

Chapter 5

JSystem Automation Framework GUI Interface



➤ In this chapter...

<i>JSystem Automation Framework - Interface Overview</i>	<i>Page 2</i>
<i>JSystem Automation Framework Main Application Interface</i>	<i>Page 2</i>
<i>Main Tool Bar Buttons</i>	<i>Page 3</i>
<i>Scenario Studio Panel</i>	<i>Page 5</i>
<i>Scenario JRunner Buttons</i>	<i>Page 6</i>
<i>Flow Control Toolbar</i>	<i>page7</i>
<i>Scenario Test Run Progress Bar</i>	<i>Page 7</i>
<i>Agent Toolbar</i>	<i>Page 8</i>
<i>Test Tree Tab</i>	<i>Page 9</i>
<i>Reporter Tab</i>	<i>Page 10</i>
<i>Fixtures Tab</i>	<i>Page 14</i>
<i>User Input</i>	<i>Page 18</i>
<i>Publisher Tab Run Properties Window</i>	<i>Page 20</i>
<i>JSystem Properties Dialog</i>	<i>Page 22</i>
<i>Edit SUT Fields Control Window</i>	<i>Page 26</i>
<i>Jar List Window</i>	<i>Page 27</i>
<i>HTML Reporter</i>	<i>Page 28</i>

5.1 JSystem Automation Framework - Interface Overview

Once the developer has written a test or tests in java using the Eclipse development environment they are then saved in the classes folder, to open the files within the JSystem Automation Framework click the **"Switch Project"** button, then follow the directory path in order to select the tests classes directory/project folder/classes and then click the **"Open"** button. The tests are then opened and appear in the **"Test Tree"** tab in the JSystem Automation Framework. The following section describes the JSystem Automation Framework GUI interface displaying a standard test scenario setup within the interface layout.

5.1.1 JSystem Automation Framework Main Application Interface

The JSystem work environment is divided into two main modules, the scenario editor where the scenarios are built and configured and the scenario JRunner, where configured test scenarios are run, analyzed and published.

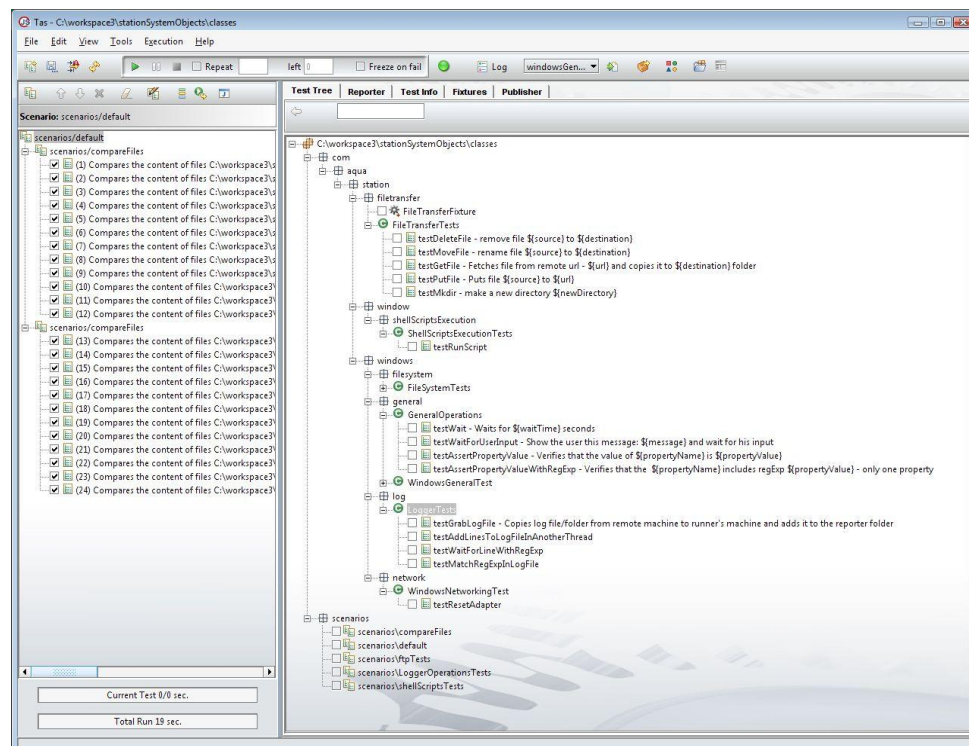


Figure 1: JSystem Automation Framework Main Window








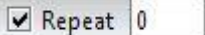
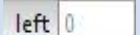
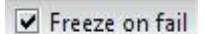




5.1.1.1 Building JSystem Scenarios

The scenario editor is divided into five tabs the most important of which is the **"Tree Tab"** that display the list of tests as they were written in java code adhering to the java hierarchical layout convention. These tests are divided into classes and function methods that appear with check boxes alongside them. To create a new scenario the user simply checks the required method or methods, clicks on the **"Add to Test Scenario"** button and the methods are added to the scenario JRunner pane.

The user can now run the test scenario by pressing the play button, JSystem then contacts the SUT and performs the test method functions that have been imported into the scenario JRunner from the scenario editor.

5.1.2 Main Tool Bar Buttons

The following table describes the JSystem Application buttons and their functionality.

Button	Name	Description
	New Scenario	This button is used to open a new scenario.
	Copy Scenario	This button is used to copy a new scenario.
	Save and Load Failed Sequence	This button enables the user to create a scenario from all the failed scenarios currently running in the Scenario runner.
	Refresh	This button is used to refresh the test after changes have been made in the test code, refreshing the JRunner.
	Play Tests	This button is used to play the test scenario.
	Pause	This button is used to pause the test scenario.
	Stop	This button is used to stop the test scenario.
	Number of scenario repeats	This button is used to set the number of times the test repeats itself.
	Number of scenario Remaining	This button is used to display the remaining amount of tests set in the repeat input field.
	Freeze on Fail	This checkbox is selected to freeze the test JRunner when it encounters a fail.
	SUT Planner	This button is pressed in order to open a wizard that is followed to create a new SUT file
	Log View	This button is used to open the test results in an HTML browser, enabling the user to drill down into the test results and verify the test results.
	SUT List	This button is a combo box that contains a list of all the SUT files.
	Edit SUT	This button enables the user to open a






		SUT and edit it.
	Export Wizard	This button is used to export the entire JSystem environment including the JRunner and all the installed drivers, classes, java code and configured tests as a zip file enabling the user to then open the zipped file and install a JSystem Player on another machine.
	SO Browser	This button is used to perform an intelligent analysis of the installed drivers and produce tests specifically tailored to the driver's individual needs.
	Switch Project	This button is used to switch between automation projects on the JSystem application suite.
	Open Reports Application	This button is used to quickly link to the reports application.
	Add Tests	This button adds selected tests to the scenario currently loaded in the scenario panel.

Table 1: JSystem Main Interface Buttons

5.1.3 Scenario Studio Panel

In order to create a new scenario the user simply checks the required method or methods, clicks on the ➡ “**Add Test**” button and the methods are automatically added to the scenario JRunner pane.

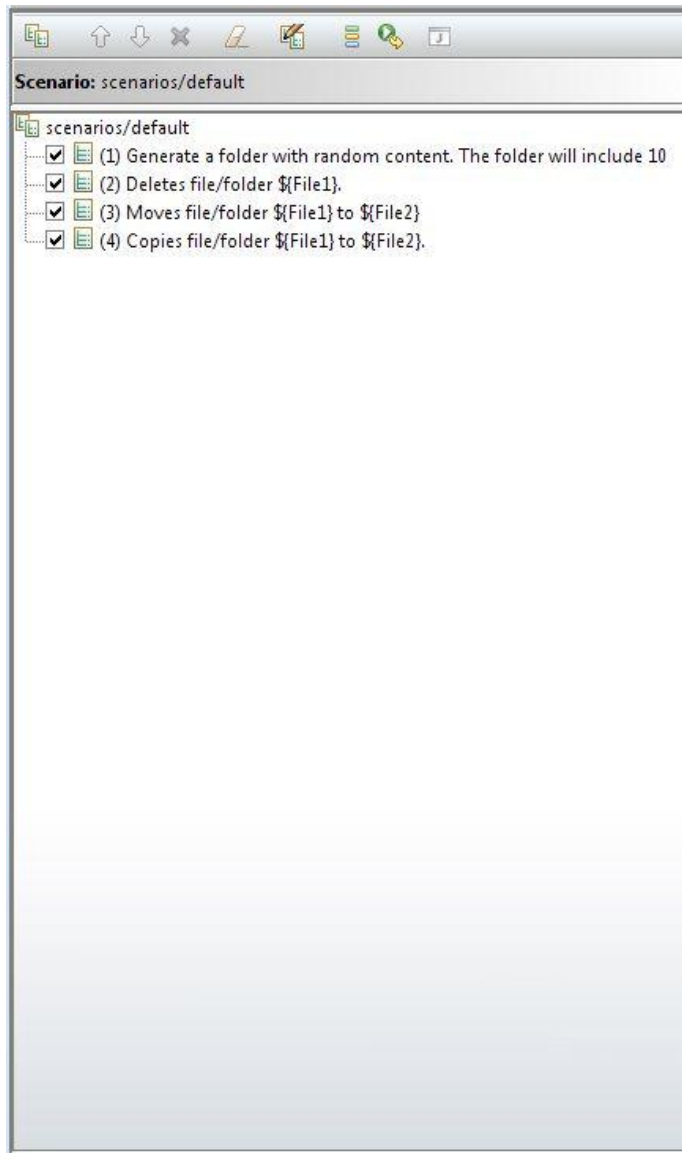


Figure 2: Scenario Studio Panel

The user can now run the test scenario by pressing the ▶ “**play**” button, JSystem then contacts the SUT and performs the test method functions that have been imported into the scenario runner from the scenario studio editor.

5.1.4 Scenario JRunner Buttons

The following table displays the function buttons of the Scenario runner.










Button	Name	Description
	Select Scenario	This button is used to select a scenario.
	Move Up Item	This button is used to move up the tree test scenario list.
	Move Down Item	This button is used to move down the test scenario tree list.
	Remove Item from Scenario	This button is used delete a scenario.
	Clear/Delete Scenario	This button is used delete The scenario.
	Edit Scenario	This button is used convert the scenario into an excel spread sheet format and then edit it and then save it automatically returning to the JSystem interface.
	Add Change SUT Event	This button enables the user to add and change an event during the test run changing the SUT file.
	Add Publish Results Event	This button adds an event that automatically publishes the test results to date, after the test cycle reaches it in the already running scenario.
	View Test Code	This button is used to open the java code in an HTML browser, where changes can be affected to the test run.

Table 2: Scenario Panel Buttons

5.1.5 Flow Control Toolbar

The purpose of flow controls is to add increased functionality to scenario test executions by adding logical programming elements such as if, else, switch case and for.



Figure 3: Flow Control Toolbar

5.1.6 Scenario Test Run Progress Bar

This progress bar visually displays the test runs progress position, and the amount of time remaining until the completion of all loaded tests in the Scenario JRunner.

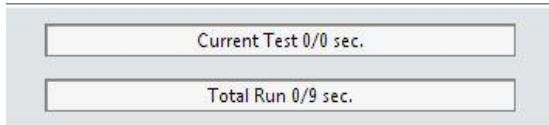


Figure 4: Scenario Test Run Progress Bar

5.1.6.1 Scenario Runner Tree Test Icons

The icons in **Table 50** appear in the Scenario JRunner panel as the test scenario runs in a downward direction, as the test passes the item one of the following icons appears reflecting the relevant information detailed in the table below.




Button	Name	Description
	Success	The test has passed the run.
	Assertion Error	Test failed on assertion, or a error report message was submitted
	Fail, General Exception	Test failed on a general exception

Table 3: Scenario Tree Test Icons

5.1.7 Agent Toolbar

The JSystems JRunner provides the user with the ability to connect to a remote agent via the "Agent Toolbar".



Figure 5: Agent Toolbar

Use the Connect to Agent "↑" button in order to connect to the remote agent address as defined in the input field.

5.1.7.1 Recent Connections

The drop down list on the right hand side of the IP input field provides the user with the agent connection recent history. The list is saved in the `jssystem.properties` file.

5.1.7.2 Agent Toolbar Icons




Button	Name	Description
	Connected	The JRunner is connected to the remote agent.
	Disconnected	The JRunner is not connected or has been disconnected from the JRunner.
	Unsynchronized	The JRunner and the agent are not yet synchronized.

Table 4: Agent Toolbar Icons

5.2 JSystem Main Window Tabs

The JSystem Automation Framework Test Editor panel is divided into five separate tab controls; these tabs provide the user the detailed testing configuration setup elements in an easy to understand workflow.

5.2.1 Test Tree Tab

The test tree tab contains the test, fixtures and test scenarios that have already been written, and imported into the JSystem Automation Framework.

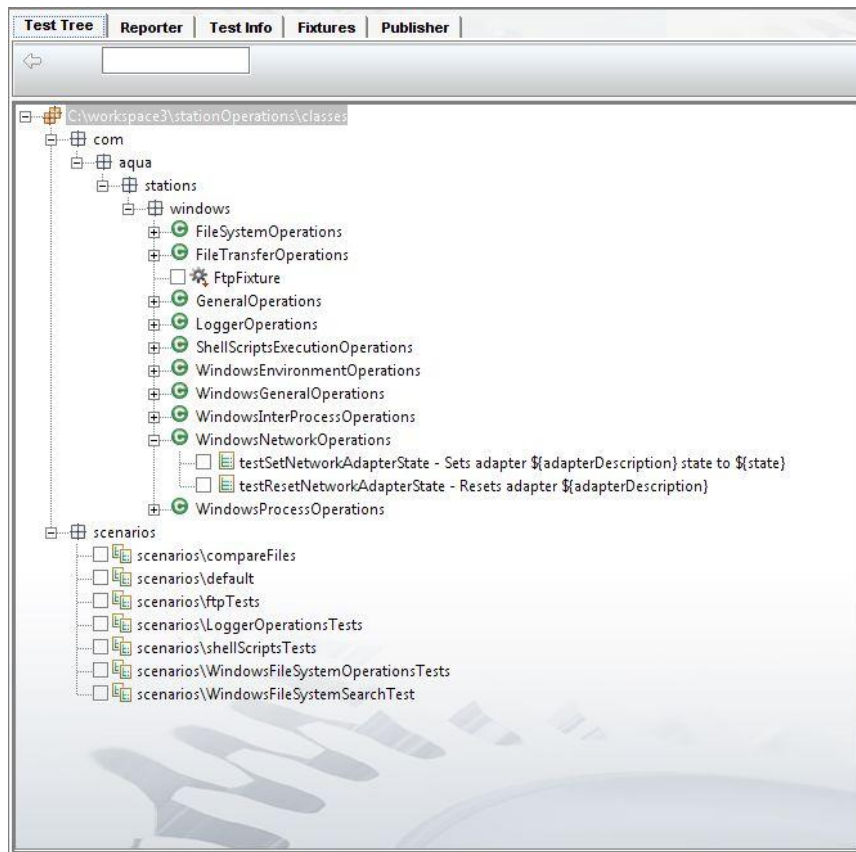


Figure 6: Test Tree Tab

5.2.1.1 Test Tree Icons

Button	Name	Description
	Package	Tests package
	Class	Java class element
	Test	Test case
	Fixture	Fixture item

Table 5: Test Tree Icons

5.2.2 Reporter Tab

The Reporter tab acts as a refined progress display window, it does not display all of the log report announcements. The JReporter tab displays all the first level log title announcements that are sent to the HTML reporter, in addition to this it also indicates that the test is advancing.

Test Tree	Reporter	Test Info	Fixtures	Publisher
Agent: local Log Init Reporters				
Commands		Status		
11:15:52: Setting jsystem properties		✓		
11:15:52: Jsystem properties were set		✓		
11:15:57: Zipping local project		✓		
11:16:12: Finished Zipping local project		✓		
11:16:12: Starting synchronization		✓		
11:16:12: Setting active project		✓		
11:16:12: Starting upload to 10.0.0.28:8999		✓		
11:16:12: Put C:\Users\laqua\AppData\Local\Temp\project42259.zip in project42259.zip		✓		
11:16:18: Setting active scenario to scenarios/default		✓		
11:16:18: Setting active sut to analysisSut.xml		✓		
11:16:18: Ended synchronizing 10.0.0.28:8999		✓		
11:16:23: Agent was signaled to run		✓		
11:18:02: Put C:\Users\laqua\AppData\Local\Temp\project42261.zip in project42261.zip		✓		
11:30:39: Setting jsystem properties		✓		
11:30:39: Jsystem properties were set		✓		
11:30:41: Zipping local project		✓		
11:30:46: Finished Zipping local project		✓		
11:30:46: Starting synchronization		✓		
11:30:46: Setting active project		✓		
11:30:47: Starting upload to 10.0.0.28:8999		✓		
11:30:47: Put C:\Users\laqua\AppData\Local\Temp\project42265.zip in project42265.zip		✓		
11:30:54: Setting active scenario to scenarios/default		✓		
11:30:55: Setting active sut to analysisSut.xml		✓		
11:30:55: Ended synchronizing 10.0.0.28:8999		✓		
11:30:57: Agent was signaled to run		✓		
11:32:47: Setting jsystem properties		✓		
11:32:48: Jsystem properties were set		✓		
11:32:49: Zipping local project		✓		
11:32:54: Finished Zipping local project		✓		
11:32:54: Starting synchronization		✓		
11:32:54: Setting active project		✓		
11:32:55: Starting upload to 10.0.0.28:8999		✓		
11:32:55: Put C:\Users\laqua\AppData\Local\Temp\project42267.zip in project42267.zip		✓		
11:33:03: Setting active scenario to scenarios/default		✓		
11:33:03: Setting active sut to analysisSut.xml		✓		
11:33:03: Ended synchronizing 10.0.0.28:8999		✓		
11:33:05: Agent was signaled to run		✓		

Figure 7: Reporter Tab

5.2.2.1 Agent Tabs

Once an agent is added to the **"Agents List"** dialog, a new sub-tab is opened for the agent in the **"Reporter"** tab. By pressing the tab, the user can see the report messages sent from the remote agent execution. The tab also shows the status of the connection to the remote agent.

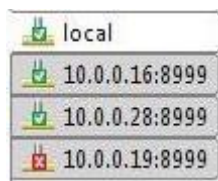


Figure 8: Agent Tab Control

5.2.2.2 Test Tree Tab Status Colors and Icons

Color	Name	Icon	Description
Green	Success message	✓	Success message
Red	Failure message	⚠	Error message

Table 6: Test Tree Status Color and Icon Definitions

5.2.3 Agent List Window

The Agent list window contains the list of remote agents and is used in order to configure additional remote agents and to synchronize the remote agent with local automation project settings

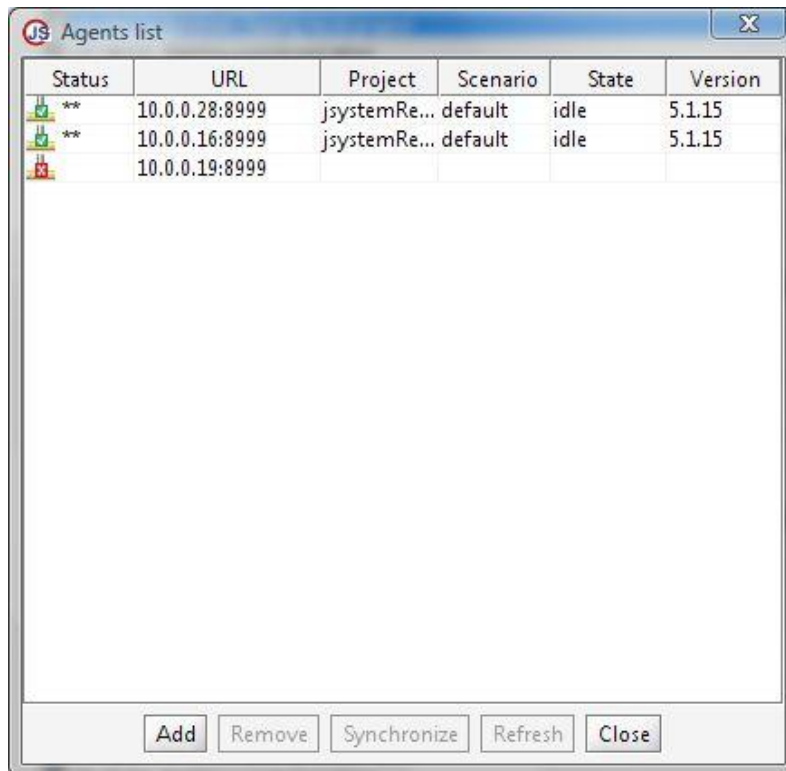


Figure 9: Agent List Window

5.2.4 Test Info Tab

The Test Info tab displays the individual test item information from each test that is selected from within the Scenario JRunner pane.

The screenshot shows a software window with a tabbed interface. The 'Test Info' tab is active. At the top, there are tabs for 'Test Tree', 'Reporter', 'Test Info', 'Fixtures', and 'Publisher'. Below these, the following text is displayed:

Name: tmp.InfoTabTest-testInfoTab
Fixture: jsystem.framework.fixture.RootFixture
Fail to fixture: not set
Last run time: unknown

Below this is a section for 'Test Documentation' and 'User Test Documentation'. The 'Test Documentation' section contains a list of steps:

- 1) run traffic on both Devices.
- 2) analyze that both devices got all packages
- 3) change bridge mode to Dot1Q
- 4) run traffic again and check that all traffic passed

The main area of the window is a table with the following data:

Name	Description	Type	Value
StringParameter*	String value description	string	
IntegerParameter*	Integer value description	int	42
BooleanParameter*	True means....	boolean	true
Compare*	used for comparing	enum	EQUAL

At the bottom of the window, there are two buttons: 'Sort Sections' and 'Apply for Scenario'.

Figure 10: Test Info Tab

Note: This tab is tightly coupled with the test java class.

5.2.4.1 Test Tab Information

The information displayed in the Test Info tab is divided into Name, Fixture, Fail to Fixture and the last time the test was run as well as Test documentation and User Test Documentation tabs.

Test Documentation – Test java doc documentation as appears in the java code written by the java programmer.

User Test Documentation –The User documentation is entered by the test operator enabling the user to add comments about individual tests while using the Scenario Runner.

Parameters Table:

Parameters	Description
Name	The name of the test parameter as it appears in the test code.
Description	The description is retrieved from the parameter set method; it describes parameter usage and functionality.
Type	This describes the type of parameter.
Value	The current value of the parameter.

Figure 11: Test Tab Information Table

5.2.5 Fixtures Tab

The fixtures tab displays the dynamic hierarchy of the fixture elements, all the control and functionality of the fixtures is controlled and run from within the fixture tab.

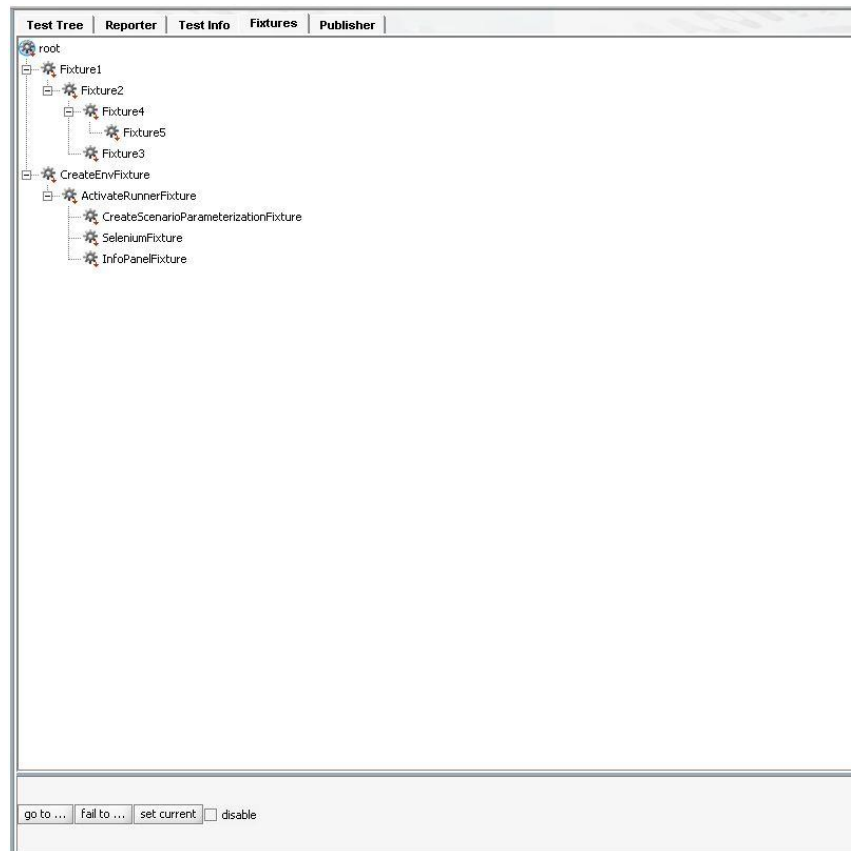


Figure 12: Fixtures Tab

Reference: In order to learn how to write a Fixture, refer to Error! Reference source not found. on page Error! Bookmark not defined..

5.2.5.1 Fixture Tab Icon

The Fixture tab contains a hierarchical tree that displays the fixtures according to their hierarchical runtime order. A fixture is equivalent to the system status that includes system configuration and their relevant configuration setup files.


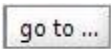
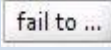
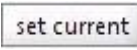
Button	Name	Description
	Fixture	
	Go To	Pressing the button navigates to the currently selected fixture.
	Fail To	By pressing this button the system performs a "Fail-to Navigation" function to the currently selected fixture.
	Set Current	This button sets the currently selected fixture to be the active fixture (the system jumps directly the fixture).

Table 7: Fixture Tab Button

Each fixture has an order of events that is instigated when the system is setup, as well as operations performed on system teardown.

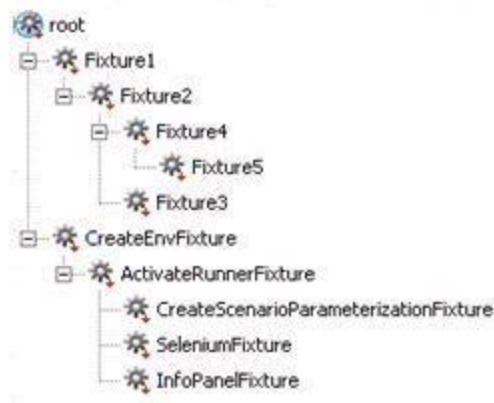


Figure 13: Fixture Root Tree

A fixture parent is a predefined system status object on which all subsequent fixtures are based. If the parent fixture remains undefined it reverts to its default state (root fixture).

Root – The Root Fixture is the built-in default fixture that is always set as the root or base fixture within the Root Fixture hierarchy.

5.2.6 Publisher Tab

The Publisher tab displays the published results of a test after it has been run, once the refresh button has been pressed the left hand side of the tab display loads and displays the tests that have been run by JSystem Automation Framework.

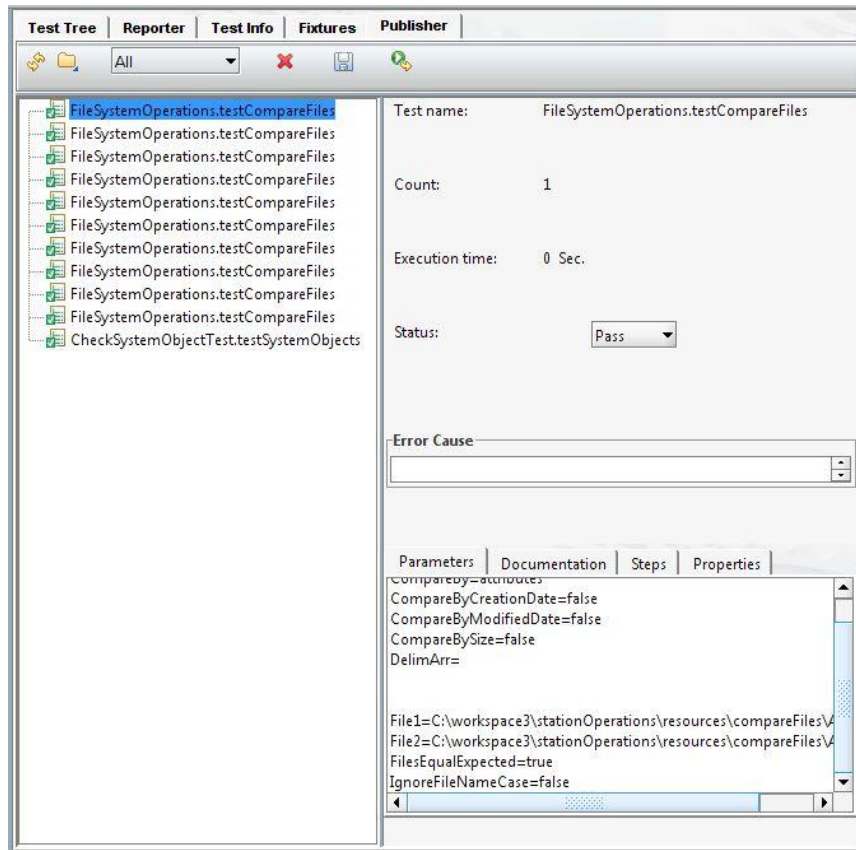


Figure 14: Publisher Tab

The test results can be published to the reports server by pressing the  "Publish XML result" button.

The right hand side of the Publisher tab displays the individual tests name information, the number of times the test has been run, the Execution time displaying the time taken to run the test and the current status of the test, the status of the test can be changed from pass to fail by selecting from the pull down menu and selecting either pass or fail.

By publishing the results the user can save the tests, statistics and run logs in a DB file (JSystem Automation Framework currently supports MySQL) for analysis in the JReporter application.

Note: The user can revise all test statistics and data and edit the content before publishing.

5.2.6.1 User Input

The user has the ability to input documentation data that pertains to the specific test after the run is complete. This input can be published with the test results and can be reviewed from the JReporter application.

5.2.6.2 Publisher Information Pane

The publisher information pane shows information regarding the present test selection being executed.

Note: *The status input field is the only field that the user can edit.*

Field Name	Description
Test Name	Details the test class and method.
Execution Time	The exact commencement time that the test scenario was executed.
Status	The test execution status can be set to one of the following settings; pass/failure/warning.
Error Cause	The user can add a reason for a test failure. Failure explanations are shown in the report application output report.
Count	Shows the amount of times the test has been run.

Figure 15: Publisher Tab Descriptions

5.2.7 Publisher Sub Tabs

The Publisher sub-tabs detail test report information that refers to the currently selected test chosen from the test list.

5.2.7.1 Parameters Sub-Tab

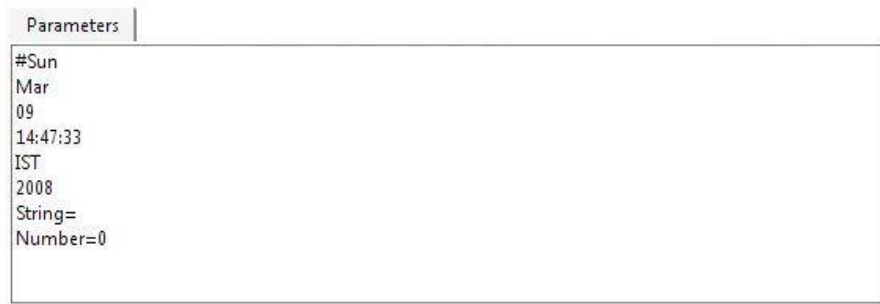


Figure 16: Publisher Sub-Tab Parameters

Parameters – Details the test parameter values at test execution.

5.2.7.2 Documentation Sub-Tab

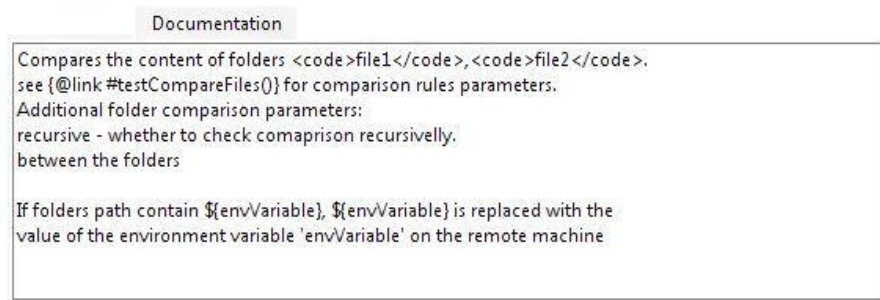


Figure 17: Publisher Sub-Tab Documentation

Documentation – The test documentation from java doc. The documentation can be altered if required.

5.2.7.3 Steps Sub-Tab

The Step sub-tab shows the current test step selections reported during test execution. These steps appear in the full executive report from within the JReporter application.



Figure 18: Publisher Sub-Tab Steps

Steps – The steps that the test printed during the test run (from the test code).

5.2.7.4 Properties Sub-Tab

The properties sub-tab shows the keys and values of the properties gathered during test executions. These properties used for creating customized reports in the JReporter application and are used for performance analysis. Key values can also be removed before publishing reports.

Warning: Keys and Values cannot contain &, # or % symbols, if these symbols are present the user will receive a "Warning Message".



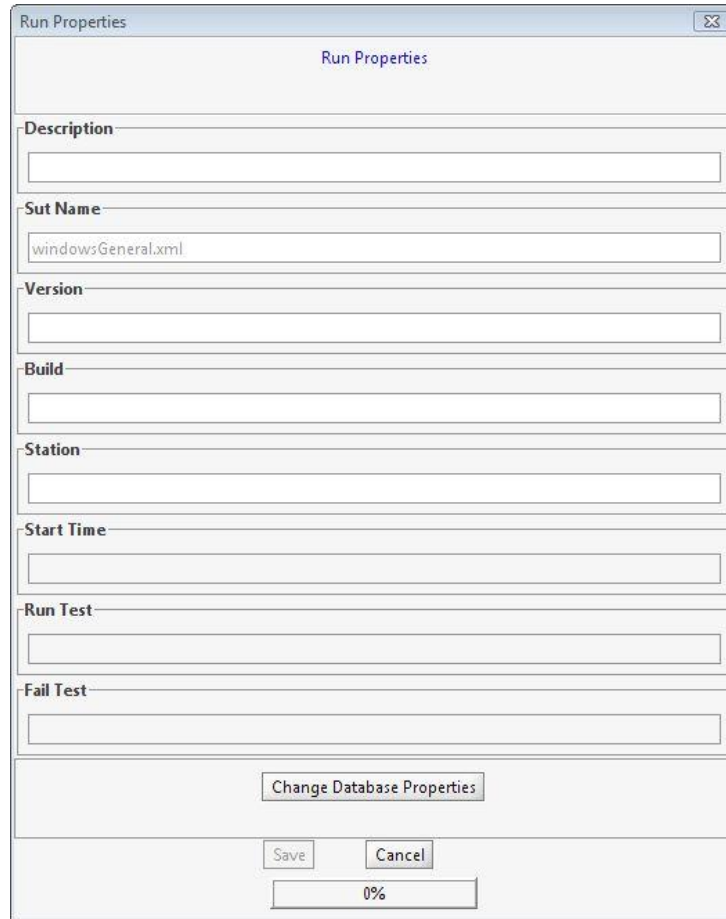
Figure 19: Publisher Sub-Tab Properties

Properties – Keys and values that are gathered during the test run.

Reference: For more information about the Custom Report Builder and about how to write tests refer to Error! Reference source not found., on page Error! Bookmark not defined..

5.2.8 Publisher Tab Run Properties Window

The Run Properties window opens after the  “**Publish XML Results**” button is pressed, enabling the user to enter the run properties description details.



The screenshot shows a window titled "Run Properties" with a close button in the top right corner. The window contains several input fields, each preceded by a minus sign in a square box. The fields are: "Description" (empty), "Sut Name" (containing "windows:General.xml"), "Version" (empty), "Build" (empty), "Station" (empty), "Start Time" (empty), "Run Test" (empty), and "Fail Test" (empty). Below these fields is a button labeled "Change Database Properties". At the bottom of the window are two buttons, "Save" and "Cancel", and a progress bar showing "0%".

Figure 20: Publisher Tab Run Properties Window

5.2.8.1 Run Properties Input Fields

Description – User input that describes the general execution description of the current run.

SUT Name – The SUT xml file that is used in the current run of the test.

Version – Editable text used to define the version of the tested product.

Build – Editable text used to define the Build number of the tested product.

Station – Local station IP address.

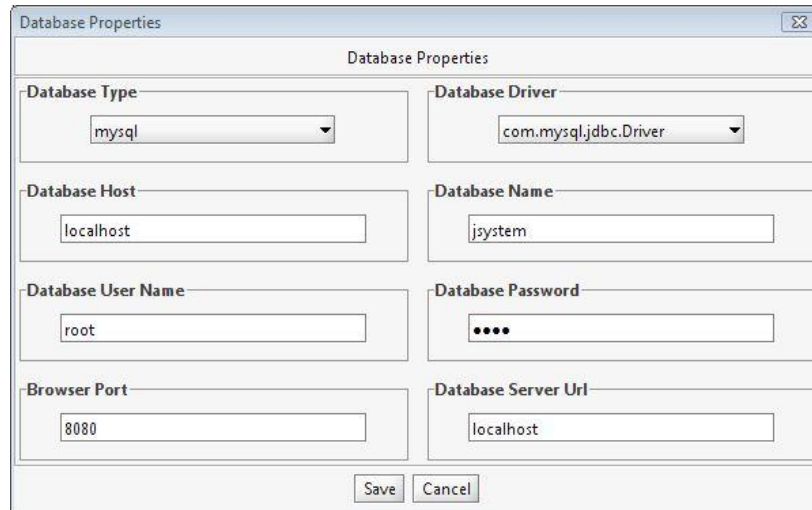
Start Time – The start time and date of the first test in the current run.

Run Test – Number of tests that have been executed.

Fail Test – The number of failed tests in the run.

5.2.9 Database Properties Window

By pressing the “**Change Database Properties**” button the “**Database Properties**” window opens, the user can now configure, update or change input configuration data that is required to configure the MySQL or Oracle database server, and select and configure the “**Database Driver**”.



The screenshot shows a window titled "Database Properties" with a close button in the top right corner. The window contains several input fields arranged in a grid:

- Database Type:** A dropdown menu with "mysql" selected.
- Database Driver:** A dropdown menu with "com.mysql.jdbc.Driver" selected.
- Database Host:** A text input field containing "localhost".
- Database Name:** A text input field containing "jssystem".
- Database User Name:** A text input field containing "root".
- Database Password:** A text input field with four dots (password masked).
- Browser Port:** A text input field containing "8080".
- Database Server Url:** A text input field containing "localhost".

At the bottom of the window, there are two buttons: "Save" and "Cancel".

Figure 21: Publisher Tab Change Database Properties Window

The database properties are entered with the first database publish execution, the input data can be modified a future publish runs if required.

5.2.9.1 Database Properties Input Fields

Database Type – The user selects the type of database to be used for storing statistics (MySQL/Oracle).

Database Driver – The java database connectivity (JDBC) driver used for the database type.

Database Host – The address of the computer hosting the database is (localhost/10.0.0.2x).

Database Name – The MySQL Schema folder where the database tables resides.

Database User Name – The user name for the database.

Browser Port – This is the port that the tomcat is installed on.

Database Password – The database password is configured by the system administrator.

Database Server URI – The address of the computer that hosts the tomcat server application is (localhost/10.0.0.12x).

5.3 JSystem Properties Dialog

The JSystem Properties dialog is a descriptive and easy to use UI that helps the user to easily alter the **JSystem.properties** file.

The JRunner exposes behavioral and customizable configuration functionality via the JSystem Properties Dialog.

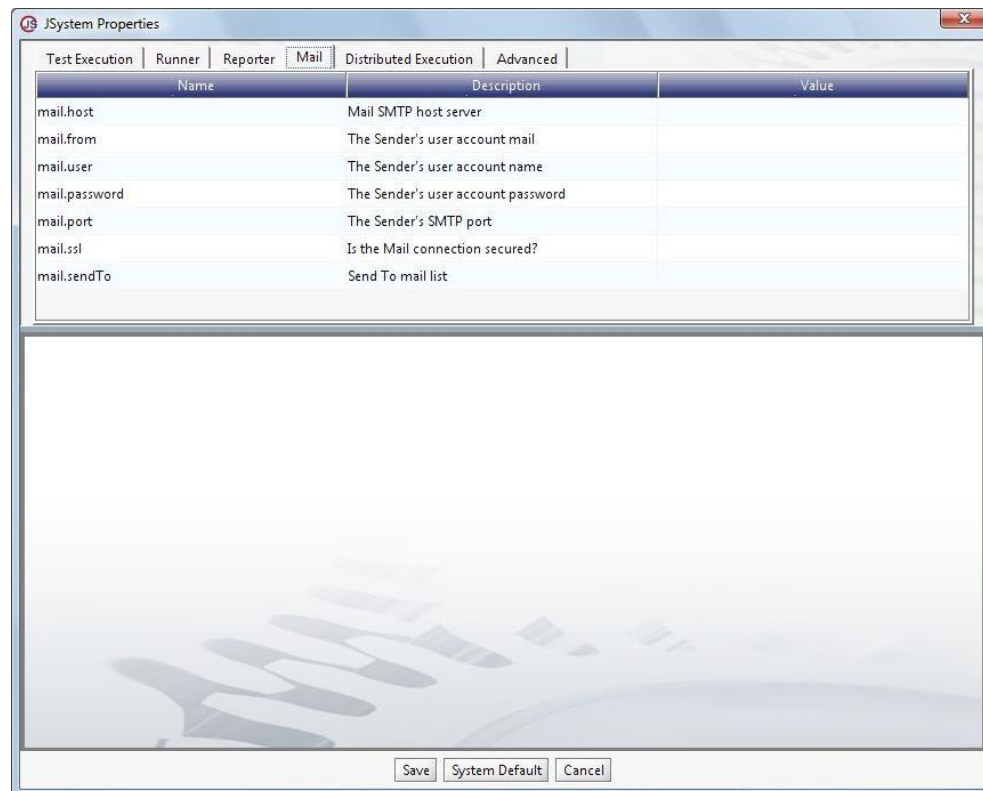


Figure 22: JSystem Properties Windows

5.3.1.1 JSystem Properties Dialog Tabs

- **Test Execution** – This tab provides user the ability to enable or disable the notification engine, control java parameters and locate jar file directories.
- **JRunner** – This tab provides user functionality for setting XML files, SUT file details, scenario editor settings, conserving computer resources settings and setting the Product customer tag information.
- **Log** – This tab provides the user settings for the JSystem reporter properties and logger settings.
- **Mail** – This tab provides the user with the ability to set the publish event mail properties.
- **Agent** – This tab provides the user with the ability to set and alter the Distributed execution during the test execution and the plug-in properties.
- **Advanced** – This tab provides the user with advanced settings related to Threads, package inclusion/exclusion, disabling the loader, disabling the scenario search function, timeout settings, metadata preset information and settings.

5.4 JSystems Export Project Wizard

The JSystem export wizard is used to configure the exportable zip file that contains selected project data.

5.4.1 Step 1

Step1 of the export wizard is used for backing up the running environment or moving and or copying it to another workspace or station.

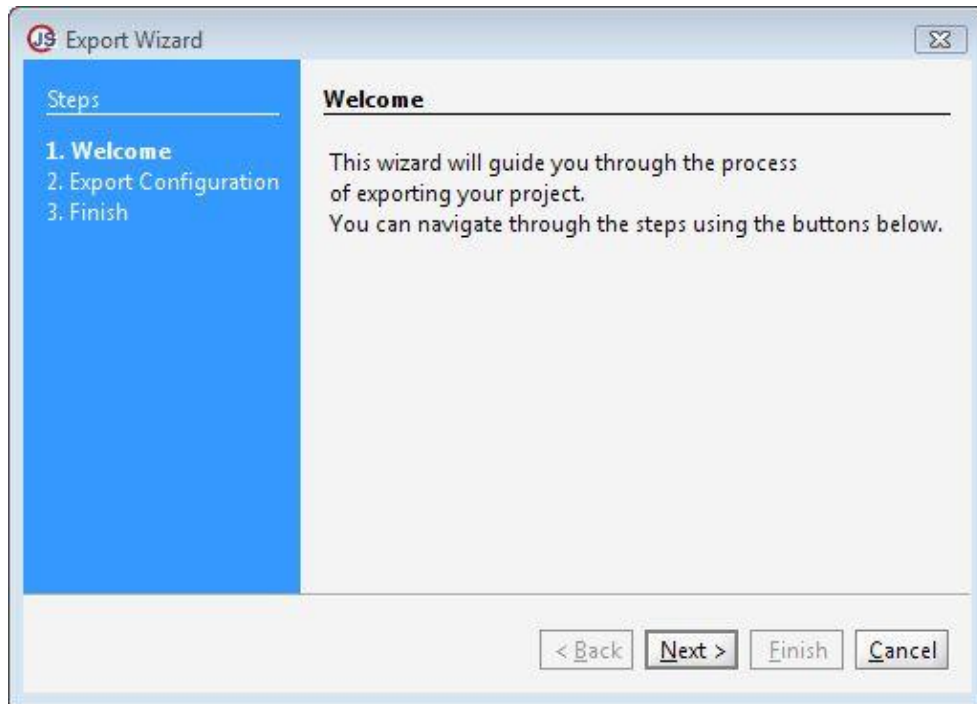


Figure 23: JSystems Export Project Wizard

The Export Wizard contains three simple steps that require the use to enter location paths and select check box items that affect the content of the export zip file.

5.4.2 Export Wizard Steps 2 and 3

Step 2 of the Export wizard, containing the configuration items and path location input fields.

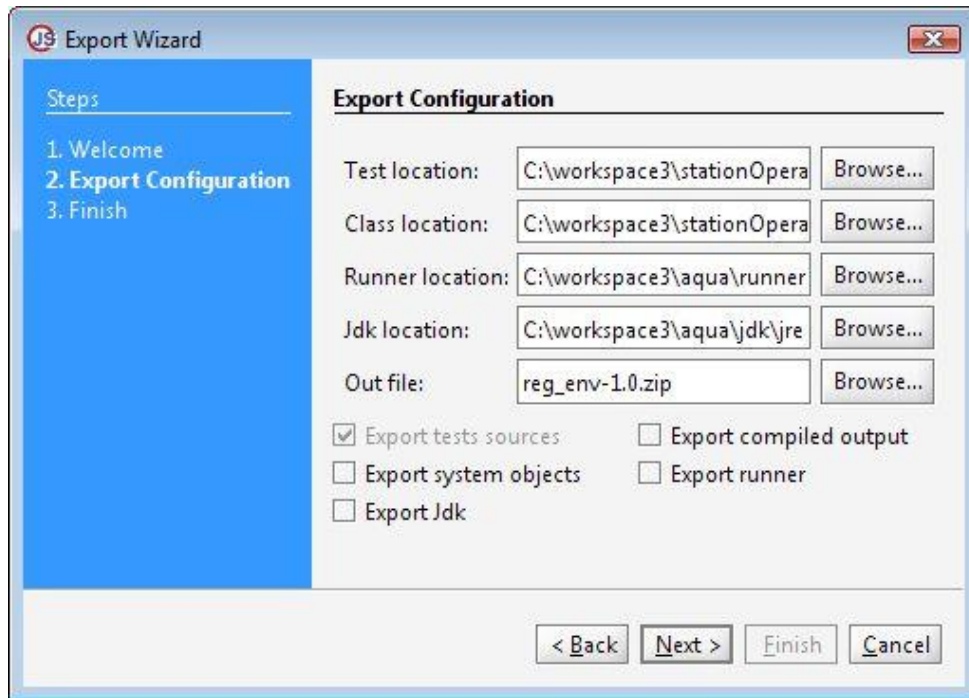


Figure 24: JSysystems Export Project Wizard 2

5.4.2.1 Export Wizard Input Fields

Field Name	Description
Export tests sources	Default export setting that exports the original java tests files.
Export system objects	An optional checkbox enabling the export of the SRC folder.
Export JDK	An optional checkbox function that enables the user to export the JDK folder.
Export compiled output	Optional checkbox enabling the export of the class folder.
Export Runner	Optional checkbox that enables the export of the JRunner.

Table 8: Export Wizard Input Fields

Step 3 offers the user the option of either completing the wizard or returning to change one or more of the configuration settings previously configured.



Figure 25: JSsystems Export Project Wizard 3

5.5 Edit SUT Fields Control Window

The Edit SUT Fields window opens after pressing the Edit SUT button enabling the user to edit the system under file (SUT).

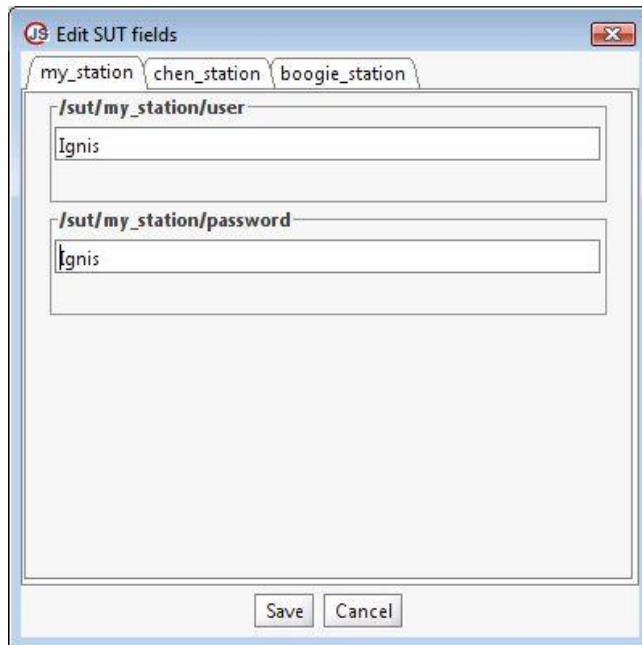


Figure 26: Edit SUT Fields Window Example

The screen shot represents an example of a formatted SUT file, as this window is a result of customization by the user requiring user input data.

5.5.1 Editing and Customizing the System Under Test (SUT)

The JSystem Automation Framework supports the ability to customize the user SUT file according to the user requirements for each project. This is achieved by enabling the service administrator with the ability to easily customize the **"Edit SUT Fields"** tabs window by performing the following simple steps.

5.5.1.1 Step 1: Notepad

By default, the SUT xml file is opened in the note pad application and all fields are visible. In order to edit and affect change to the **"Edit SUT Fields"** window the user must perform the following operations:

The system administrator performs the customization of the DUT file of the JRunner. In order to edit the tags in the SUT file the administrator must enable the tags by adding **edit = "enable"** inside each individual tag.

5.5.1.2 Step 2: Compile

In order to compile the newly written tag the administrator must **"save"** the **"Notepad"** file in order to save the changes.

5.5.1.3 Step 3: View Tab Window

In order to view the resulting changes made in the **"Edit SUT Fields"** window the user simply opens it from the JRunner application by pressing the **"Edit SUT"** button.

Note: There is no revert function on SUT file where the customized changes were made to the file.

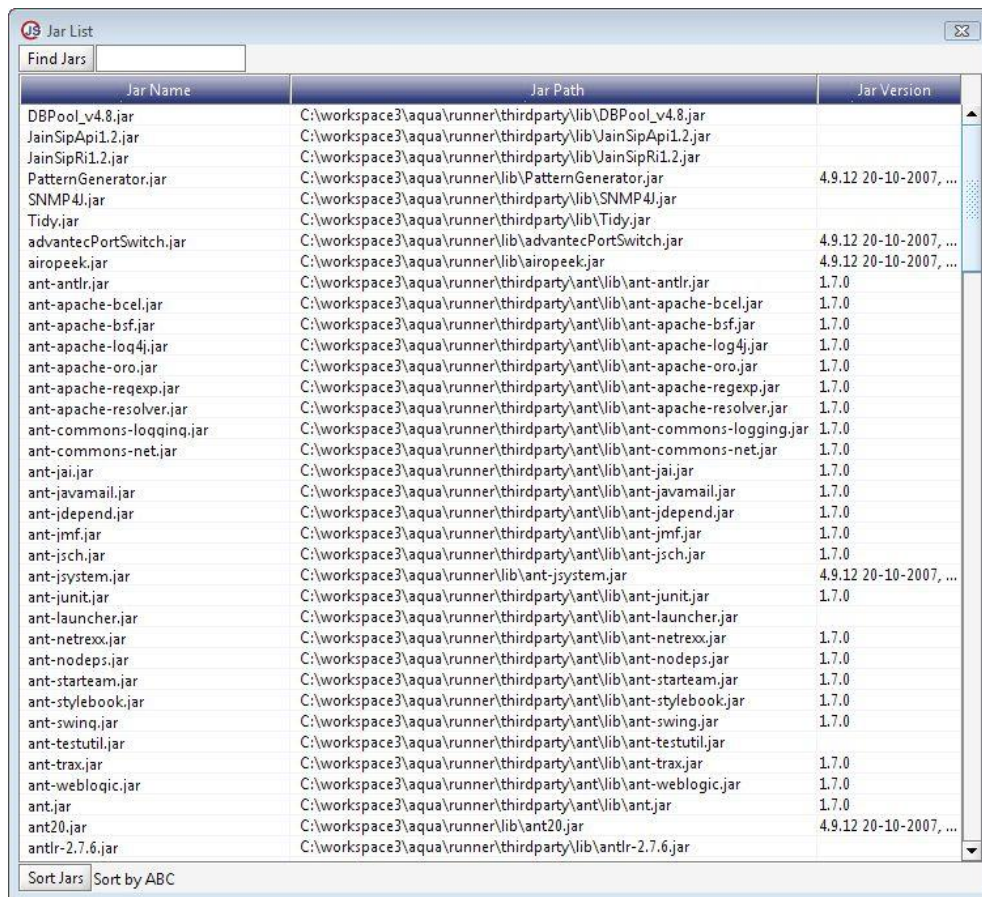
In order to revert to the default setting of the SUT file, the user must open the folder that contains the user defined xml file or files and delete them manually. Repeat the process until the desired results are achieved.

Note: The user can replace the default Notepad with Notepad ++ enabling the programmer to copy java code with ease.

Jar List Window

The Jar List window shows all of the Jar “**packets**” that are present in the JRunner environment, the JRunner Library “**lib**” and third party folders.

Note: Each Jar packet has a path and version number.



Jar Name	Jar Path	Jar Version
DBPool_v4.8.jar	C:\workspace3\aquarunner\thirdparty\lib\DBPool_v4.8.jar	
JainSipApi1.2.jar	C:\workspace3\aquarunner\thirdparty\lib\JainSipApi1.2.jar	
JainSipRi1.2.jar	C:\workspace3\aquarunner\thirdparty\lib\JainSipRi1.2.jar	
PatternGenerator.jar	C:\workspace3\aquarunner\lib\PatternGenerator.jar	4.9.12 20-10-2007, ...
SNMP4J.jar	C:\workspace3\aquarunner\thirdparty\lib\SNMP4J.jar	
Tidy.jar	C:\workspace3\aquarunner\thirdparty\lib\Tidy.jar	
advantecPortSwitch.jar	C:\workspace3\aquarunner\lib\advantecPortSwitch.jar	4.9.12 20-10-2007, ...
airopeek.jar	C:\workspace3\aquarunner\lib\airopeek.jar	4.9.12 20-10-2007, ...
ant-antlr.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-antlr.jar	1.7.0
ant-apache-bcel.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-apache-bcel.jar	1.7.0
ant-apache-bsf.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-apache-bsf.jar	1.7.0
ant-apache-log4j.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-apache-log4j.jar	1.7.0
ant-apache-oro.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-apache-oro.jar	1.7.0
ant-apache-regexp.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-apache-regexp.jar	1.7.0
ant-apache-resolver.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-apache-resolver.jar	1.7.0
ant-commons-logging.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-commons-logging.jar	1.7.0
ant-commons-net.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-commons-net.jar	1.7.0
ant-jai.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-jai.jar	1.7.0
ant-javamail.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-javamail.jar	1.7.0
ant-jdepend.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-jdepend.jar	1.7.0
ant-jmf.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-jmf.jar	1.7.0
ant-jsch.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-jsch.jar	1.7.0
ant-jsystem.jar	C:\workspace3\aquarunner\lib\ant-jsystem.jar	4.9.12 20-10-2007, ...
ant-junit.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-junit.jar	1.7.0
ant-launcher.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-launcher.jar	
ant-netrexx.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-netrexx.jar	1.7.0
ant-nodeps.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-nodeps.jar	1.7.0
ant-starteam.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-starteam.jar	1.7.0
ant-stylebook.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-stylebook.jar	1.7.0
ant-swing.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-swing.jar	1.7.0
ant-testutil.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-testutil.jar	
ant-trax.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-trax.jar	1.7.0
ant-weblogic.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant-weblogic.jar	1.7.0
ant.jar	C:\workspace3\aquarunner\thirdparty\ant\lib\ant.jar	1.7.0
ant20.jar	C:\workspace3\aquarunner\lib\ant20.jar	4.9.12 20-10-2007, ...
antlr-2.7.6.jar	C:\workspace3\aquarunner\thirdparty\lib\antlr-2.7.6.jar	

Figure 27: Jar List Window

5.5.2 JRunner Environment Jar List Names

Jar Name – The name of the Jar file.

Jar Path – The address path of the Jar file.

Jar Version – Release date of the Jar file.

5.6 HTML Reporter

The HTML report (log) file is a detailed report of a run set of tests. The HTML report is created during the tests that are run and shows the details of each test.

5.6.1 Tracking Errors

By utilizing the reporter the user can see all of the messages printed during tests that are run and the errors that occurred. The errors appear in red, and the warnings appear in orange. The reporter also has a summary report function that shows the entire pass, fail and warning report that occurred.

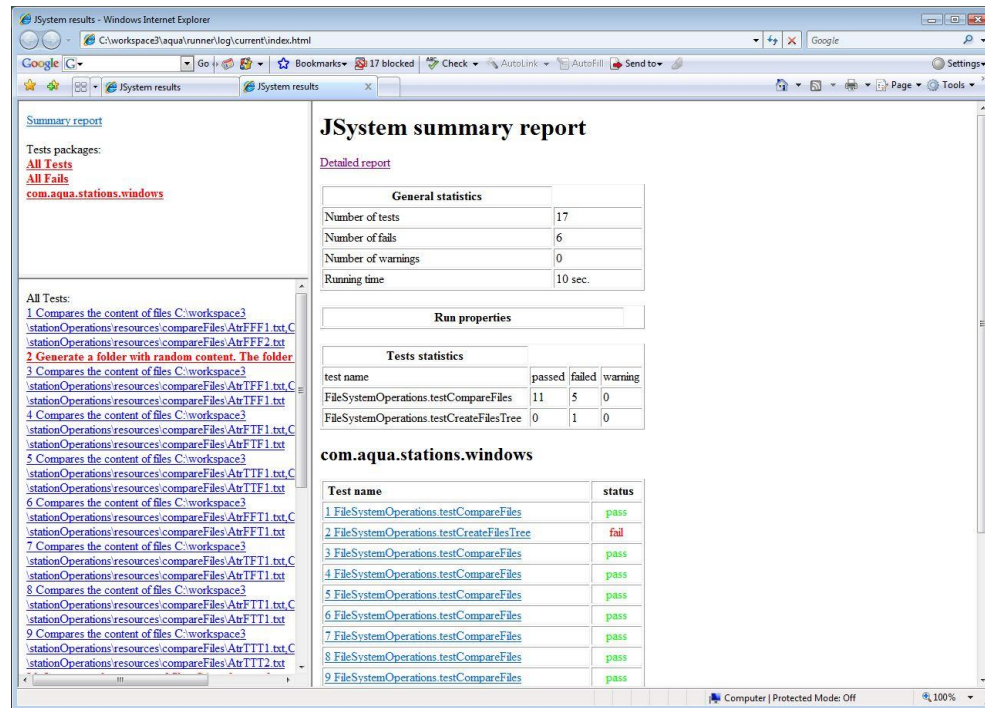


Figure 28: HTML Reporter