Figure 117. Each subsequent page in a multi-page form will automatically create a Previous button

For more than 2-page forms, change the **Submit** buttons to **Next** buttons by clicking on the button, then on **General**, and then change its **Field Type**.

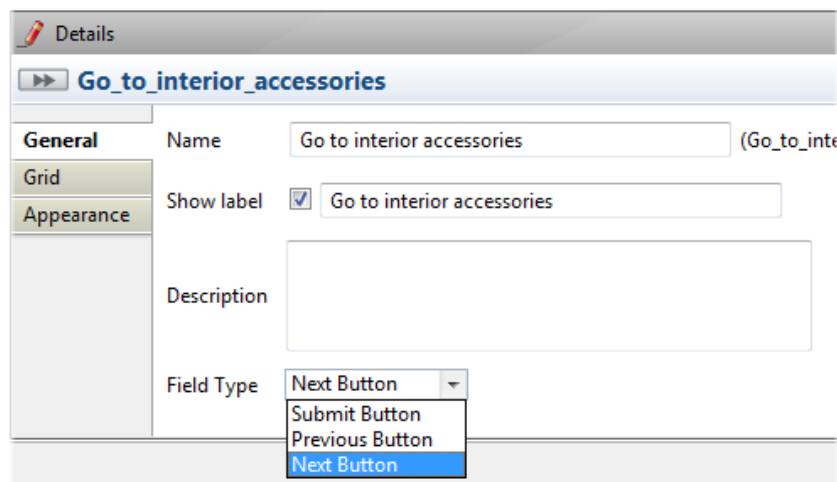


Figure 118. For preceding Forms, change Submit buttons to Next Buttons

Navigate between multiple form pages with the arrows as shown:

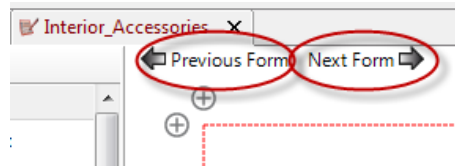


Figure 119. Navigate between multiple forms

Multi-page forms are presented in the order they are listed in **Details -> Forms**. To change their order, move them **Up** or **Down**.

3.3 How to create a Form for a Process

Defining a Form at the Pool or Process level is essentially the same as for a Step. If a Form is needed to start a case of the Process, define it here. If the Process is to be started and the first Form presented is that in the first Step, choose **Skip the initial forms at process start**.



Figure 120. Add an Overview Form at the Process level

3.4 How to customize templates for Web Forms

Each Form created for a Step has essentially 3 levels of template that define its look and feel:

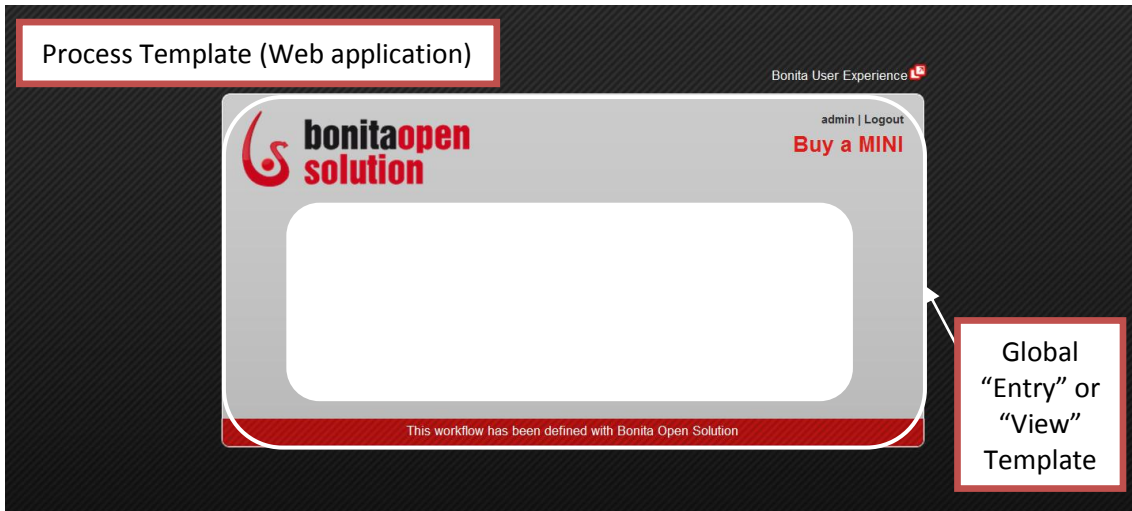


Figure 121. Customizable Process template and web application template

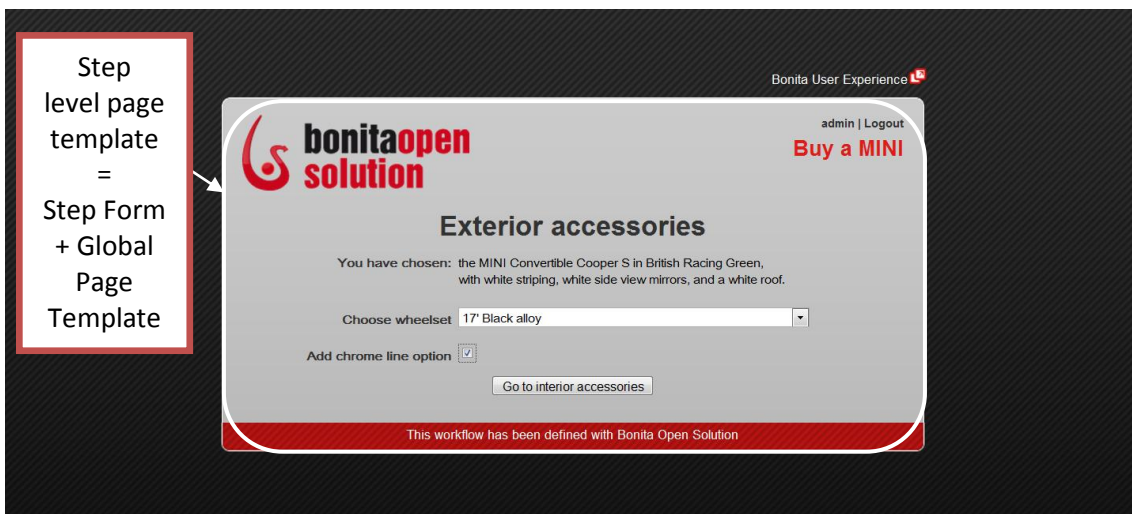


Figure 122. Customizable Page template

- A Process Template (Web application) which “surrounds” all Forms and defines the overall look-and-feel.
- A Global Page Template which surrounds a Form defined for a Step, and specifies the location of common elements, such as the title of the Form, the name of the user assigned to the Step, etc.
- A Step level Template which incorporates both the Global “Entry” or Global “View” template and the individual Form (data Fields and their characteristics) defined for the Step (as in [Design a Step Form](#)).

The following paragraphs describe how to customize these templates.

3.4.1 Customize the Process Template (Web application)

The built-in, default template in which all Bonita Open Solution Forms are presented is an html file called at the Process level. Five built-in process templates are provided. Each contains a div `bonita_form` which is filled with the appropriate form at runtime.

To see the templates provided, go to **Pool -> Details -> Applications -> Look'n'Feel -> Apply a Look'n Feel**.

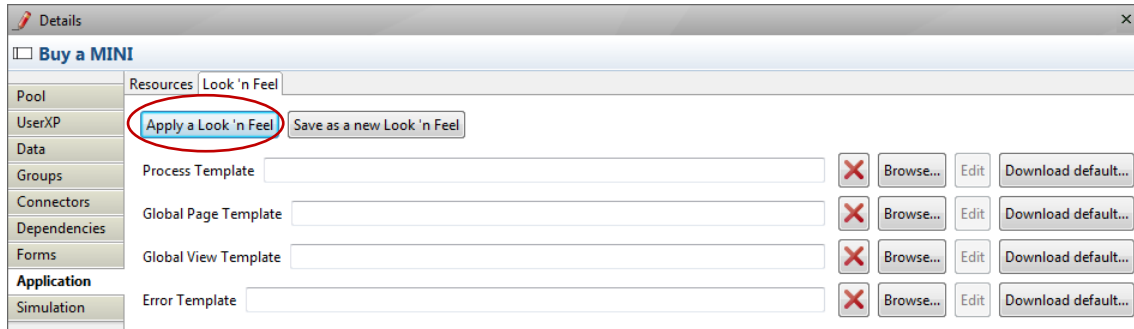


Figure 123. Change the Look and Feel of all process Forms

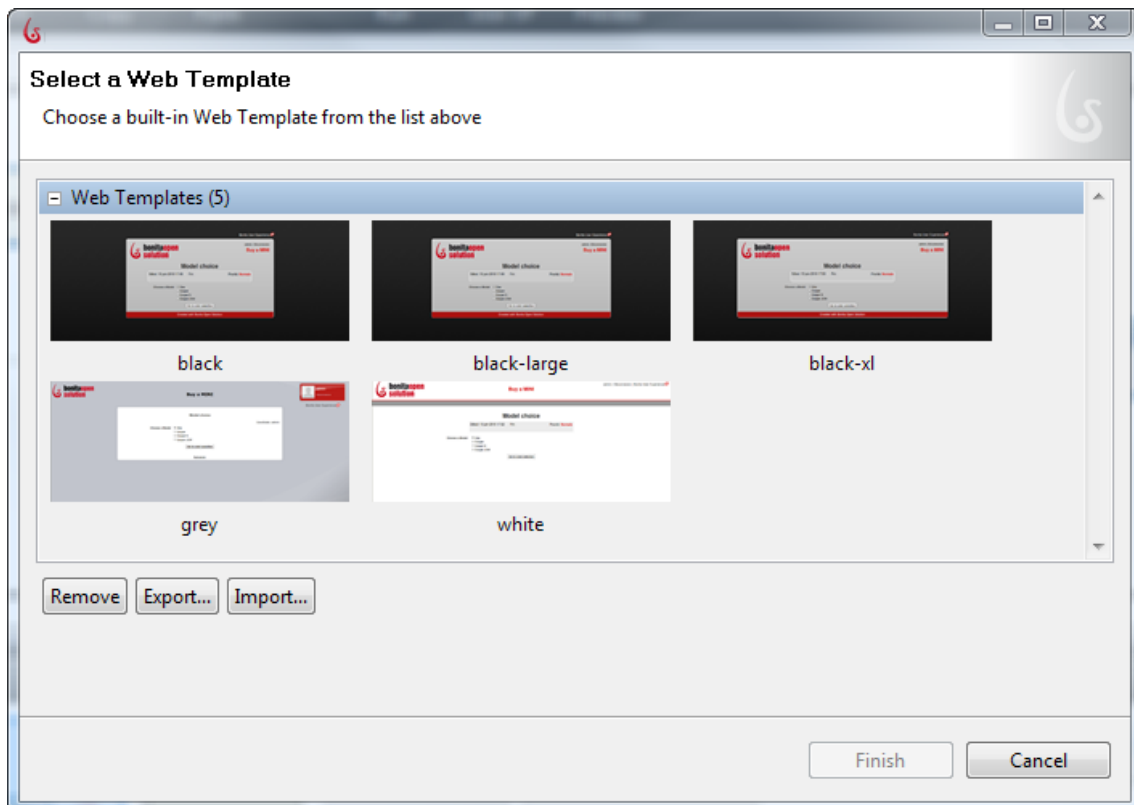


Figure 124. Change template for all forms presented in this Process

From here you can import another template, remove any of the built-in templates, or export any templates in this group.

When you select a built-in template, its resources are uploaded automatically.

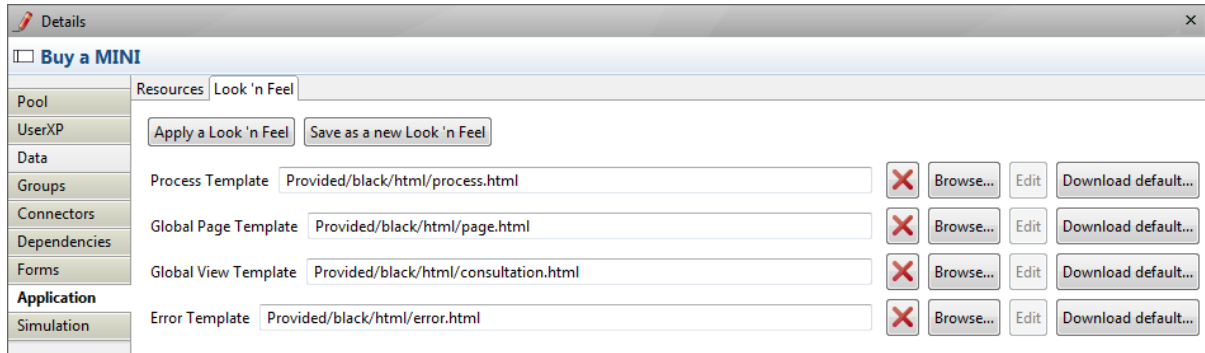


Figure 125. Resources for default “black” template

The default template is “black.” When it is applied manually, its resources (CSS and images) are shown as in the Figure above.

You can change or replace it with your own template here too. **Browse** to upload a **Process Template**.

If you choose a non-built-in template, it must contain an element with the id `bonita_form`. The following table shows both mandatory and optional element IDs. Be sure to also upload the appropriate resource folders or files (use **Add folder** or **Add file**.)

ELEMENT ID	Defines	Optional/Mandatory
<code>bonita_logout_box</code>	logout box	optional
<code>bonita_user_xp_link</code>	link to the User Experience	optional
<code>bonita_process_label</code>	process name	optional
<code>bonita_form</code>	container for the template for this form page	mandatory
<code>bonita_refresh_button</code>	button to refresh the form	optional

Table 1. ELEMENT IDs for forms defined at the Process level

Here is an example of a process template:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html dir="ltr" xml:lang="en"
xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
    <meta name="description" content="Bonita Forms
Application"/>
    <title>My application</title>
    <link href="css/bonita_form_default.css"
rel="stylesheet "type="text/CSS"/>
  </head>
  <body>
    <div element ID="bonita_form">
    </div>
  </body>
</html>
```

3.4.2 Customize a Step level Template

After you define a Form or Forms for a Step (see [Define a Step Form](#)), a step level default page template is generated by Bonita Open Solution and linked with an html file that directs how the Form is to be displayed inside the Global “Entry” or “View” Template. The Step level and Global page templates make up the Step level Template.

To modify the Step level Template, go to **Step -> Forms** and select any Form you have defined. Click **Edit**. Select **Download default**, and save the html file generated.

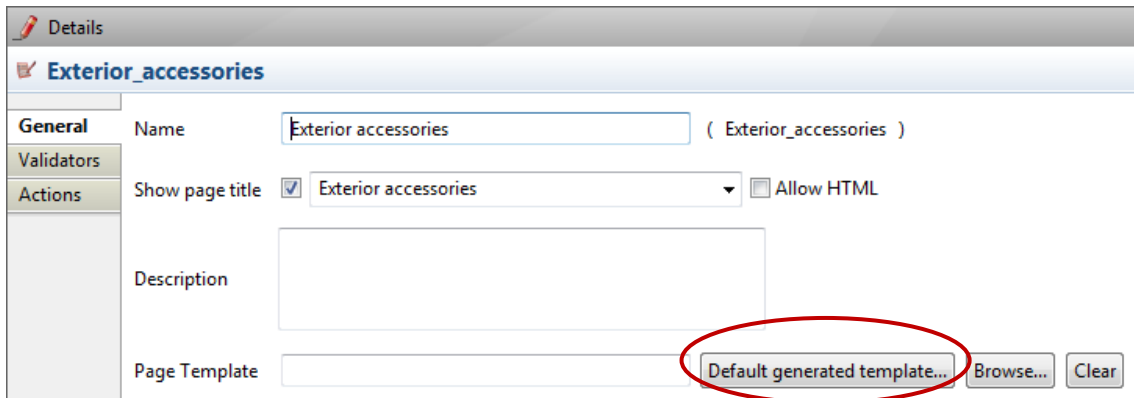


Figure 126. Download Step-level form template generated by Bonita Open Solution

You can now edit this `generatedTemplate.html` file.

For each data field present in the form, the html contains a div with its name as an ID. The entire Form is displayed inside the Process level template where the div `bonita_form` is located.

To create a generic page template for all Forms based on this one, you can remove the specific `bonita_form_container` code generated with this template and modify the rest of the html content as desired.

To apply this new template, go to **Process -> Application -> Look 'n Feel** and **Browse** to upload the new Global “Entry” Template.

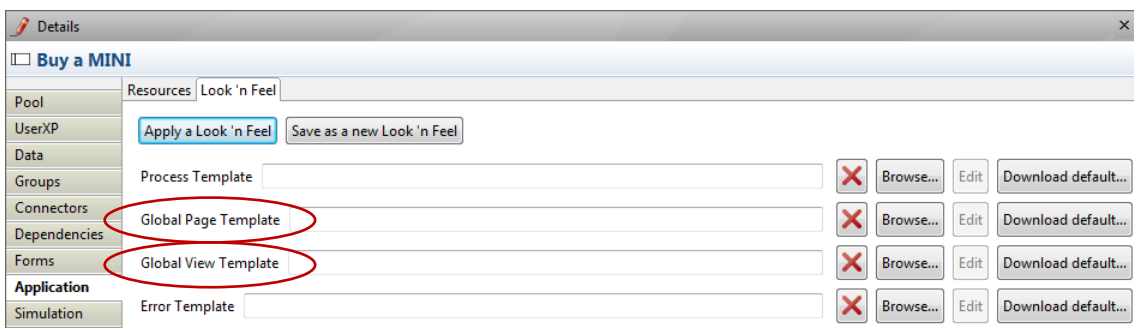


Figure 127. Upload modified Global “Entry” or “View” Template

Alternatively, you can upload your own html page template here.

All individual page Forms will subsequently be generated with their unique `bonita_form_container` code, according to how you have defined the fields in the Form Builder .

To use a unique page template for a single specific Form, go to **Step -> Forms**. **Select** the Form you want to customize and **Edit -> General**. Go to **Page Template**, **Browse** to where your html file is and upload it.

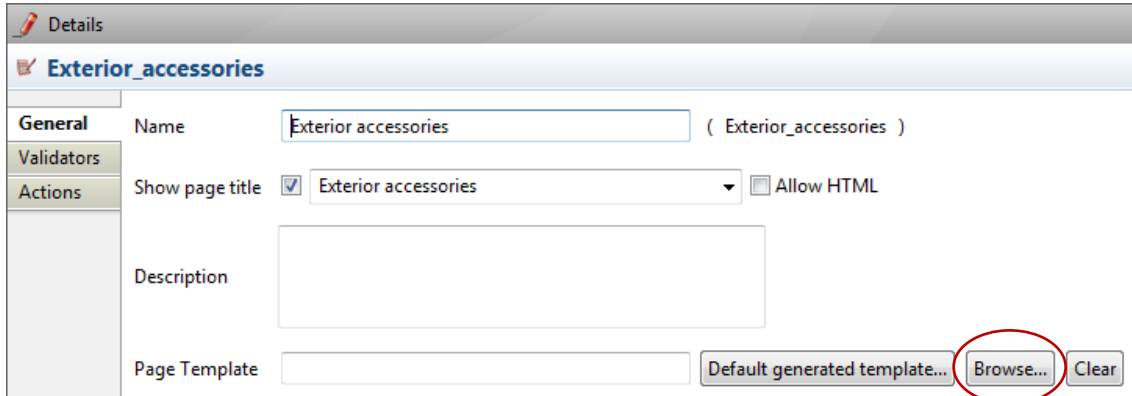


Figure 128. Upload your custom html Form template

Note: There must be an element ID included for each data field, and an element ID for each validator, or the form will not display.

ELEMENT ID	Defines	Optional/Mandatory
<code>bonita_form_page_label</code>	title of the form presented on this page	optional (BOS V 5.1.1 and higher)
for each data field		mandatory
for each validator		mandatory

Table 2. ELEMENT IDs for forms defined at the Step level

The ID `bonita_form_page_label` might be useful to display the label of the current form.

In addition, you can put the following elements in your page template (if defined, they will be replaced at runtime by the real value for the step):

placeholder	Defines
<code>\$bonita_step_state</code>	The current state of the Step
<code>\$bonita_step_assignee</code>	The Step Actor
<code>\$bonita_step_candidates</code>	The candidate Actors for this Step
<code>\$bonita_step_lastUpdate</code>	The last time the Step was changed
<code>\$bonita_step_label</code>	Name of the step as defined in Studio
<code>\$bonita_step_description</code>	description of the step as defined in Studio
<code>\$bonita_step_createdDate</code>	Date and time that the Step was created in runtime
<code>\$bonita_step_readyDate</code>	Date and time the step became available for execution by a candidate
<code>\$bonita_step_startedDate</code>	Date and time the Step Form was Submitted
<code>\$bonita_step_endedDate</code>	Date and time Case moves to the next Step, i.e., after variables (if any) are set by the

	runtime
\$bonita_step_executedBy	Actor that submitted the Step Form
\$bonita_step_expectedEndDate	The readyDate plus any duration that was specified in Studio
\$bonita_step_priority	Step priority as defined in Studio or the User XP
\$bonita_step_remainingTime	expectedEndDate less the current date and time

See the example below, a page template for a simple Form which contains:

- a data field named `CheckBox1`;
- a validator named `CheckBox1_validator`; and
- a submit button named `Submit`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html dir="ltr" xml:lang="en"
xmlns="http://www.w3.org/1999/xhtml" lang="en">
<body>
  <element ID="bonita_form_task_candidates">
</div>
  <element ID="bonita_form_page_label">
</div>
  <element ID="CheckBox1_validator">
</div>
  <div id="CheckBox1">
</div>
  <element ID="Submit">
</div>
</body>
</html>
```

3.5 How to customize the Error template, Welcome page, and Log-in page

These are all defined at the Process level.

3.5.1 Customize error messages

An error message is shown instead of the form page when an error occurs. The message is presented in the Error template. It is found in **Process -> Application -> Look 'n Feel**.

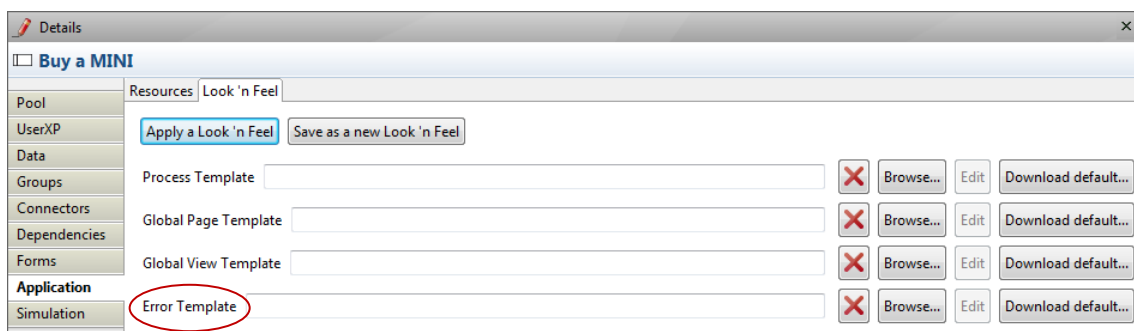


Figure 129. Upload a custom template for error messages

You can customize it: download the default, modify the file, and then Browse to upload the new template. If you have already defined a template, you can simple Browse to upload it.

If you want to use your own template for the global error message, include the div with the IDs `bonita_form_error_message` and `bonita_form_cause_message` to display the error message and error cause. Note that there can be only one error template per process/application.

ELEMENT ID	Defines	Optional/Mandatory
<code>bonita_form_error_message</code>	error message	mandatory
<code>bonita_form_cause_message</code>	cause of the error	mandatory

For an example, see the default error template below:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html dir="ltr" xml:lang="en"
xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
    <meta name="description" content="Bonita Forms
Application"/>
    <link href="css/bonita_form_default.css"
rel="stylesheet" type="text/CSS"/>
  </head>
  <body>
    <div class="bonita_form_container">
      <element ID="bonita_form_error_message"
class="bonita_form_error_message"></div>
      <element ID="bonita_form_cause_message"
class="bonita_form_cause_message"></div>
    </div>
  </body>
</html>
```

3.5.2 Customize Welcome page

The Welcome page is displayed before the login page when the User goes directly to the Process application URL (with no parameters). It can be located internally (local html file) or externally (web URL).

The Welcome page template is found in **Process -> Application -> Resources**.

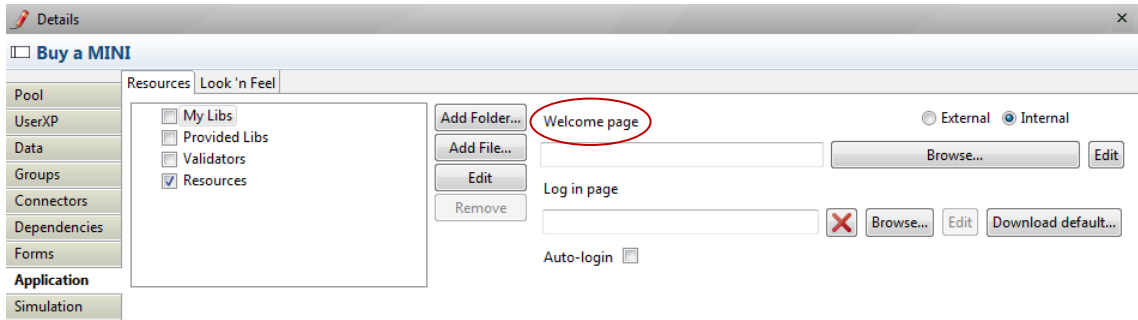


Figure 130. Upload a custom template for the Welcome page

Choose “internal” and browse to the html file to use, or choose “external” to enter the URL.

An example of an internal welcome page:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html dir="ltr" xml:lang="en"
xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
    <title>Welcome</title>
  </head>
  <body>
    <a href="?process=MyProcess4-1.0">start my
process</a>
  </body>
</html>
```

3.5.3 Customize Log in page

The Log in page, which authenticates users, is a JSP (JavaServer Page) and uses a Bonita API to check the user credentials. The **Download default** button will bring up the default `login.jsp` which you can then save and modify. When your customized login page is completed, browse to upload it here.

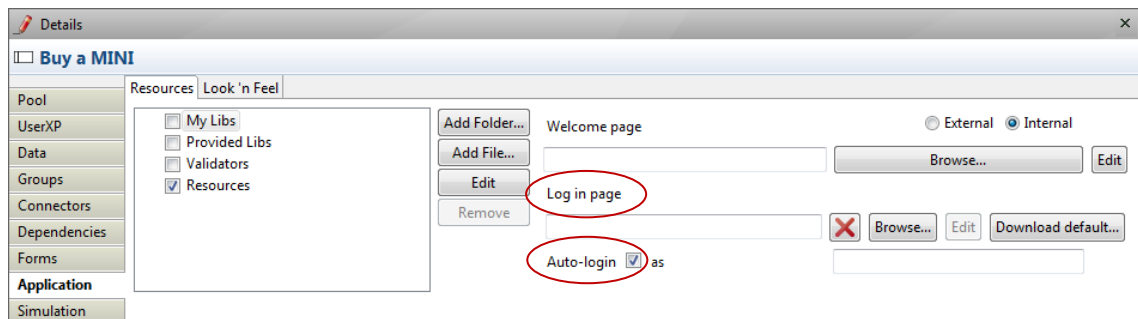


Figure 131. Upload a custom template for the Log in page

Auto-login can be activated for multi-instantiated steps. If Auto-login is checked and the field left blank, the end User(s) completing the Log In form will be identified as “Anonymous.” If the “as” field is filled, that “User” must be identified in the User List (eg, “guest,” “voter,” “requestor.”)

3.6 How to customize the Confirmation template and messages

The confirmation message is shown whenever a Process Case moves from one Actor / User to the next. That is, at the end of a sequence of Steps performed by one User, a confirmation message is shown and the subsequent (Human) Step shows up in the inbox of the next Actor / User. (For example, this is what happens when a Case is started by an Initiator submitting a final form.) The confirmation template is used for this message. The confirmation template can be customized for each Step where it is presented. It can also be defined at the Process level for an initial form.

Go to **Process -> Pool -> Forms**.

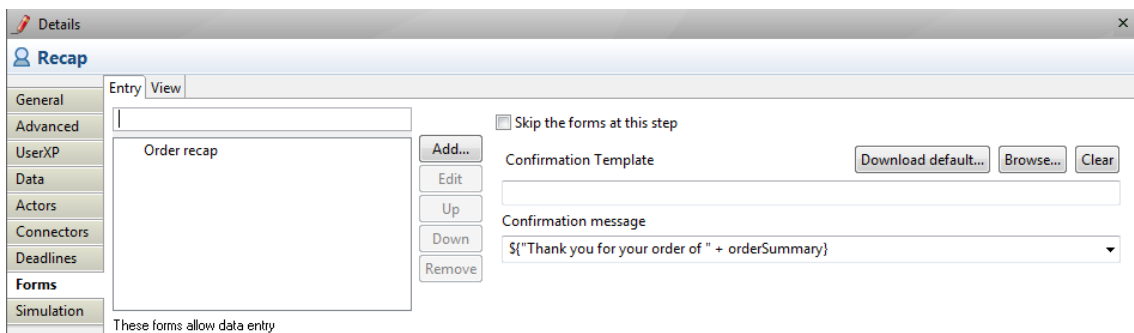


Figure 132. Upload a custom html template or create a message for a confirmation page at the Step level

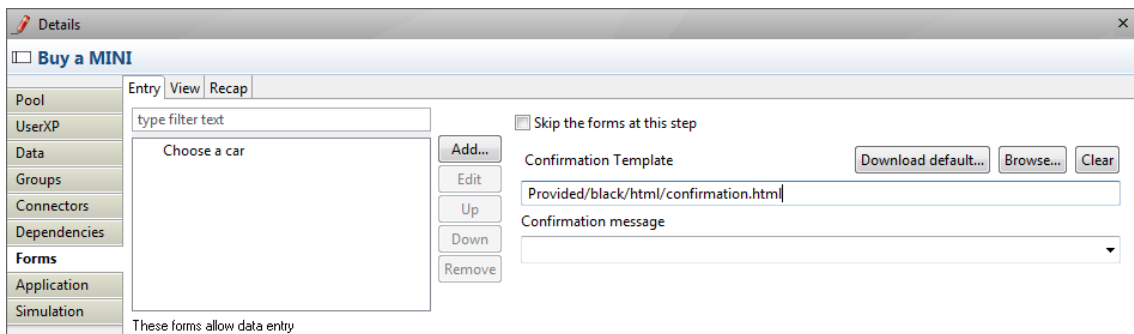


Figure 133. a custom html template or create a message for a confirmation page at the Process level

See the example below.

In this example template a div with the ID `bonita_form_confirm_message` is used to display the default confirmation message.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html dir="ltr" xml:lang="en"
xmlns="http://www.w3.org/1999/xhtml" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
    <meta name="description" content="Bonita Forms
Application"/>
    <link href="css/bonita_form_default.css"
rel="stylesheet" type="text/CSS"/>
  </head>
  <body>
    <div class="bonita_form_container">
      <div ID="bonita_form_confirm_message"
class="bonita_form_confirm_message"></div>
    </div>
  </body>
</html>
```

Part 4. How to use Bonita User Experience

4.1 Overview

Bonita User Experience (User XP) provides the interface for managing Steps, Cases, and Processes.

In this section we'll look at three ways to interact with the User Experience.


- As the Process Developer (the person who can design, deploy, modify Processes in Bonita Studio)
- As the Process Administrator (the person responsible for managing a set of Processes deployed by the developer)
- As an End User internal to the Process (e.g., an employee within the company with responsibility to act in order to complete a Step or Steps in the Process case)

4.2 Bonita User Experience – Developer's View

Bonita User Experience for the developer permits him/her to see and test all of various User functions (Administrator's View and End User's View) as he/she is designing (or modifying) a Process for deployment.

This section defines the functions of the User Experience unique to the developer.

Note that all of the functions in [Bonita User Experience – Administrator's View](#) and [Bonita User Experience – End User's View](#) are also visible to the developer.

To open **Bonita User Experience** (UserXP), click on the UserXP button  in the Bonita Studio Task bar OR Select **Run** -> **User XP** from the Bonita Studio menu bar.

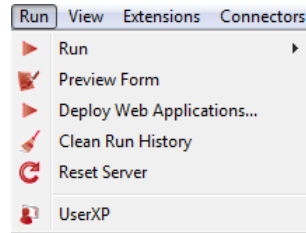


Figure 134. Open Bonita User Experience from the Menu bar

Generally, when you click **Run** to deploy a Process from Bonita Studio, an initial Form (or series of Forms) is presented in the default Bonita Web application (Web page) in your usual Web browser. You can change these: see [Customize Form page \(change page html\)](#) , [Incorporate the Bonita Form into your Web application](#), and [Change Web browser for User Experience](#).

User Experience can be opened from the default Bonita Web application (web page).

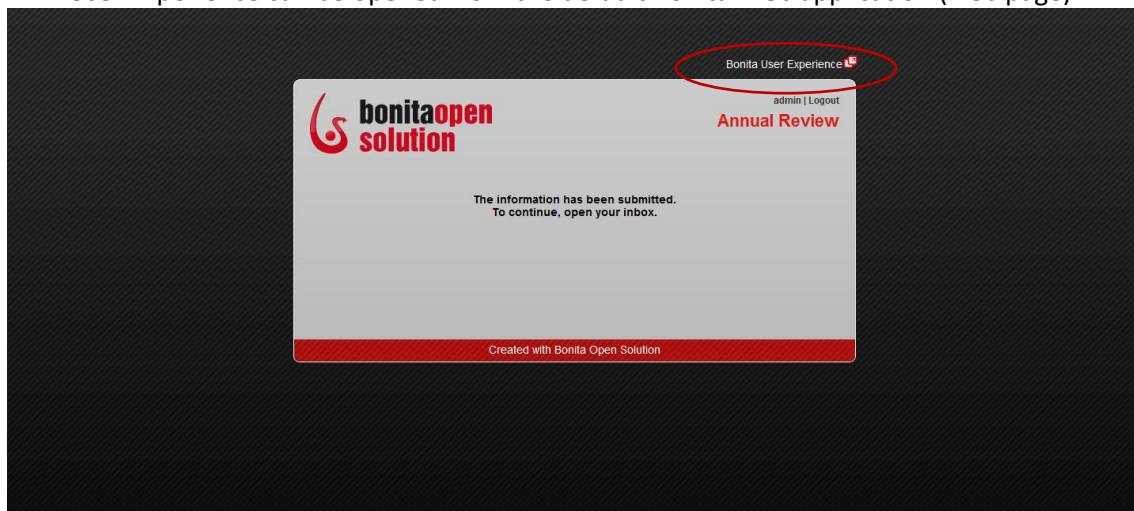


Figure 135. Open Bonita User Experience from the Bonita Web application

The Bonita User Experience inbox is a Web-based, e-mail like interface. As the Process developer entering from Bonita Studio, you are automatically logged in as Admin with the password bpm. To change this default, see [Change user for whom User Experience will open when a Process is Run](#).

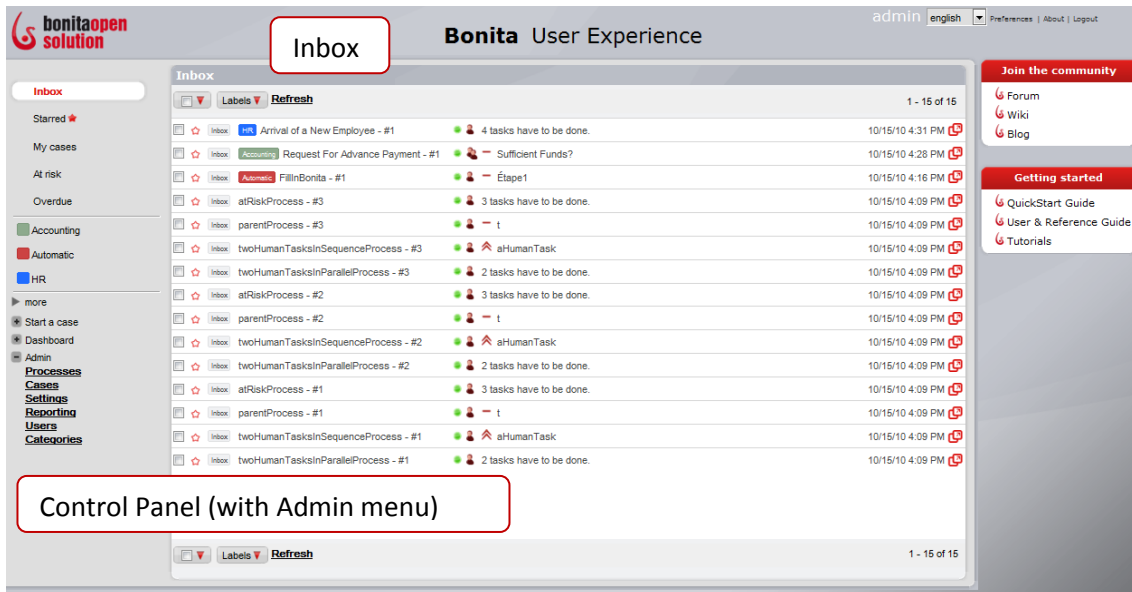


Figure 136. Bonita User Experience as deployed by developer/admin, showing Inbox

The Inbox will be used by the Process administrator (as configured by the developer) to manage Steps, Cases and Processes:

- A Case is a single instance (instantiation) of a Process (each recurrence of a Process is a new Case).
- Deployed Processes show up only in the User Experience of the Admin; that is, only the Admin (and not End Users) can manage Processes.

The User Experience is the interface between Step Actors (Users) and the Process Cases they are acting in. At each (Human) Step in a Case of a Process, a Form is presented – the User enters data, chooses options, etc in a Form that appears much like an email message.

4.2.1 How to define or change Bonita User Experience preferences

To change User Experience preferences, go to Bonita Studio and from the menu bar, select **Edit -> Preferences -> Bonita -> User Experience**.

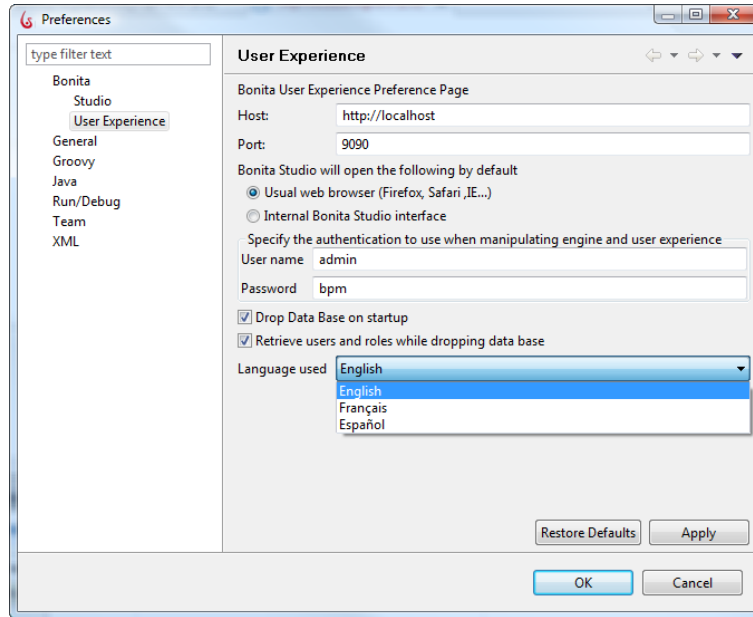


Figure 137. Change Bonita User Experience preferences

- Host : the host where the Web Browser is opened
- Port : the port used by Embedded Jetty server

4.2.1.1 Change Web Browser for User Experience

In this dialog, you can select your system's usual web browser or the internal Bonita Studio interface.

4.2.1.2 Change user for whom User Experience will open when a Process is Run

The default User for whom User Experience first opens when a Process is **Run** (deployed) from Bonita Studio is the Administrator (User name `admin`). Here you can change the User name and password, after you have defined your Users, their Roles, and their passwords (see [Define Users and User Roles](#)).

4.2.1.3 Keep User Experience contents after closing Bonita Studio

By default the contents of Bonita User Experience during development are deleted when you close Bonita Studio. To keep the contents during development, uncheck **Drop Data base on startup**. Check **Retrieve users and roles** to keep users and roles when the database is dropped.

4.2.1.4 Change language

Language for User Experience is automatically detected through your Web browser. You change it via **Edit -> Preferences -> User Experience** in Bonita Studio, or from the User Experience inbox itself.

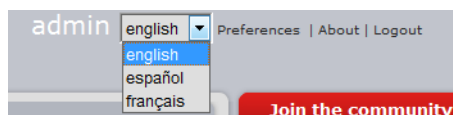


Figure 138. Change language from within User Experience

4.3 Bonita User Experience – End User’s View

The User Experience for End Users is intended for the Actors who have tasks to perform in a Process. Each End User’s inbox:

- permits them to see what tasks are waiting for them to complete
- provides a Form for data entry
- shows the overview and history information of Cases

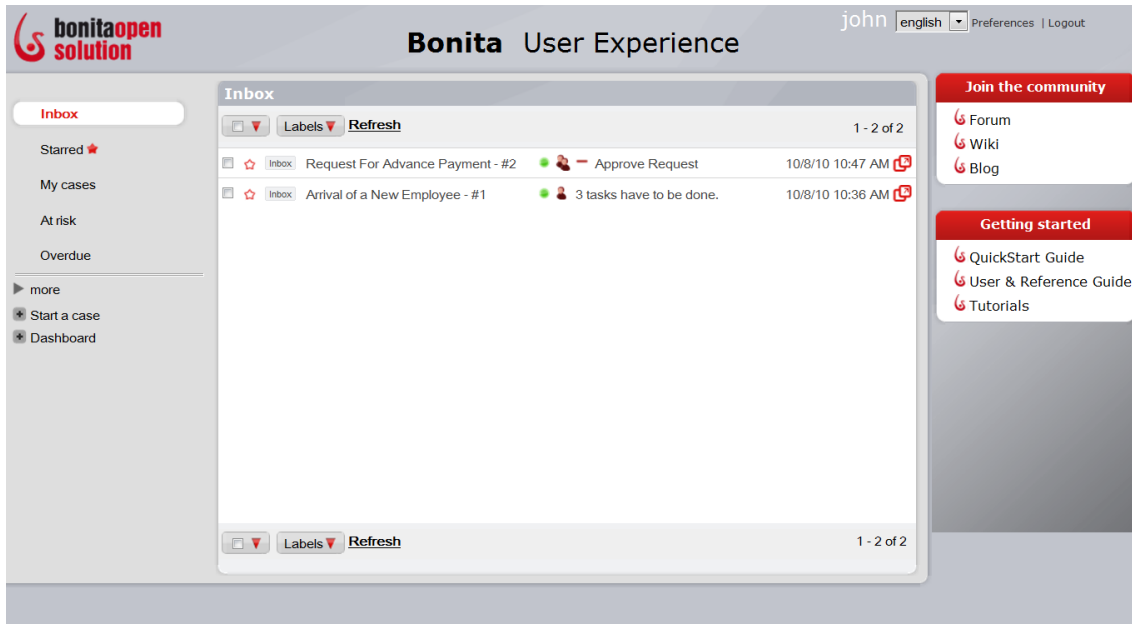


Figure 139. Bonita User Experience inbox as seen by end user

What information is presented in the User Experience – what information an individual has access to – is defined by the Process designer.

4.3.1 How to execute and manage Steps in a Case

Click on a line (Case) in the Inbox to open the next Step(s) requiring action.

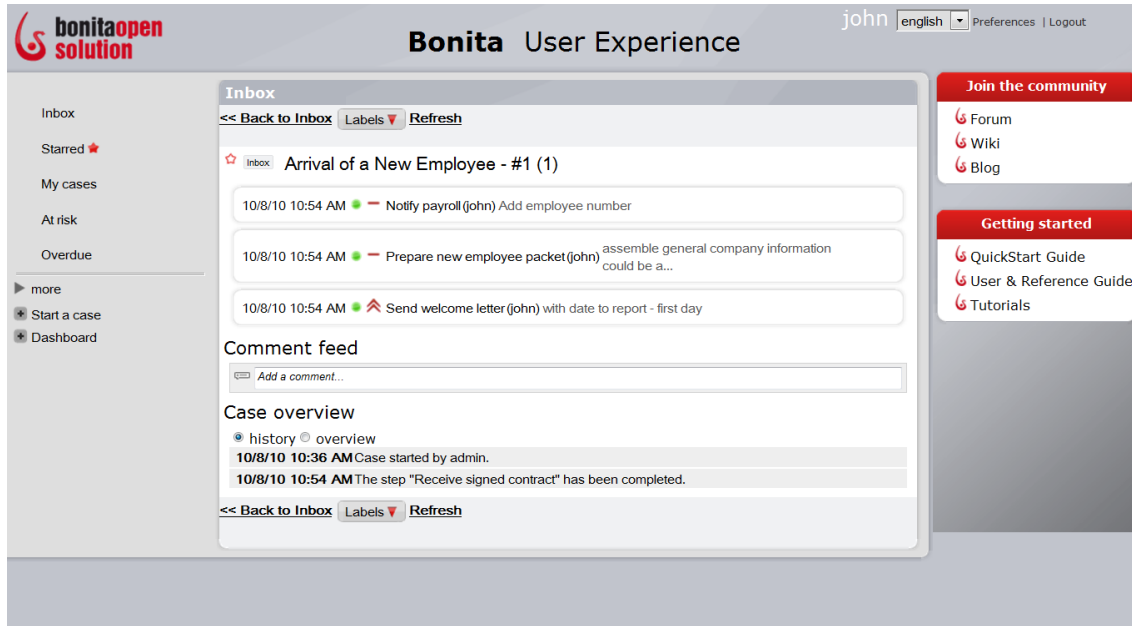


Figure 140. Managing Steps in Bonita User Experience

In this view, the User can see pending Steps waiting to be acted on in the Process, comments attached to this Case of the Process, and the overview of what has already happened in this Case.

For each Step you will see the following markers. Mouse over each one to identify it:

Item	Example /explanation
date and time stamp	date and time the last action was taken in this Case
a colored bullet: green, orange, or grey	<p>green: a Step is to be completed</p> <p>orange: a Step has been suspended</p> <p>grey: all Steps have been completed</p> <p>click on this button to expand or close a step</p>
priority chevron	<p>— normal</p> <p>^ high</p> <p>^^ urgent</p>
Title	The title of the Step as defined in the Process. This can include dynamic content (as defined in General details for a Step).
Actor(s)	Actor assigned to Step (in parentheses)
Description	Description of the Task as defined in the Process. This can include dynamic content (as defined in General details for a Step).

Click on a pending Step to open it.

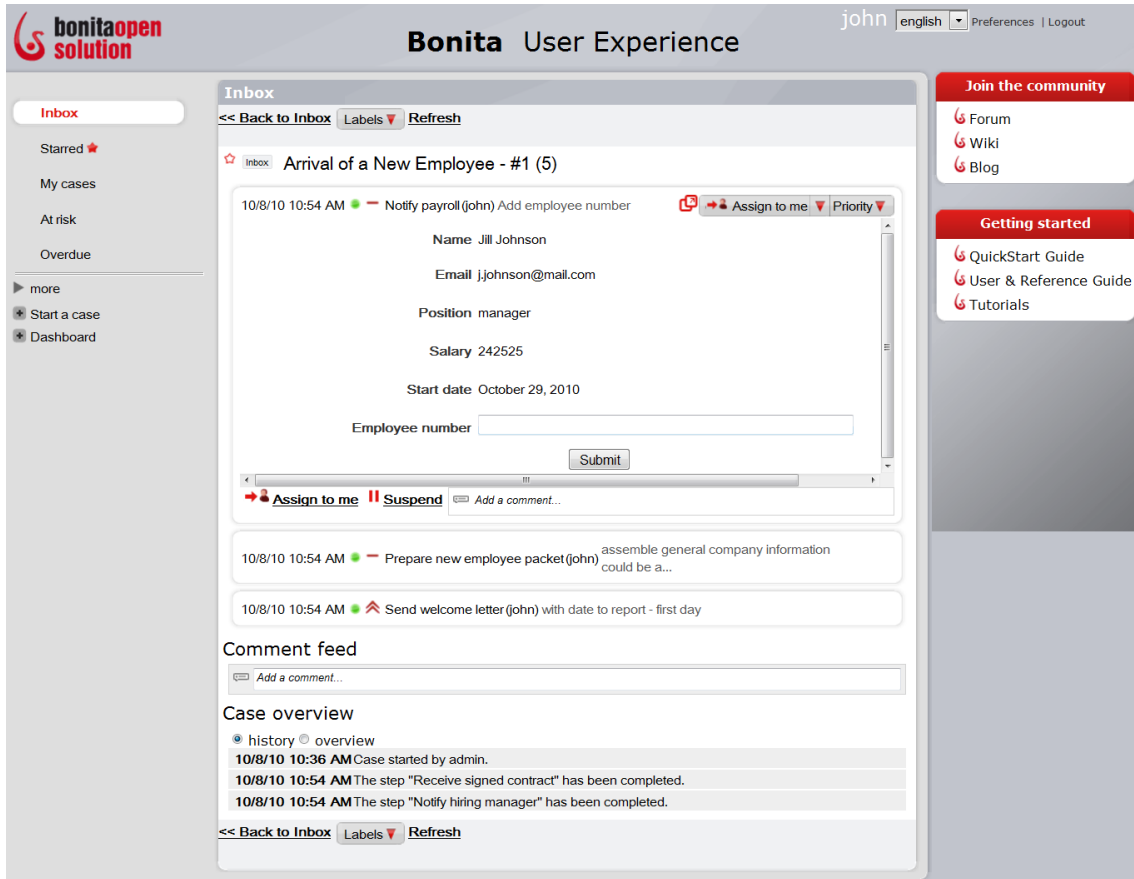

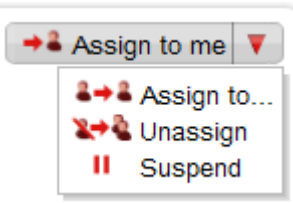
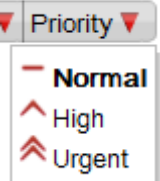


Figure 141. Pending Step awaiting action by the Actor

The User is presented with a Form to complete, as defined in the Process. There are now several more actions that the User can take.

<p>new window icon </p>	<p>open this Step in the designated Web app / Web page</p>
<p>assign to me menu</p> 	<ul style="list-style-type: none"> • If the Step assigned to multiple Actors, it can be claimed or forwarded by this User • Step can be suspended for later action
<p>priority menu</p> 	<p>Can change the priority of this Step</p>

The User can add comments or note to a Step here. Click **Add a comment** and enter. (The comment is not editable once you hit *Save*).

When the form is submitted, the Process continues to the next Step.

If a Step can be processed by more than one Actor in an assigned Group, it shows up as pending in all of those Users' inboxes. However, when one person completes that task it shows up as completed, instead of pending, in other Users' Cases lists.

4.3.1.1 Assign a Step to another Actor

When the User is presented with a Step that requires his/her input, s/he can **Assign** that Step to another Actor in the assigned group:

4.3.1.2 Unassign a Step (take back from another Actor)

The User can use the **Unassign** function to undo an **Assign to**. Once the User has left the inbox, any Steps assigned (i.e., forwarded) to other Users will no longer appear.

4.3.1.3 Suspend a Step

Instead of completing a Step in the inbox, a User can Suspend it, which puts it temporarily on hold. It will show in the inbox with an orange bullet.

4.3.1.4 Resume a Step

A User can resume a Suspended Step in the Inbox by clicking on it, then selecting Resume. The Case will resume, and the next Step will show up in the inbox of the Actor it is assigned to.

4.3.1.5 Change a Step Priority

A User can change the priority assigned to a step by selecting a new priority from the dropdown menu.

4.3.2 How to Manage Cases

The Inbox shows the individual Case (instantiation) histories of deployed Processes.




Figure 142. Managing Cases in Bonita User Experience

As in an email interface, Cases can be

- **Labelled**
- **Starred** or **unstarred**

For each Case you will see:

Item	Example /explanation
checkbox	select or deselect the Case
star	to mark the case for sorting
label	Shows inbox, and the category labels defined in the Process, (see Define Details for a Pool), by the Admin (see Define Categories), or by this User (see Define and apply labels to cases.)
*Name of Process followed by - #N (N)	name of Process followed by case number
a colored bullet: green, orange, or grey	green a Step is to be completed orange a Step has been suspended grey all Steps have been completed
human icon (single or multiple)	For the Actor to whom the current Inbox belongs; shows up when he/she has an action to take
priority chevron	— normal ^ high ^^ urgent
Name of Step	name of Step
date and time stamp	date and time the Case was initiated
new window icon 	for an incomplete Step requiring input, open in a new window in the proper Form assigned to it

*This information can be customized. See [Customize Process label in Inbox](#).

4.3.2.1 Define and apply Labels to Cases (User)

The end User can create Labels for Cases, and customize them with names and colors.

From the left-hand column of the User Experience window, select **More -> new label**.

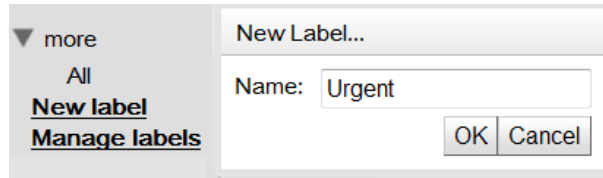


Figure 143. Apply a Label to a Case

To add a color, click on the label you've just created and select a style for the label.

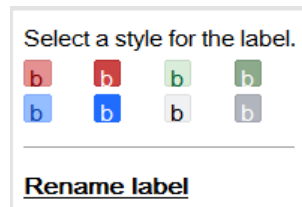


Figure 144. Change the characteristics of a Label

To apply Labels to the Cases, select the Cases and then select **Labels -> Apply** from the task bar at the top of the inbox.

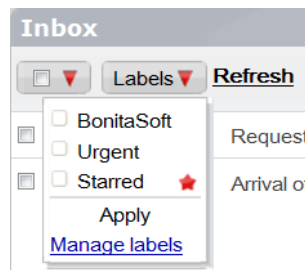


Figure 145. Apply a Label to Cases

4.3.2.2 Hide or unhide Labels in the User Experience Control Panel

You can change the status of any of the Labels so they are either shown or not shown in the Control Panel.

From the left-hand column of the User Experience window, select **More -> Manage labels**.

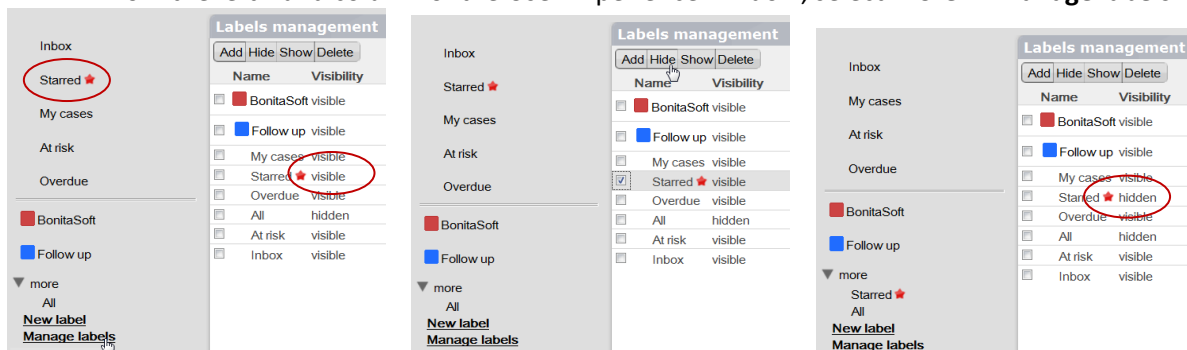


Figure 146. Manage Labels

4.3.2.3 View Case overview

When a Case is selected, it shows a thread of previous actions taken. This is useful where the decision made may depend on other decisions taken before.

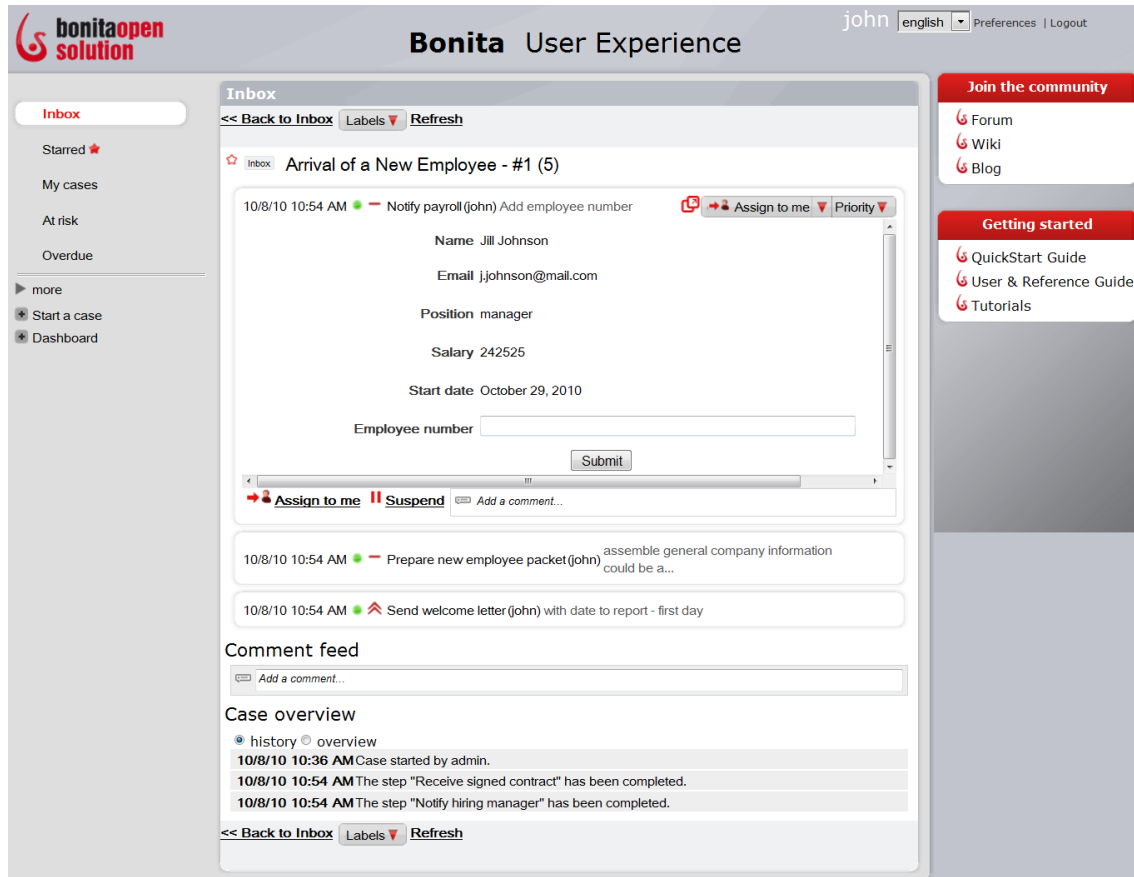


Figure 147. See a Case history in Bonita User Experience

The User can add comments or note to a Case here. Click **Add a comment** and enter. (The comment is not editable once you hit *Save*). All comments entered by all Actors in this Case will be visible here.

History shows a condensed series of Steps. The default Step Summary is “The Step [Step name] has been completed.” This can be changed to something more descriptive, and can include dynamic content, in Bonita Studio (as defined in [General details for a Step](#)).

Overview shows the Overview Form designed in Bonita Studio, with data as of the current state of the Process Case.

If a Step has been configured in the Process design so that its form is to be skipped, a message will appear in the Step history to inform the User of this (instead of a blank Step).

4.3.3 Consult workload status (Dashboard)

To see the full Dashboard, go to **Dashboard** in the Control Panel, expand it, and select **More**.

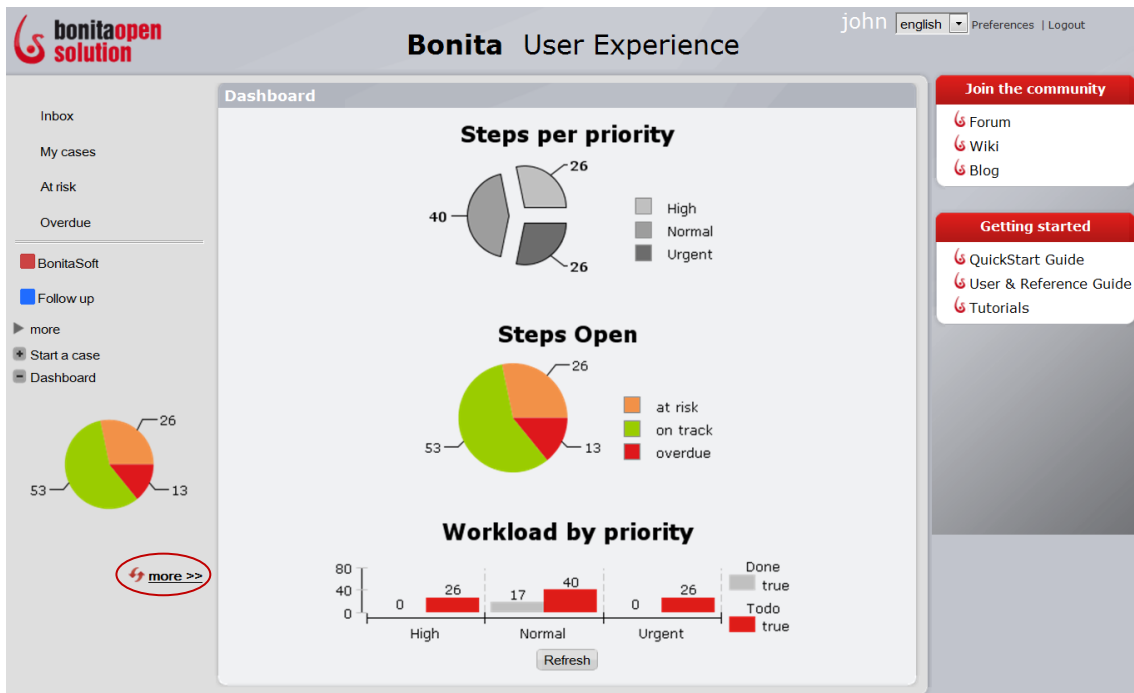


Figure 148. Dashboard shows the status of User's workload

The Dashboard gives a snapshot of the current status of this User's workload.

- Priorities of all Steps
- Open Steps categorized by
 - At risk
 - On track
 - Overdue
- Open and completed Steps categorized by priority

4.3.3.1 Change Dashboard graph

The User can change this Dashboard report. Go to the upper right and select **Preferences -> Reporting** to select a different graph type.

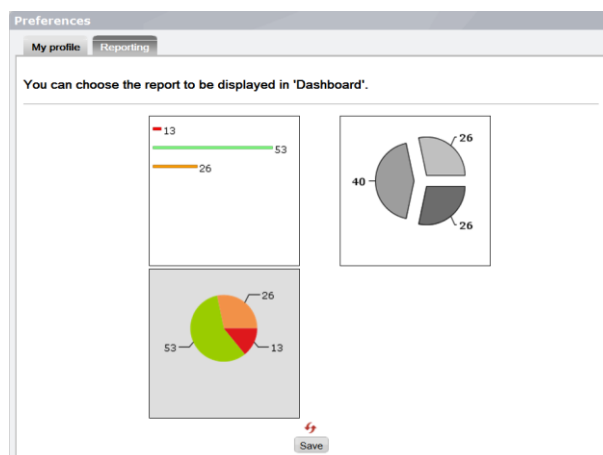


Figure 149. Change Dashboard graph

4.3.4 Change User Profile

The User can change some of his/her User Profile. Go to the upper right and select **Preferences -> My Profile**. (The only editable fields are the password and password confirmation.)

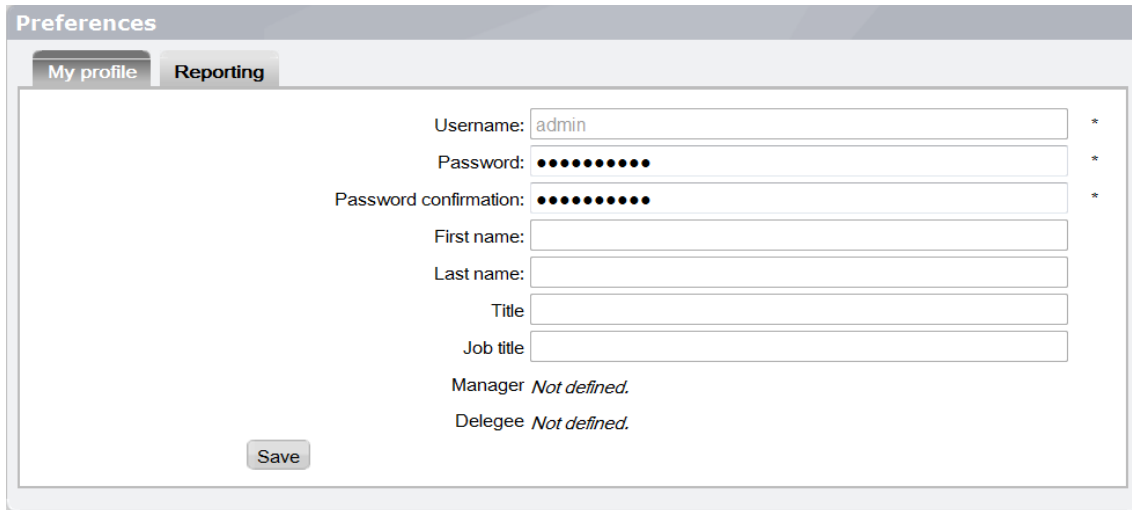


Figure 150. Change Profile data

4.3.5 Start a new Case

The User can start a new Case of a Process. Go to the Control Panel and click on **Start a case** to show the Processes available to this User. Click on the Process to start a Case..

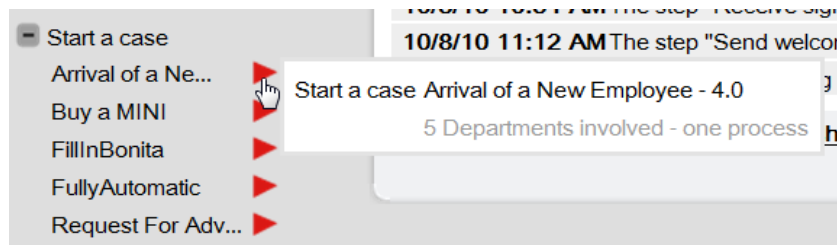


Figure 151. Start a new Case of a Process

4.4 Bonita User Experience – Administrator’s View

The User Experience assigned to the Administrator permits him/her to monitor and interact with Processes deployed by the developer. This section describes the User Experience functions that are unique to the Administrator.

Note that all of the functions in [Bonita User Experience – End User’s View](#) are also available to the administrator.

In the User Experience of the Admin, there is an “Admin” section in the Control Panel of the User Experience that allows the Admin to manipulate:

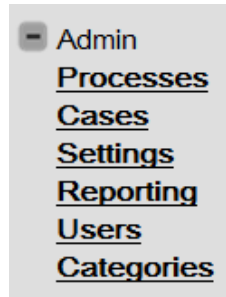


Figure 152. Admin menu in Control Panel

4.4.1 Manage Processes as Administrator

Select Processes from the Admin Menu in the Control Panel.

The **List of all processes** shows all Processes (and subprocesses) that have been run (deployed) by the developer and accessible to this administrator.

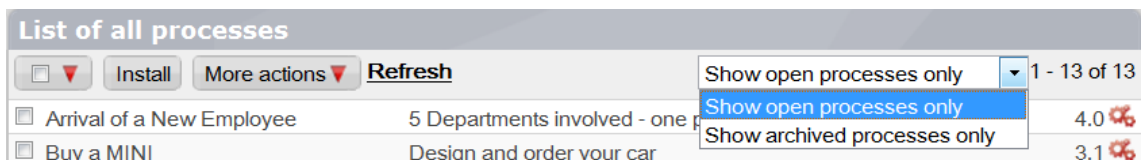


Figure 153. See the List of all Processes

Here you can select all, some or none of the Cases using the checkbox and:

- **Install** Any Process created in Bonita Studio exported as a *.bar file can be installed directly into the Bonita User Experience. Click **Install** and Browse to upload the Process.
- **More Actions** From here, you can:
 - **Open design** Show the process map for the selected Processes
 - **Delete all cases** All the implemented Cases of the selected Process(es) will disappear systemwide (from the Admin and the Users' boxes)
 - **Disable** This disables a Process but does not delete it
 - **Enable** To use a disabled Process again, **enable** it
 - **Remove** This deletes a Process and all Cases associated with it
 - **Archive** This archives the Process; it is no longer active or accessible but a record is kept

You can choose to show open Processes or archived Processes only.

The version number and state of each Process is shown at the right of each Process.

4.4.2 Manage Cases as Administrator

The **List of all cases** shows all Cases that have been implemented for your Processes.

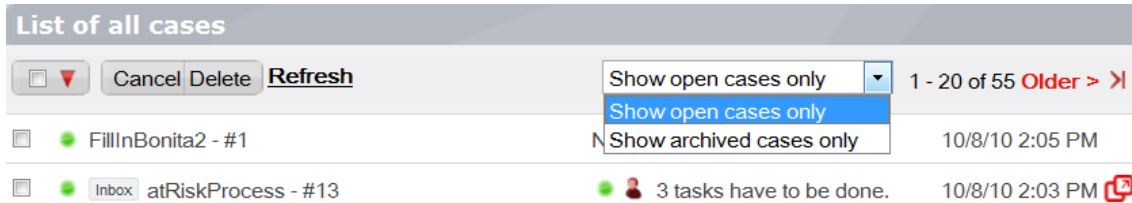


Figure 154. Manage cases in Cases list

Here you can select all, some or none of the Cases using the checkbox and:

- **Cancel** Cancels this Case and Archives it
- **Delete** Cancels the Case and deletes it entirely

You can choose to show open Processes or archived Processes only.

The time stamp and “Open the current Form in a Web app/web page” icon is shown at the right of each Case.

For each line (Case) there is a set of icons/actions similar to those in the Inbox. (See [How to manage a Step.](#)) However you’ll see a colored bullet with slightly different options.

a colored bullet: green, orange, blue, or grey	<p>green a Step is to be completed</p> <p>orange a Step has been suspended</p> <p>blue a Step is executing</p> <p>grey all Steps have been completed</p>
--	--

4.4.2.1 See Process graphically from inside a Case

You can see the Process diagram from the Case history. Click on a Case in the **List of all Cases**, then select **More actions -> Open design**.

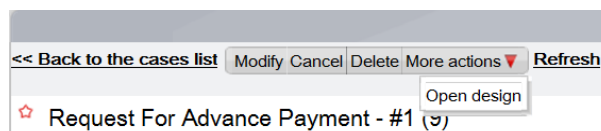


Figure 155. See Process diagram

4.4.3 Configure Global Settings of User Experience

4.4.3.1 Show or hide User Experience Settings & Calculate Steps at Risk

By Selecting **Admin -> Settings -> General**, you can:

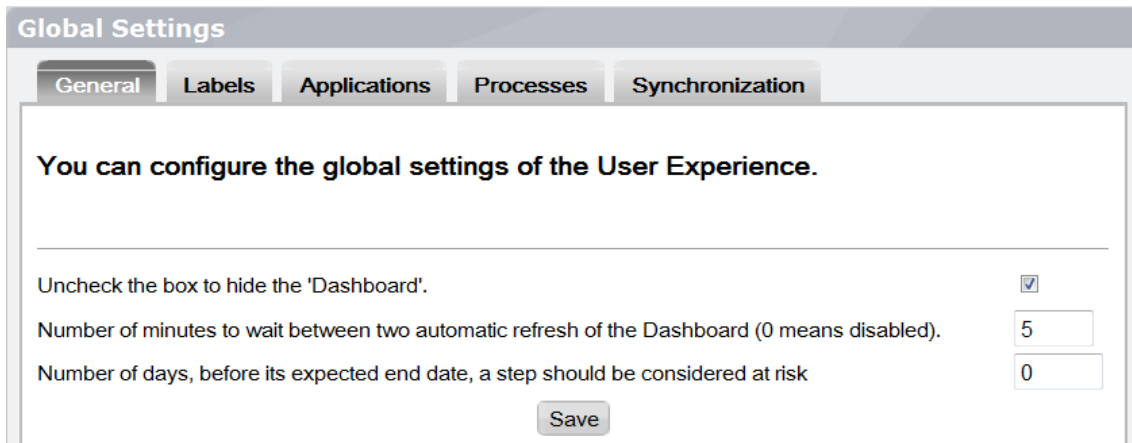


Figure 156. Change general settings in Bonita User Experience

- hide or show the Dashboard on the Control Panel
- Change the refresh rate for the Dashboard (If set to 0, you can refresh manually)
- Calculate when Steps should be considered “at risk.” (The estimated execution time can be specified for each Step in Bonita Studio by the Process designer.)

4.4.3.2 Configure how Users can use Stars and Labels

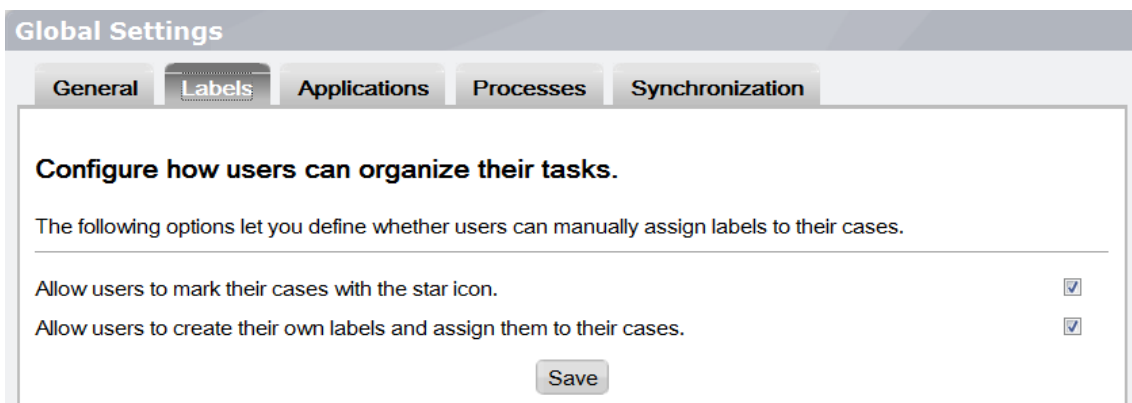


Figure 157. Change Label and Star settings in Bonita User Experience

Here the Admin can allow the Users inside a Process to use Star and create labels, or not.

4.4.3.3 Specify the URL for a custom web application

Go to **Admin -> Settings -> Applications** to change the Web application for your Process. Use the pointer here to identify the location of your *.war files by URL.

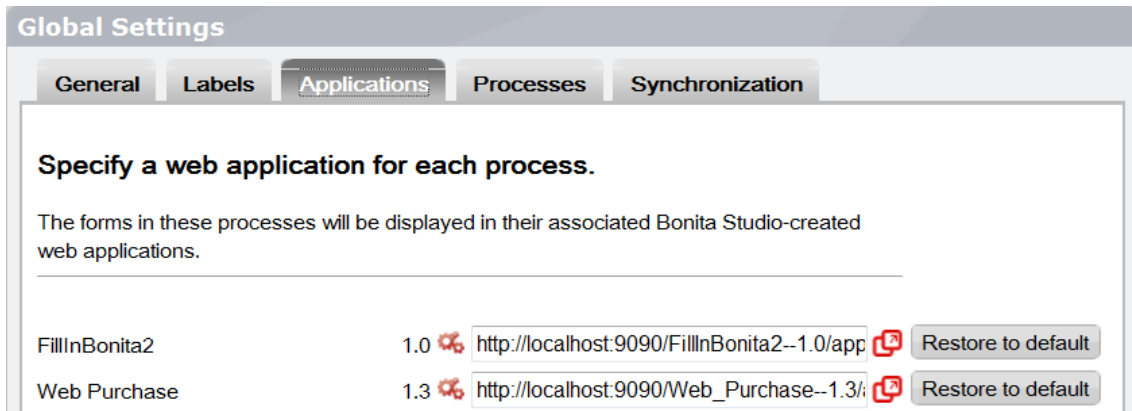


Figure 158. Specify the URL for your custom web application

4.4.3.4 Customize Process label in Inbox

You can change how the names and numbers of Cases are presented in your inbox (see [How to Manage Cases](#)).

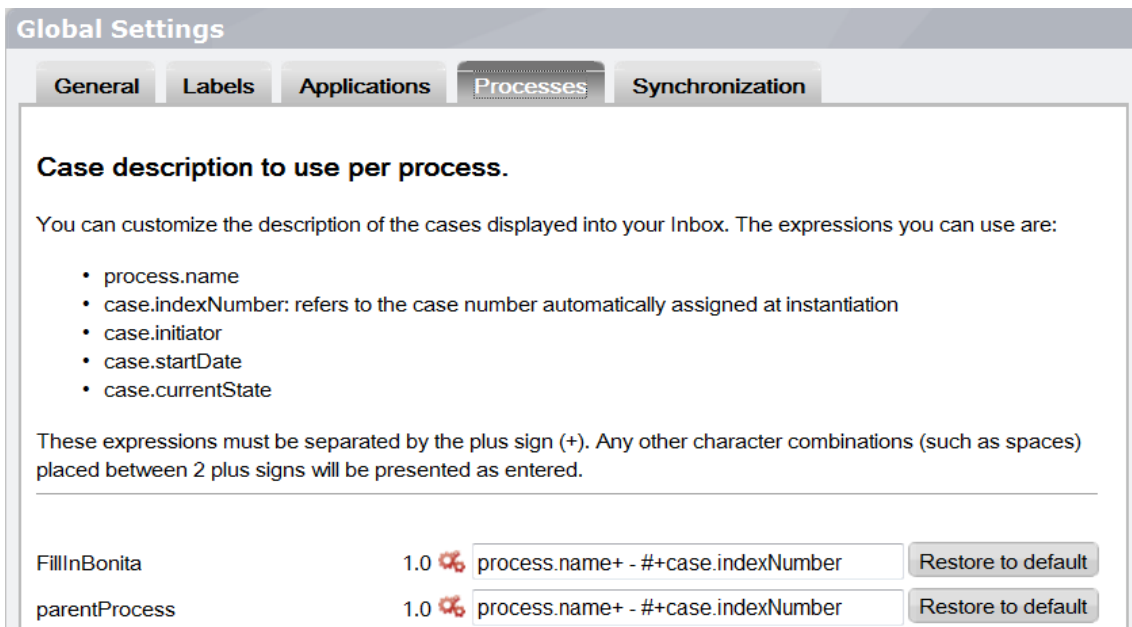


Figure 159. Change how names and numbers of Cases are presented

Enter the items you want to see presented in the order you want them, separated by a + . Character strings between two plus signs (example: + string of characters +) will be presented as entered.

4.4.3.5 Fix the count of cases (synchronization)

If the tally of cases is incorrect, you can instruct the Bonita Execution Engine to recalculate – however, this action is not recommended as it may significantly degrade the performance of the engine.

4.4.4 Consult status of all Steps, Cases, and Processes

The **Reporting** function shows an overview of all Process Cases run and running, incorporating data from all Users.

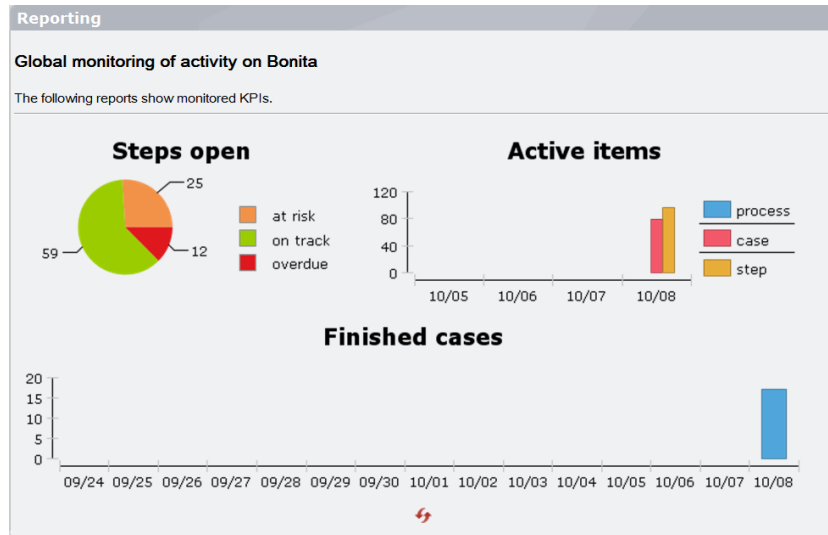


Figure 160. Reporting shows status of Cases

4.4.5 Define Users and User Roles

To define the Administrator and Users for your Processes, create user accounts in **Users**:

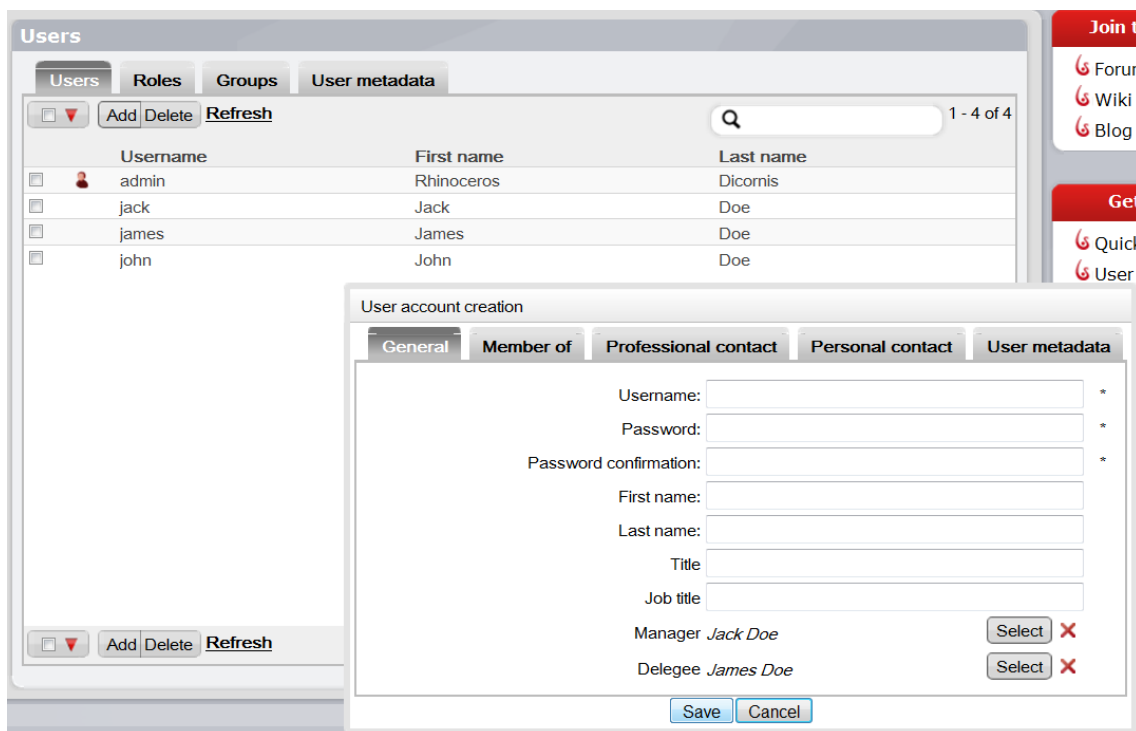


Figure 161. Define Users

User Account Creation comes up for **Add**. Use User Account Creation to define:

- **General** Name, password and other information as shown.
- **Member of** choose a group and a role for this User.
- **Professional and Personal contact information** these fields are predefined, but you can add more fields all for User Accounts in [Define User Metadata](#).
- **User metadata** this sections contains any additional fields you have defined. See [Define User Metadata](#).

4.4.5.1 Define Users and User Roles

To define the Administrator and Users Roles for your Processes:

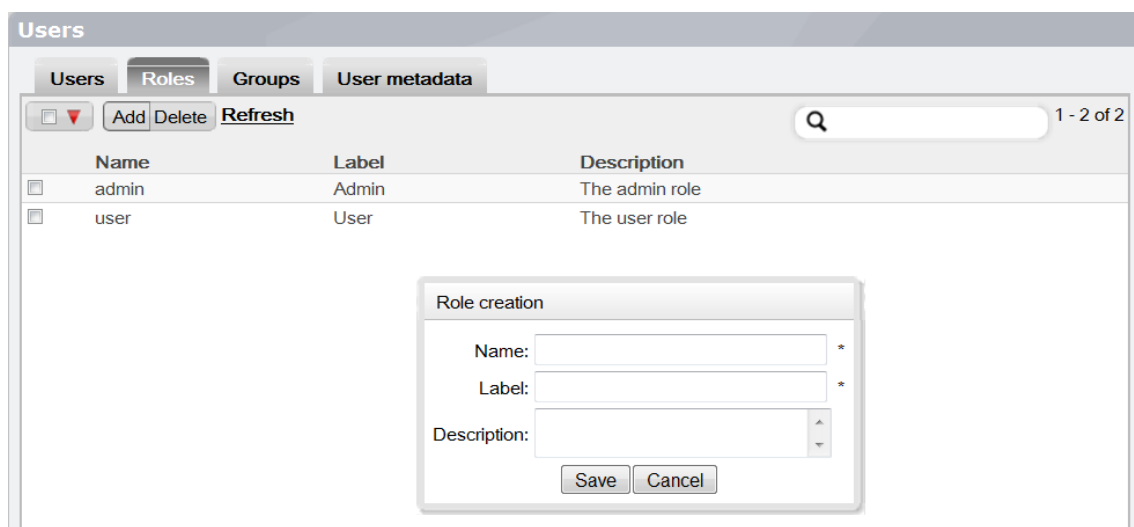


Figure 162. Define User Roles

4.4.5.2 Define Groups

To define User Groups for your Processes:

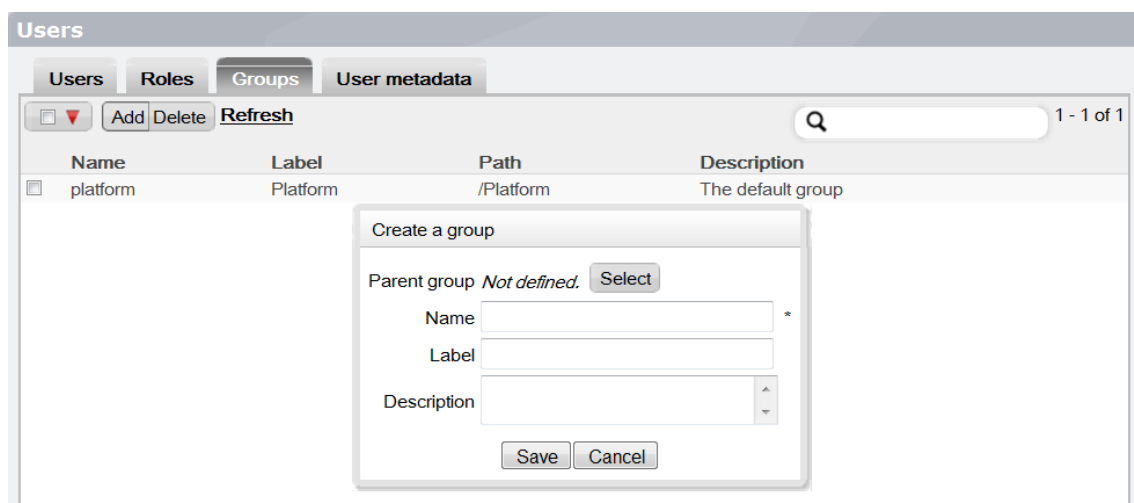


Figure 163. Create a new Group

4.4.5.3 Define User Metadata

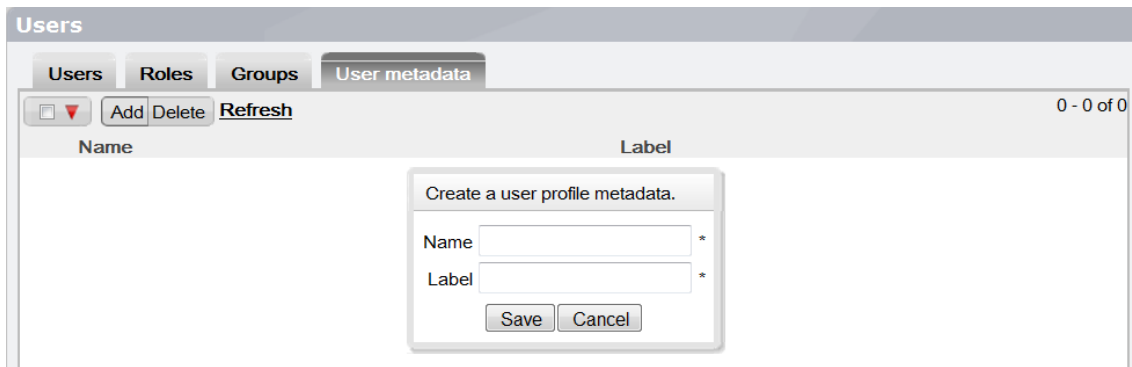


Figure 164. Define User metadata

If you want to add more fields to your User’s account, go to Users -> User metadata. The fields you add will be available in all User accounts.

4.4.5.4 Filter Users, Roles, or Groups

You can apply a filter to the list of Users, Roles, or Groups so that only matches are shown. Note that short common words such as “the” and “a” are ignored, so they cannot be used to filter the list.

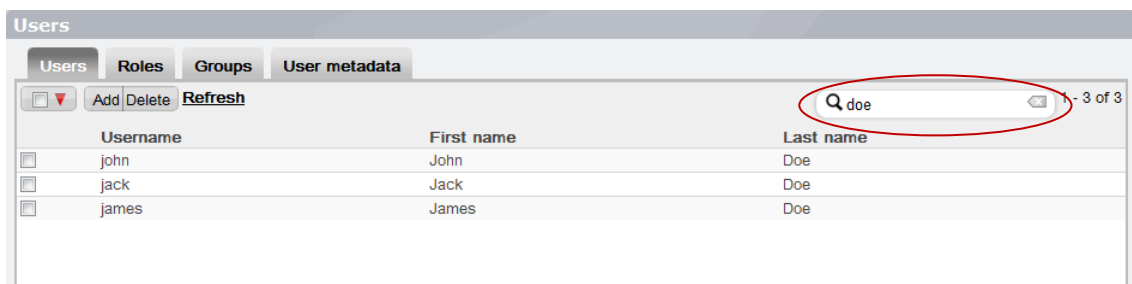
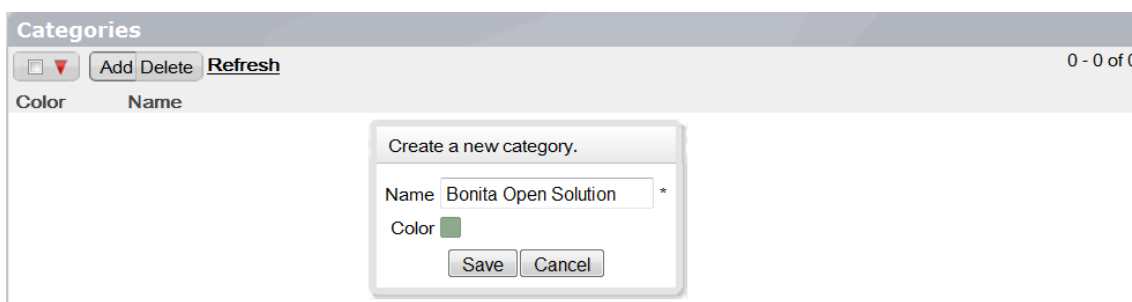


Figure 165. Apply a filter to list a subset

4.4.6 Define Categories

Categories are labels defined by the Process designer (see [How to Define Categories](#)) or the Admin and applied to all User’s inboxes. To define Categories from the User Experience as Admin, go to Categories in the Admin tool in the Control Panel.



4.5 Externally deployed *.war files – End User's View

End user forms can be also be presented in a dedicated web application, designed and developed in a Web application archive (*.war file) external to Bonita User Experience.

Part 5. Bonita Open Solution Hardware, Software and Configuration Requirements

5.1 Hardware Required

A 1GHz processor is recommended, with a minimum of 512 Mb of RAM. Windows users can avoid swap file adjustments and get improved performance by using 1 Gb or more of RAM.

5.2 Operating Systems, Databases, Application Servers

Bonita Open Solution has been successfully tested in the following environments. (It should work in others, but only these have been tested in Bonita Open Solution continuous integration).

Operating Systems

- GNU/Linux
- Windows XP
- Mac OS X
- Java Virtual Machines 1.5 and 1.6

Relational Databases

- MySql 5.1
- MS SQL Server 2005
- Postgresql 8.3
- Oracle 10g
- H2 1.2.132

Application Servers

- Tomcat 6
- JOnAS 4.10.3
- JBoss-4.2.2.GA
- JOnAS 5.1.1
- JBoss-5.1.0.GA

Bonita Open Solution requires Apache Ant 1.7 or higher. Apache Ant will allow users to manage configuration and administration operations. It can be downloaded from <http://ant.apache.org>.

5.3 Subdirectory files

When you download Bonita All-in-One, a new runtime directory will be created. To access this directory, go to Bonita Studio, **Process -> Export application** and then check **Export runtime**. In the Runtime subdirectory you will find:

build.properties	This file contains the J2EE properties required to deploy and use Bonita Open Solution deployed on a remote J2EE server (JOnAS and Jboss properties are provided to allow remote execution of Bonita Open Solution.)
build.xml	This file is an "ant" file (aka makefile) that provides tasks for administration operations
commons.xml	
about_files/	This directory contains information about software licenses for software incorporated into / distributed with Bonita Open Solution.
conf/	This directory contains default configuration files for Bonita Open Solution, including: <ul style="list-style-type: none"> • bonita-environment xml file (with services and objects used as default by the Bonita Execution Engine), • login module configurations (JAAS compliant login modules samples) • hibernate persistence configuration (default implementation to handle Bonita Open Solution persistence)
ear or war	<ul style="list-style-type: none"> • Deployment descriptors for JEE application server / description for ReST server
lib/	This directory contains the libraries used in Bonita Open Solution. BOS can be integrated into your application/IS in several ways (ie, integrated into a web application, inside a rich client application, remotely deployed in a JEE application server, etc). The libraries required by your system will depend on your integration environment.

Part 6. How to install Bonita Open Solution

Bonita Open Solution can be installed and used in Standard and Enterprise environments. (Note that wherever this document references server descriptors, that includes the server library+ all its dependencies; client descriptors similarly refer to client library + all its dependencies.

6.1 Standard installation (Bonita Open Solution as a library)

To embed Bonita in your java application, add the libraries located under `/lib/server` directory in your application classpath. The main library required by Bonita Open Solution is:

```
bonita-server-<version>.jar
```

6.2 Enterprise installation (Bonita Open Solution as a server)

Enterprise installation is intended for BPM deployments in which Bonita Open Solution will be deployed on a dedicated server, allowing different applications to reach BOS remotely. All client applications require a single BOS library:

```
bonita-client-<version>.jar
```

In this configuration, the Bonita Open Solution server will often be deployed in an application server.

If you are using JBoss, JOnAS, WebLogic, or GlassFish application servers, you can directly deploy Bonita Open Source examples. Specific descriptors and classpath configurations for those servers are included in this package. The `bonita.ear` file generated through `ant ear.ejb3` can be deployed in any EJB2 or EJB3 compliant application server; standard descriptors should work in any EJB2 or EJB3 environment.

To install JBoss, see [JBoss installation and deployment](#), below.

To install Jonas, see [JOnAS installation and deployment](#), below.

If you are using another JEE application server (such as Websphere, Oracle, Geronimo), you will need to add specific descriptors for your application server into the `bonita.ear` file and configure your client side to access the Bonita APIs.

In this section we will look at configuring Bonita Open Solution for JBoss. (To see this example, see the video tutorial [How to Integrate Bonita into JBoss.](#))

6.2.1 Server side

First, on the server side, install the Bonita Runtime component in the JEE application server. In this example, use `jboss` as the name of the JBoss JEE application server installation folder.

- Get the runtime distribution from **Bonita Studio -> Process -> Export Application.**

- Generate a `bonita.ear`.
 - For JBoss (and for JOnAS, WebLogic, and GlassFish), use the provided Apache Ant task. Call the tasks `ant ear.ejb2` and `ant ear.ejb3`, respectively, to generate the `bonita.ear` file corresponding to either the JEE 1.4 or 1.5 specifications. Deploy this file into your JEE 1.4 or 1.5 application server.
 - To deploy Bonita Runtime on another application server:
 - Copy the necessary deployment descriptors in `<path to bonita runtime>/ear/<ejb2 or ejb3>`.
 - Use `ant -f: <path to bonita runtime>/build.xml ear.<ejb2 or ejb3>`
- 2) Copy the `bonita.ear` file into your JEE server deployment directory (e.g., `jboss/server/default/deploy`)
- Add BonitaAuth and BonitaStore login modules to the JAAS configuration for your JEE server:
 - `org.ow2.bonita.identity.auth.BonitaIdentityLoginModule`
 - `org.ow2.bonita.identity.auth.BonitaRemoteLoginModule` (must be stacked with your JEE JAAS propagation login module)
 - edit `jboss/server/default/conf/login-config.xml` to add:

```
<application-policy name="BonitaAuth">
  <authentication>
    <login-module
code="org.ow2.bonita.identity.auth.BonitaIdentityLoginModule"
flag="required"/>
  </authentication>
</application-policy>
<application-policy name="BonitaStore">
  <authentication>
    <login-module
code="org.ow2.bonita.identity.auth.BonitaRemoteLoginModule"
flag="required"/>
    <login-module
code="org.jboss.security.ClientLoginModule" flag="required">
      <module-option name="password-
stacking">useFirstPass</module-option>
    </login-module>
  </authentication>
</application-policy>
```

- Start the server.

6.2.2 Client side

Then, on the client side, configure each Bonita Runtime client (e.g. your web applications) to access the Bonita Runtime component in the JEE application server.

- Install the Bonita Web applications (which you can get from **Bonita Studio -> Process -> Export Application**) in your web container. In this example, `jboss/server/default/deploy`.
- Define properties to access the Bonita Runtime installed on your JEE server:

```
-Djava.naming.factory.initial=<your factory> (eg.
org.jnp.interfaces.NamingContextFactory)
-Djava.naming.provider.url=<your URL> (i.e.
jnp://localhost:1099)
-Dorg.ow2.bonita.api-type=<EJB2 or EJB3>
```

6.2.3 Jboss 4.x and 5.x installation and deployment

To run Bonita Open Solution in JBoss applications:

- Download JBoss 4.x or 5.x from www.jboss.org and follow the JBoss application server installation instructions.
- Edit the `build.properties` file to set your Jboss configuration settings:
 - URL provider (default: localhost, with port 1099)
 - set `jboss.home` and `jboss.client` property values (these properties must be initialized with the path corresponding to the directory in which Jboss was installed and the path in which the JBoss client libraries are available).
- Generate the `bonita.ear` file (see [Server side](#), above).
- Copy the `bonita.ear` file generated into the designated directory (see [Server side](#), above).
- Start the Jboss application server by executing `run.bat` or `run.sh` in the `jboss/bin` directory

Bonita Open Solution should now run under Jboss.

6.2.4 JOnAS 4.x and 5.x installation and deployment

To run Bonita Open Solution in JOnAS applications:

- Download JOnAS 4.x or 5.x from <http://jonas.ow2.org> and follow the JOnAS application server installation instructions.
- Edit the `build.properties` file to set your JOnAS configuration settings:
 - URL provider (default: localhost with port 1099) and set `jonas.root` and `jonas.lib` properties values
 - initialize these properties with the path corresponding to the directory in which JOnAS was installed and the path in which JOnAS client libraries are available.
- Type `ant ear.ejb2` or `ant ear.ejb3` (depending on the JOnAS version you want to use) under your Bonita installation directory to generate the `bonita.ear` file
- Copy the `bonita.ear` file generated into `JONAS_ROOT/apps/autoload` directory (default configuration)
- Start the JOnAS application server by executing `jonas start.bat` or `jonas start.sh` in the `JONAS_ROOT/bin` directory.

Bonita Open Solution should now run under JOnAS.

6.3 ReST Installation

REST installation is intended for deployments in which Bonita Open Solution will be deployed on a dedicated web application server, allowing different applications to reach BOS remotely via standard HTTP. All client applications require a single BOS library:

```
bonita-client-<version>.jar
```

In this configuration, the Bonita Open Solution server will often be deployed in an application server.

If you are using Tomcat 5.x or Tomcat 6.x application servers, you can directly deploy Bonita Open Source examples. Specific descriptors and classpath configurations for those servers are included in this package. The `bonita-server-rest.war` file generated through ant war can be deployed in the application server.

6.3.1 Server side

On the server side, install the Bonita Runtime component in the application server (Tomcat 5.x, Tomcat 6.x).

- Get the runtime distribution from Bonita **Studio -> Process -> Export Application**.
- Generate a `bonita-server-rest.war`, or use the provided Apache Ant task. Call the task `ant war` to generate the `bonita-server-rest.war` file. Deploy this file into your application server.
- Start the server.

NOTE: by default the ReST server will ask for authentication using HTTP Basic authentication. This means that to access a ReST resource, you need to authenticate using a user name and a password from the data base. However, the authentication requires additional verifications for each request and this may decrease the performance. In this case, if the resources will not be exposed on the Web, you can deactivate the authentication by setting property `org.ow2.bonita.activate-rest-authentication` as `false`. However, remember that all users who have access to the host where the bonita war is deployed will be able to access all ReST resources.

6.3.2 Client side

On the client side, configure each Bonita Runtime client (e.g. your web applications) to access the Bonita Runtime component in the application server.

- Install the Bonita Web applications.
- Add BonitaREST to the JAAS configuration:
`org.ow2.bonita.identity.auth.BonitaRESTLoginModule`
- Define properties to access the Bonita Runtime installed on your server:

```
-Dorg.ow2.bonita.rest-server-address =<your URL> (i.e.  
http://localhost:8080/bonita/)  
-Dorg.ow2.bonita.api-type=REST
```

6.4 Environment switch through APIs

If the Process has been developed in a local environment (e.g. on a Tomcat server), it can be switched to a production environment by changing the system properties. For example, on the client side, you can set the API property type:

```
-Dorg.ow2.bonita.api-type=<Standard or EJB2 or EJB3>
```

6.4.1 Using the ReST API

To use the ReST API in Java, configure the Runtime (see the series [Building your BPM applications with Bonita Runtime](#)):

- Set the Bonita API type access to REST: `org.ow2.bonita.api-type=REST`
- Include in the JAAS configuration file:
`org.ow2.bonita.identity.auth.BonitaRESTLoginModule`
- Set a new property on the client side to give the server address using
`org.ow2.bonita.rest-server-address` (for example,
`org.ow2.bonita.rest-server-address=http://localhost:8080/bonita/`)
- Set a new boolean property on the server side to allow to activate or deactivate the user's authentication: `org.ow2.bonita.activate-rest-authentication`. By default, the server will ask for a user authentication.

If you are using a language other than Java, you can access the ReST API using HTTP POST. NOTE the warning about user authentication in [Server Side configuration](#) for ReST.

6.5 Configuring logs in Bonita Runtime

Bonita Runtime uses standard JDK logging APIs
(<http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/LogManager.html>).

You can use your own logging.properties file by defining the following system property:

```
-> -Djava.util.logging.config.file=<path to your logging.properties file>
```

Part 7 How to configure Bonita Open Solution for a production environment

7.1 How to configure a database

Bonita Open Solution comes with a default configuration for data persistence in an H2 Database. To use another database in a production environment, both Hibernate configuration files for the core and for the history database should be modified and a JDBC driver should be provided to the application server for the target database.

The location of the Hibernate configuration files is declared in the Bonita environment configuration file, which is provided to the application server by setting its location in the property `org.ow2.bonita.environment` in the `JAVA_OPTS` environment variable.

Here is the default configuration:

```
<hibernate-configuration name='hibernate-configuration:core' >
<properties resource='hibernate-core.properties' />
...
<hibernate-configuration name='hibernate-configuration:history' >
<properties resource='hibernate-history.properties' />
...
```

With the above syntax, the Hibernate files should be present in the classpath as they are declared as resources. To place them somewhere else, they should be declared as files (be sure to double the file separators on windows, i.e. "\\") :

```
<hibernate-configuration name='hibernate-configuration:core' >
<properties file='/usr/tmp/hibernate-core.properties' />
...
<hibernate-configuration name='hibernate-configuration:history' >
<properties file='/usr/tmp/hibernate-history.properties' />
...
```

The location of the large data repository where deployment and attachment files are stored can also be configured:

```
<large-data-repository name='large-data-repository'
class='org.ow2.bonita.services.impl.FileLargeDataRepository'>
<arg><string value='property:java.io.tmpdir' /></arg>
</large-data-repository>
```

Note that if you do not set the **org.ow2.bonita.environment** property with the environment file location, the default environment and Hibernate files present in `bonita-server.jar` will be used.

7.2 How to initialize a database

Be sure that both core and history databases have been created first with the appropriate user privileges.

You can then initialize the database schema for the core and the history databases using the `init-db` tool. To run it, go to Bonita Studio, **Process -> Export Application** and select **Export Runtime**.

In the conf subdirectory, modify `bonita-environment.xml`, `hibernate-core.properties` and `hibernate-history.properties` to fit your database configuration as described in **How to configure a database**, above.

Place the JDBC driver for your database vendor and version in the server subdirectory.

Run the ant task `init-db` using the following command in the runtime directory.

```
ant init-db hibernate-configuration:core hibernate-
configuration:history
```

7.3 How to configure security

Bonita Open Solution comes with a default JAAS (Java Authentication and Authorization Service) configuration. This configuration is provided to the application server by setting the property `java.security.auth.login.config` to JAAS configuration file path in `JAVA_OPTS` environment variable.

This configuration comprises two `LoginContexts`: `BonitaAuth` and `BonitaStore`. These `LoginContexts` are mandatory if you want to use the Bonita User Experience and Bonita forms. Otherwise (if you want to use only the engine), name your `LoginContext` as you wish as you will be the one providing the source code for the login operations.

- `BonitaAuth` is used by the User Experience when a User tries to log in, to verify that the credentials he provided are correct.
- `BonitaStore` is used every time the User Experience calls the engine, to supply it with the logged-in user's identity.

7.4 How to configure authentication (BonitaAuth LoginContext)

In the default configuration, `BonitaAuth` consists of only one `LoginModule` :

```
BonitaAuth {  
    org.ow2.bonita.identity.auth.BonitaIdentityLoginModule required;  
};
```

`BonitaIdentityLoginModule` performs the authentication of the users using the authentication service configured in Bonita environment configuration file:

```
<authentication-service name='authentication-service'  
class='org.ow2.bonita.services.impl.DbAuthentication'>  
  <arg><string value='bonita-session:core' /></arg>  
</authentication-service>
```

The default implementation of this service (`DbAuthentication`) is provided for a development environment only and should not be used in production as it goes by the engine database to verify the user credentials.

In production, as you need the authentication to be performed against your own users referential, you need to provide your own authentication service to replace `DbAuthentication` in the engine environment file. You can also provide your own JAAS `LoginModule` to replace `BonitaIdentityLoginModule` in JAAS configuration file.

To provide your own authentication service, implement the interface `org.ow2.bonita.services.AuthenticationService`

```
public interface AuthenticationService {
    /**
     * Check whether a user has admin privileges or not
     * @param username the user's username
     * @return true if the user has admin privileges, false otherwise
     * @throws UserNotFoundException
     */
    boolean isUserAdmin(String username) throws UserNotFoundException;

    /**
     * Check some user's credentials
     * @param username the user's username
     * @param password the user's password
     * @return true if the credentials are valid, false otherwise
     */
    boolean checkUserCredentials(String username, String password);
}
```

Note that if you provide your own JAAS LoginModule you still need to implement the authentication service, as the `isUserAdmin` method will be used to check whether the user is permitted access to the User Experience administration features.

7.5 How to configure Login (BonitaStore LoginContext)

In the default configuration, BonitaStore consists of two LoginModules for a deployment in a JEE application server and only one LoginModule for a deployment in a simple servlet container (Tomcat, Jetty).

For example, here is the BonitaStore LoginContext for a JBoss JAAS configuration :

```
BonitaStore {
    org.ow2.bonita.identity.auth.BonitaRemoteLoginModule required;
    org.jboss.security.ClientLoginModule required password-
stacking=useFirstPass;
};
```

The first LoginModule (`BonitaRemoteLoginModule`) is used to retrieve the credentials of the user and save them in the context shared between the LoginModules stacked in the LoginContext.

The second LoginModule (`ClientLoginModule`) is used to propagate the identity of the user to the application server. This LoginModule is specific and provided by the application server.

In the BonitaStore LoginContext of the provided JAAS configuration for simple servlet containers, both operations are performed by one single LoginModule (`LocalStorageLoginModule`).

7.6 How to configure multi-tenancy

The Bonita Execution Engine offers full multi tenancy support. Data isolation is established physically: data are isolated in different database schemas. Data from one tenant are isolated in a database schema and are not shared with data from another tenant.

In a multi tenancy implementation the Bonita Execution Engine accepts a different configuration according to each tenant. An engine configuration wraps the database, the repository properties, etc.

It is not necessary to duplicate execution engines. One engine can work with multiple configurations and thus multiple tenants. Each tenant will work with its own database, repository, etc.

7.6.1 Runtime configuration

Create a set of config files for each tenant:

- bonita-environment-tenantId.xml
- bonita-core-tenantId.properties
- bonita-history-tenantId.properties

Be sure to set a different large data repository for each tenant. (See [How to configure a database.](#))

Create a file bonita-tenants.properties:

- tenantId1 = <path of bonita-environment of tenant1>
- tenantId2 = <path of bonita-environment of tenant2>

Modify environment variable `org.ow2.bonita.environment` to point to the `bonita-tenants.properties` file.

7.6.2 Client configuration

Add the parameter “domain” to the JAAS context login modules and duplicate the login context for each tenant. This will give you the possibility to configure a different authentication mechanism for each tenant. For example:

```
BonitaAuth-tenantId1 {
    org.ow2.bonita.identity.auth.BonitaIdentityLoginModule
required domain=tenantId1;
};
BonitaAuth-tenantId2 {
    org.ow2.bonita.identity.auth.BonitaIdentityLoginModule
required domain=tenantId2;
};

BonitaStore-tenantId1 {
    org.ow2.bonita.identity.auth.LocalStorageLoginModule
required domain=tenantId1;
};
BonitaStore-tenantId2 {
    org.ow2.bonita.identity.auth.LocalStorageLoginModule
required domain=tenantId2;
};
```

When you log into the engine, you will need to specify the tenant you want to log in to by using the right login context. For example :

```
LoginContext loginContext = new LoginContext("BonitaAuth-
tenantId1",
                                new SimpleCallbackHandler(username,
password);
    loginContext.login();
    loginContext.logout();
    loginContext = new LoginContext("BonitaStore-
tenantId1",
                                new SimpleCallbackHandler(username,
password);
    loginContext.login();
    loginContext.logout();
```

When using the Bonita Open Solution web interface (User Experience and Process Web Applications), respect the following patterns for the JAAS login contexts:

```
BonitaAuth{
    org.ow2.bonita.identity.auth.BonitaIdentityLoginModule
required domain=<tenantId>;
};
BonitaStore{
    org.ow2.bonita.identity.auth.LocalStorageLoginModule
required domain=<tenantId>;
};
```

Warning: There is no java policy to control third-party execution code.

Part 8. How to analyze a problem in Bonita Open Solution

Your Bonita Open Solution log files (for Bonita Studio and Bonita Execution Engine) are available via the Menu Bar: **Help -> Show log** and **Help -> Show engine log**.

When you encounter a problem, please post a description of the problem and a copy of your log file on the Bonita Community Forum at www.bonitasoft.org/forum/.

BonitaSoft developers, among others, actively contribute to the Bonita Community and will post a response.