



Version 5.4

Release Notes

Disclaimer

This document is part of JSystem open source project. JSystem is released under Apache 2.0 license. The license can be found [here](#).

© 2005-2010 Ignis Software Tools Ltd. All rights reserved.



Table of Contents

Version 5.4	1-1
Release Notes.....	1-1
Disclaimer.....	1-1
Table of Contents.....	1-2
1. Purpose of this Document	1-3
1.1 New Features Summary	1-3
1.2 Licensing Note	1-3
2. Model Changes	2-4
2.1 Test Parameters in Sub-scenarios.....	2-4
2.2 Mark Scenario as a Test	2-5
2.2.1 Marking Scenario as Test	2-5
2.2.2 Changing “Scenario as Test” Internal Steps Parameters	2-7
2.3 Mark Test as Known Issue	2-7
3. Execution from command line	3-9
4. Scenario Studio	4-10
4.1 Saving Scenario File	4-10
4.2 Undo/Redo	4-10
4.3 Easy Scenario Navigation	4-12
4.3.1 Navigating to sub scenario	4-12
4.3.2 Navigating back and forward between scenarios	4-13
4.4 Enhanced Test Parameters GUI	4-14
4.4.1 List option select dialog	4-14
4.4.2 Generic UI for JavaBeans	4-15
4.4.3 JavaBean Array.....	4-16
4.4.4 Custom Data Model for Bean Array Generic UI	4-18
4.5 Writing your own UI parameter provider	4-20
5. Reporter	5-21
5.1 Parameters into Level	5-21
5.2 Errors HTML reporter	5-24



1. Purpose of this Document

These release notes describe the new features and bug fixes for version 5.4 of **JSystem Automation Platform**.

1.1 New Features Summary

JSystem 5.4 includes features that span over all of the automation development lifecycle and users functionality. The new features include the following:

- Model changes
- Execution from command line
- Scenario Studio changes
- JSystem GUI Improvements
- Reporter add-ons

1.2 Licensing Note

JSystem 5.4 is released under Apache version 2.0 license.

JSystem 5.4 uses many open source projects. A complete list of all projects and their licenses can be found in [JSystem Trac](#).

Note that JSystem core uses only projects with BSD like licenses (BSD, Apache 2.0, LGPL etc.) with the exception of java2html (GPL). Java2html project is removed in JSystem 5.5. If you don't wish to use it you can download the nightly build ([windows](#),[linux](#)).



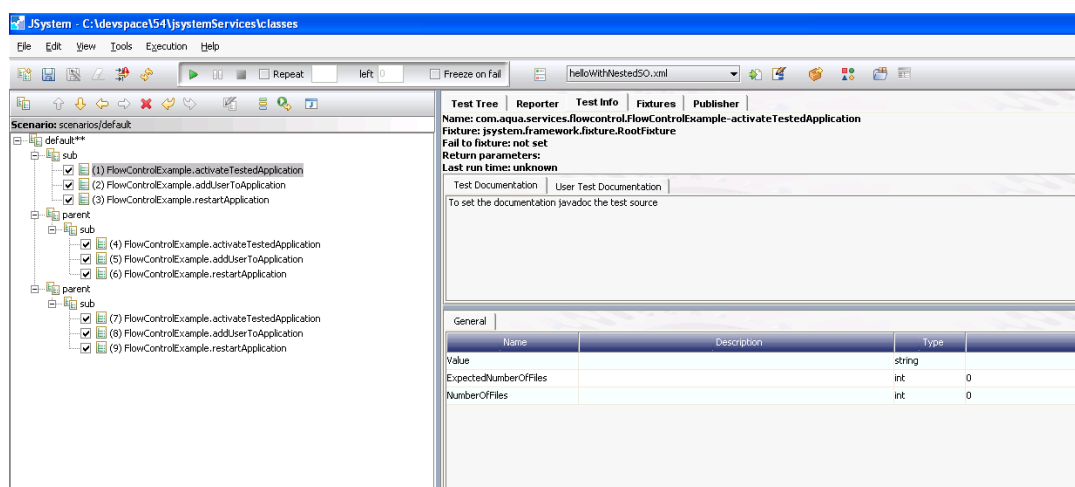
2. Model Changes

JSystem 5.4 introduces some model enhancements which improve the usability of the runner scenario studio and HTML reporter, and increase the power of the JRunner as a development tool.

2.1 Test Parameters in Sub-scenarios

Starting at JSystem 5.3, the user can assign different value for test parameter in different instances of sub-scenario.

For example in this scenario:



The test parameter “Value” of the test ‘*activetTestedApplication*’ which appears in several instances of the *sub*-scenario can get different values in each instance of the *sub*-scenario.

Note that in order to support this functionality, scenario data is now saved in two files:

1. scenario-name.xml – holds ant script
2. scenario-name.properties – holds test parameters values.



2.2 Mark Scenario as a Test

When creating a large scenario with hierarchy of sub-scenarios in it, the scenario becomes overloaded, and hard to work with.

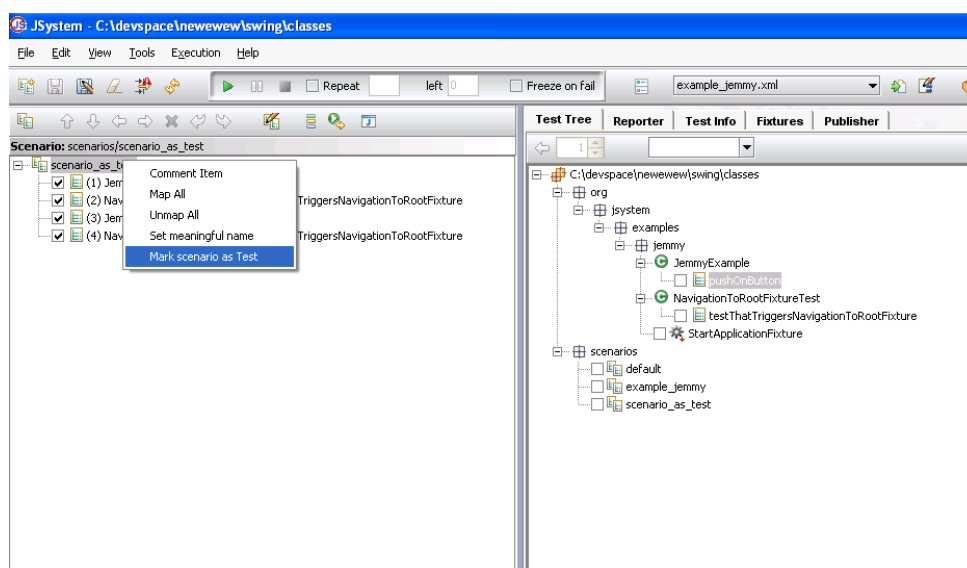
To deal with this drawback, the “Mark scenario as test” model change was implemented. When marking scenario as a test, the scenario is packed and shown in the JRunner as a test. In addition, in the HTML reporter the scenario is shown as a test. When drilling down into scenario (as test) log information, internal steps log information is shown in internal levels.

In addition, when a scenario is marked as a test, and one of the internal scenario steps fail, scenario execution is stopped and the runner skips to the next test.

2.2.1 Marking Scenario as Test

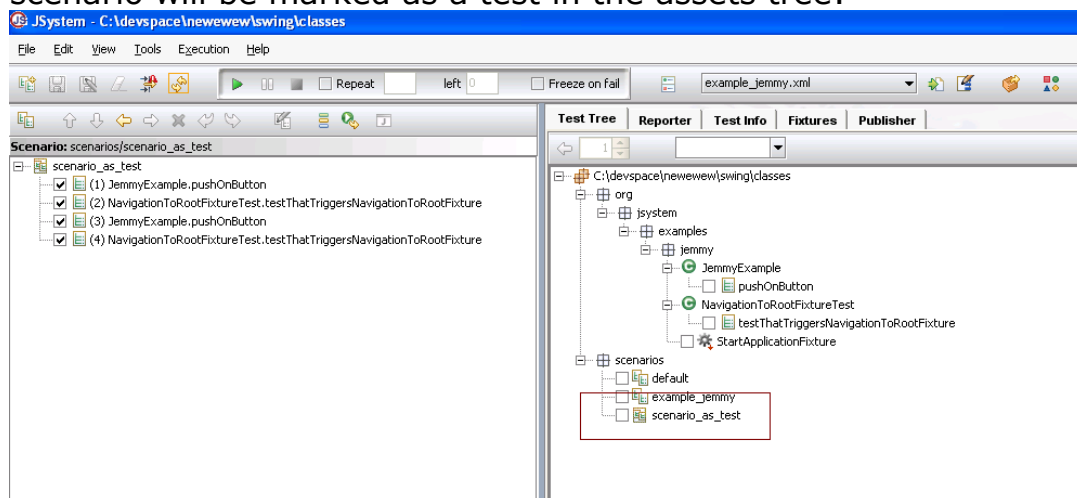
To mark a scenario as a test, perform the following:

1. Select the scenario that you want to mark as a test as a root scenario.
2. Right click on the scenario and select the “Mark Scenario As Test” menu item





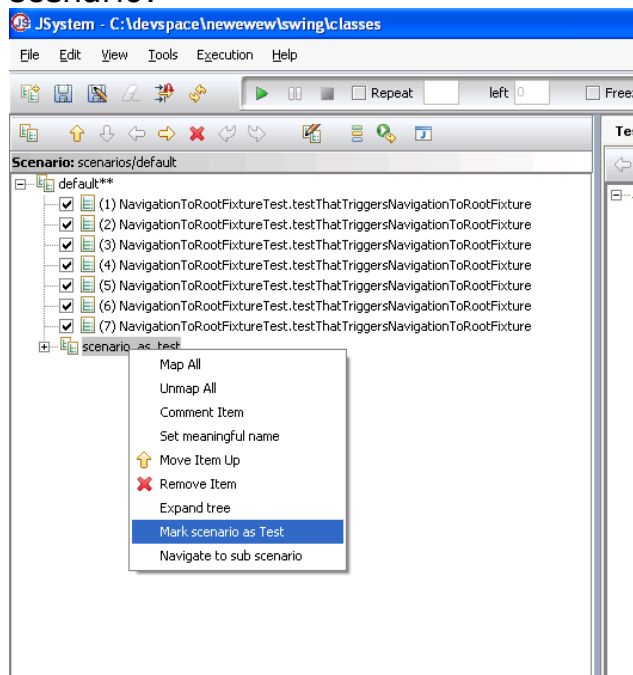
When marking scenario as a test when it is the root scenario, the scenario will be marked as a test in the assets tree:



And when adding the scenario as a sub scenario it will be marked as test by default.

If scenario was added to a parent scenario before marking it as a test, it's status in the parent won't be changed.

Scenario can be marked as a test when it is a sub scenario of another scenario:

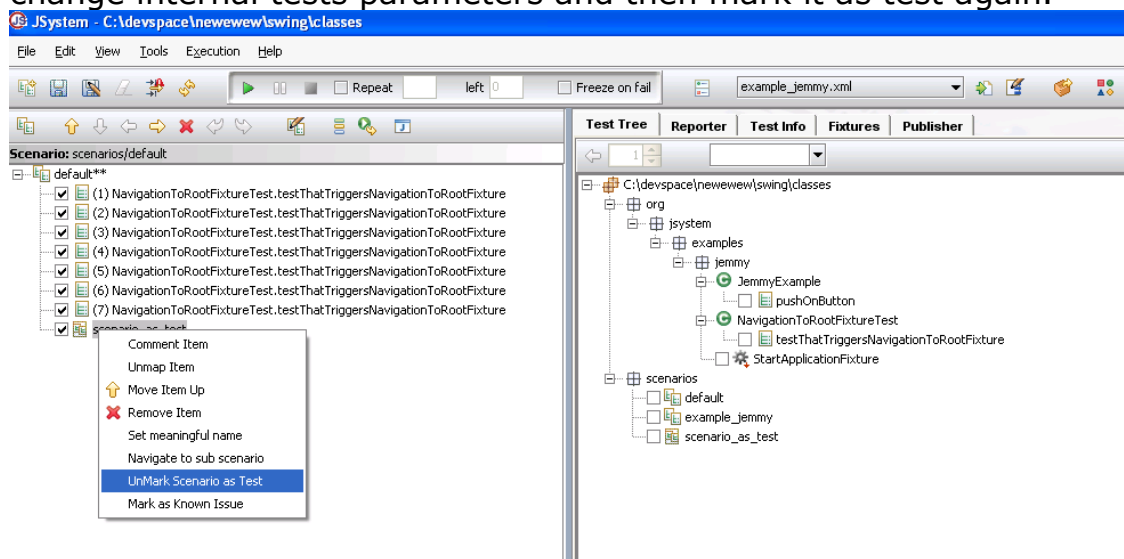


In this case, the scenario will be marked as a test only under the specific master scenario.



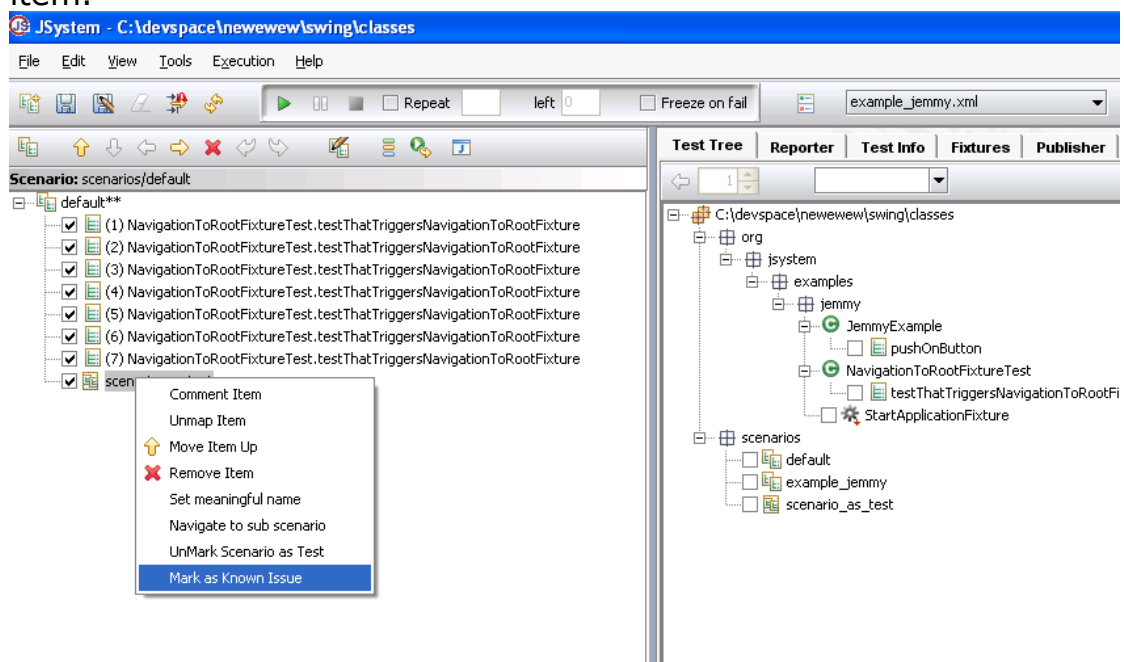
2.2.2 Changing “Scenario as Test” Internal Steps Parameters

While working on the scenario as test, the user can un-mark it as test, change internal tests parameters and then mark it as test again.



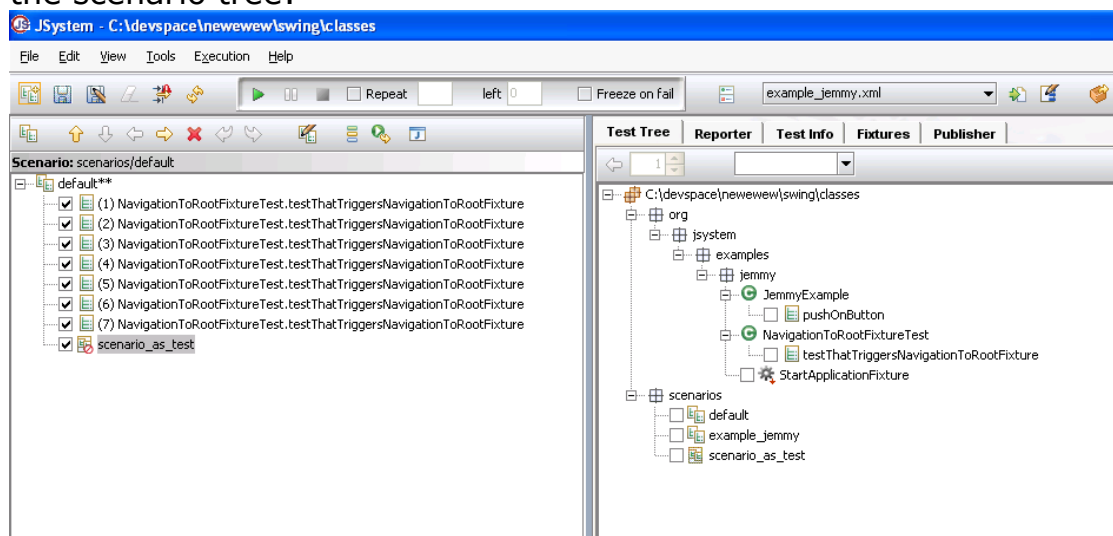
2.3 Mark Test as Known Issue

Sometimes a test fails due to a bug /problem which is a known issue, and the user do not wish to see the failure each time she runs the scenario. To set test as a known issue select the test (or scenario marked as test) right click and select the “Mark as known issue” menu item.





When marked as a known issue a test has a different presentation in the scenario tree:



When executing a test which is marked as a known issue, if the test fails it will be shown in the JRunner and in the reporter with a warning indication. If it passes, it will be shown with a success indication.



3. Execution from command line

For easy execution of JSystem scenarios from nightly build systems we have created a batch script that executes scenario ant script from command line without invoking runner GUI..

The name of the script is runScenario (.bat and .sh).

The script gets two arguments:

1. Automation project 'classes' folder path
2. Scenario name (without .xml extension)

Example:

runScenario.bat c:\jssystem\runner\projects\jssystemServices\classes scenarios/default

Note that jssystem.properties file should include the following properties:

currentScenario=scenarios/default (yes, it looks redundant, but it is required for correct execution)

sutFile=mySut.xml



4. Scenario Studio

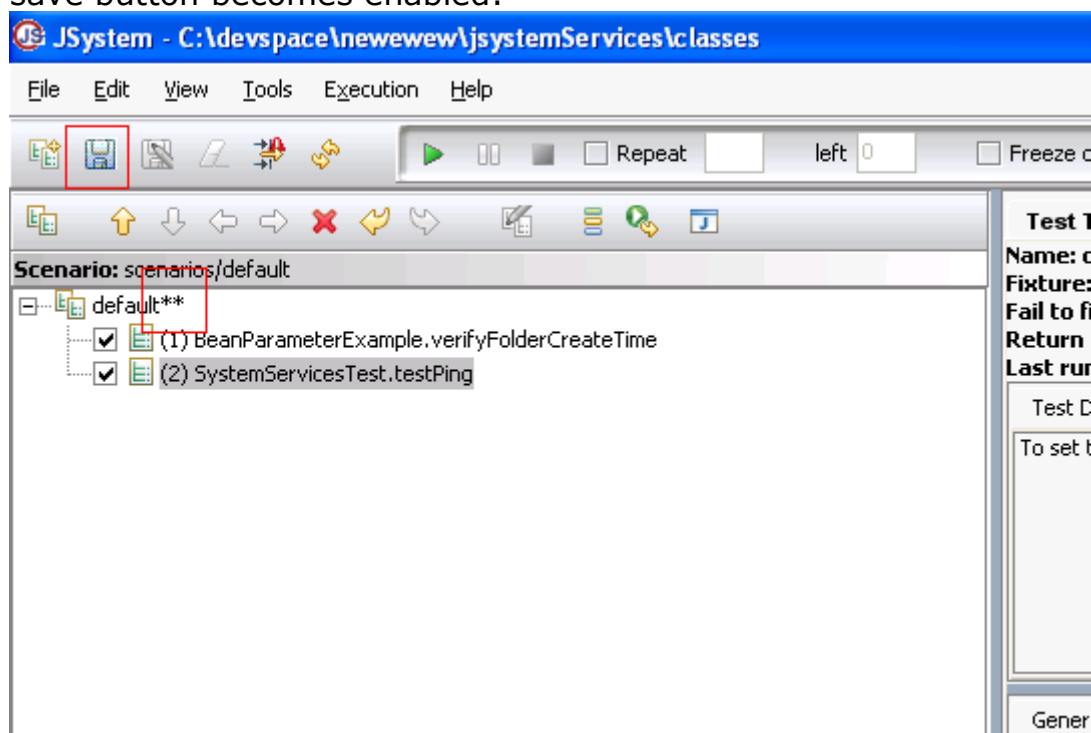
JSystem 5.4 introduces some changes in the Scenario Studio module of the JRunner that improve the work with it and give the QA engineers that work with it more programmatic capabilities.

4.1 Saving Scenario File

Until JSystem 5.4 the scenario file was saved to the file system whenever an operation on the scenario was performed (change of parameter, add/remove of test etc').

With the new JSystem capabilities, scenario files are becoming larger and saving them upon each operation slows the work with the JRunner.

Starting at JSystem 5.4 a save button was added to the JRunner, when changing scenario the scenario is marked with asterisk and the save button becomes enabled:



Scenario is saved to the file system only when pressing on the save button.

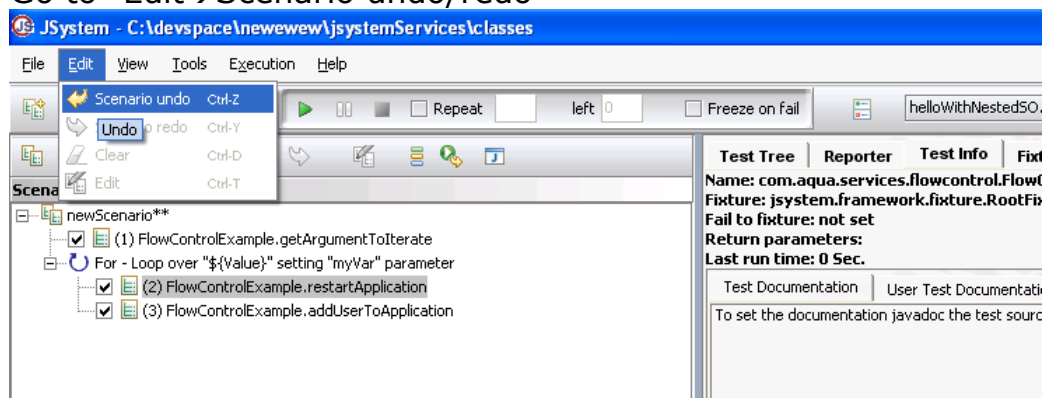
4.2 Undo/Redo

JSystem 5.4 supports undo/redo to operations done on tests parameters.

To perform undo/redo, do one of the following:



1. Go to “Edit→Scenario undo/redo”



2. Press on Ctrl-Z/Ctrl-Y

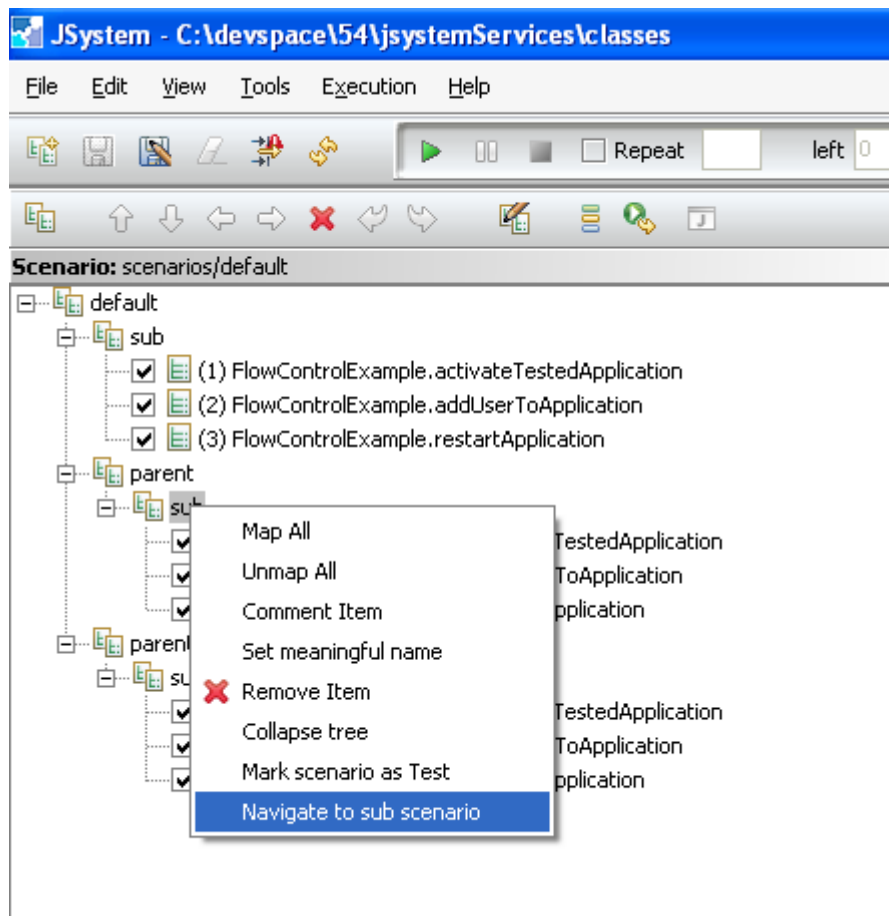


4.3 Easy Scenario Navigation

In order to improve the work with the runner when working on nested scenarios, we have improved the navigation between scenarios.

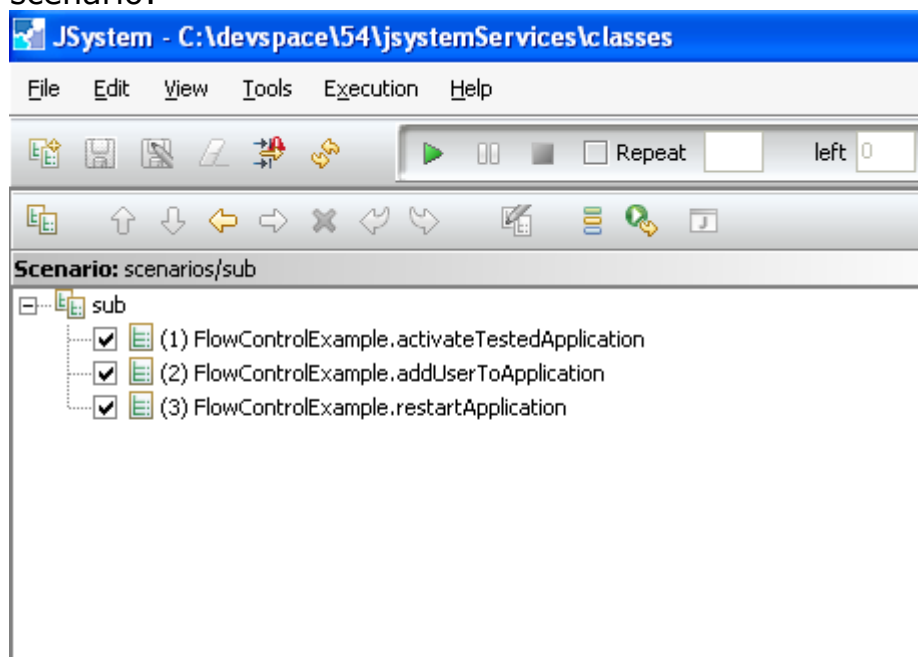
4.3.1 Navigating to sub scenario

To easily navigate to a sub-scenario, right click on sub-scenario and select the 'Navigate to sub scenario' menu item.



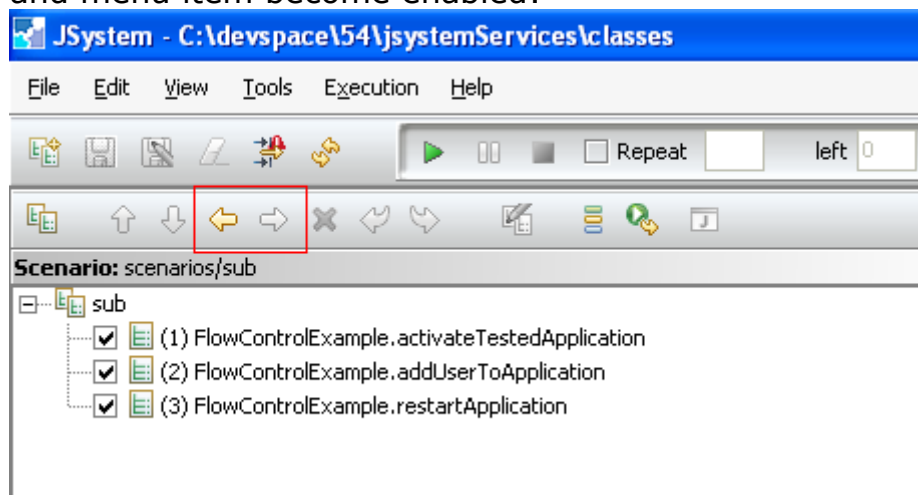


Once selecting the menu item, selected scenario will become the root scenario:



4.3.2 Navigating back and forward between scenarios

Once selecting a scenario (using the 'navigate to sub scenario' or by opening it using the 'Open' menu item) scenario navigation buttons and menu item become enabled:



You can now press on 'Previous Scenario' or 'Next Scenario' and easily navigate between scenarios.



4.4 Enhanced Test Parameters GUI

Until JSystem 5.4, JSystem supported predefined set of test parameters types.

These types included all java primitives, *java.io.File* object and *java.util.Date* object.

As more and more users use the runner as a development tool, the need to expose beans to the end users was raised. To attend this demand we have developed an infrastructure for developing custom UI elements for test parameters of any type.

Using this infrastructure we have implemented several GUI elements:

1. Multi option select dialog
2. Generic UI for java beans
3. Generic UI for java bean array

The following sections describe the above GUI elements and how to use them; the last section explains how to write your own parameter GUI provider.

4.4.1 List option select dialog

Until JSystem 5.4 when the test developer defined options for a parameter (using the *getXXOptions()* method) the user could select only one option. Starting at JSystem 5.4 the test author can define multi-select list.

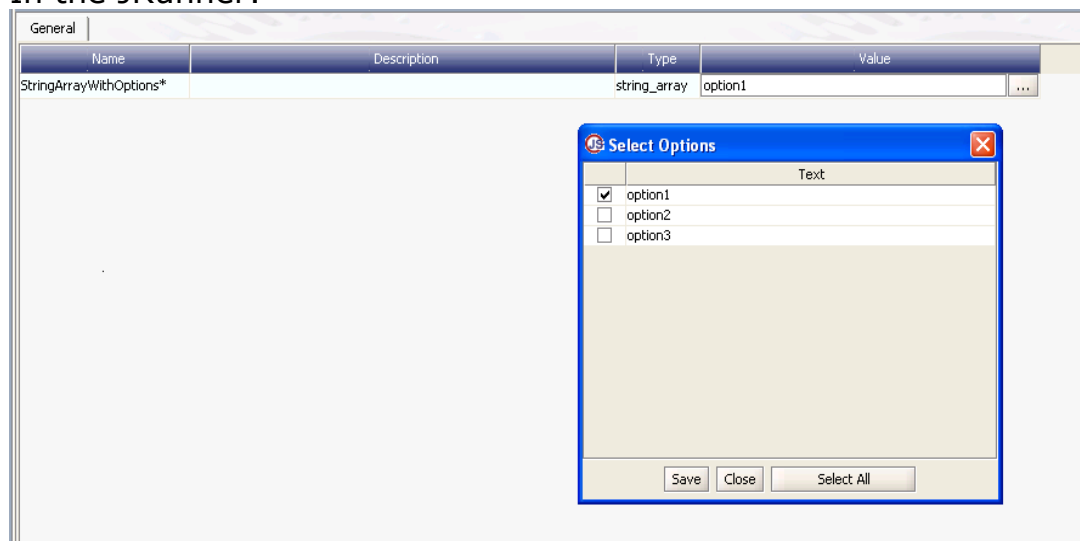
To make a test parameter a multi-select list, define it of type String array, and make sure it has an "options" method.

For Example:

```
package com.aqua.services.multiuser;
import junit.framework.SystemTestCase;
public class MultiSelectParameterExample extends SystemTestCase {
    private String[] multiSelect = {"option1"};
    public void testVerifyFolderCreateTime() throws Exception{
        for (String s:multiSelect){
            report.report("Val is " + s);
        }
    }
    public String[] getMultiSelectOptions() {
        return new String[]{"option1","option2","option3"};
    }
    public String[] getMultiSelect() {
        return multiSelect;
    }
    public void setMultiSelect(String[] stringArray) {
        this.multiSelect = stringArray;
    }
}
```



In the JRunner:



4.4.2 Generic UI for JavaBeans

JSystem 5.4 introduces a generic UI for custom JavaBeans written by test developers.

To associate a test parameters with the *enericObjectParameterProvider* perform the following:

1. Write your JavaBean and define a test parameter of bean type.
2. Add the `@UseProvider(provider=GenericObjectParameterProvider.class)` annotation to the setter method of the test parameter.

Bean Example Code:

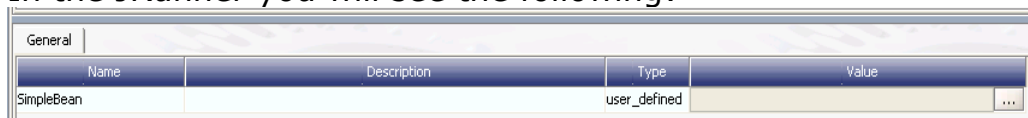
```
package com.aqua.services.multiuser;
public class SimpleBean {
    private String name;
    private String description;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}
```



Test Example Code:

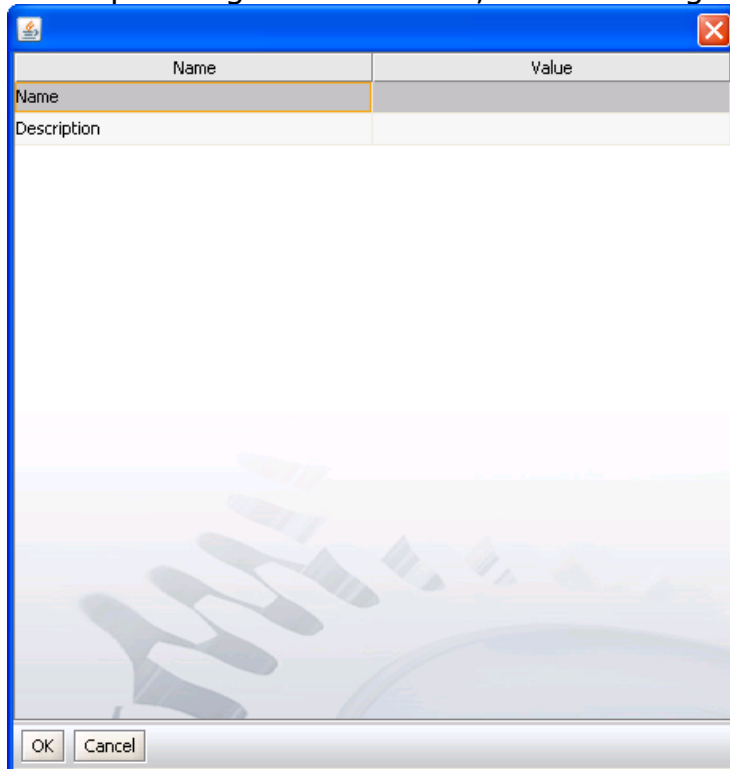
```
package com.aqua.services.multiuser;
import junit.framework.TestCase4;
import org.junit.Test;
/**
 */
public class BeanParameterExample extends SystemTestCase4 {
    private SimpleBean simpleBean;
    @Test
    public void verifyFolderCreateTime() throws Exception{
        //test code here
    }
    public SimpleBean[] getSimpleBean() {
        return simpleBean;
    }
    public void setSimpleBean(SimpleBean[] simpleBean) {
        this.simpleBean = simpleBean;
    }
}
```

In the JRunner you will see the following:



Name	Description	Type	Value
SimpleBean		user_defined	...

When pressing on the button, the following dialog is opened:



Name	Value
Name	
Description	

OK Cancel

4.4.3 JavaBean Array

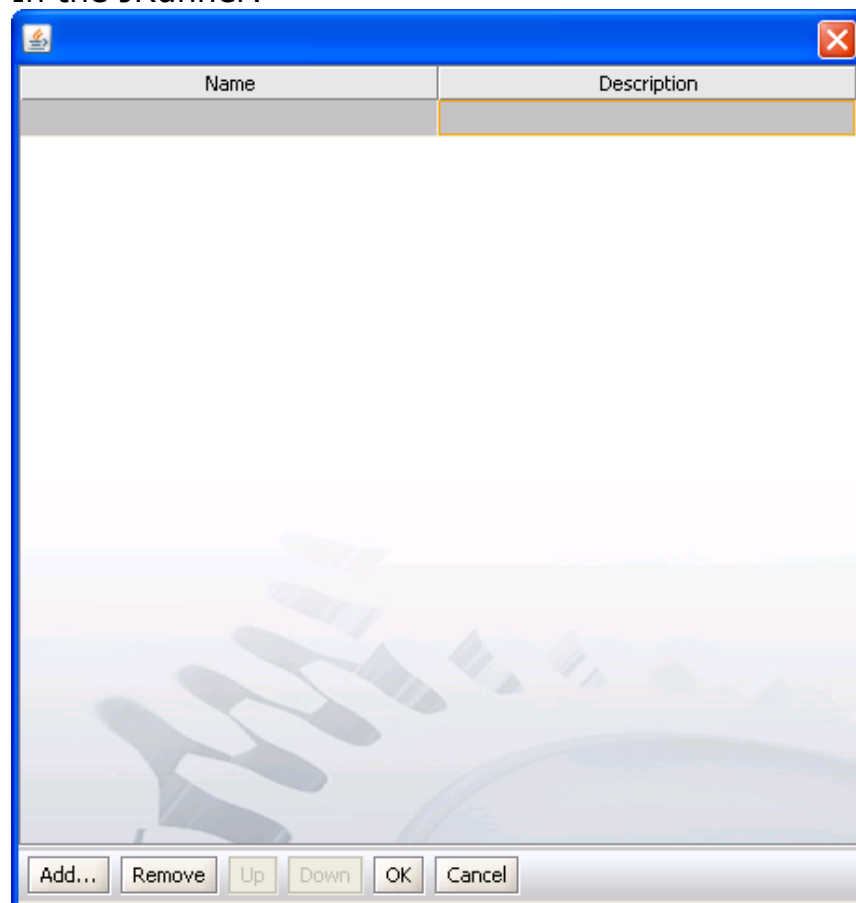
If the test parameter is of JavaBean array type, annotate parameter setter with *ObjectArrayParameterProvider.class*.



Test Code Example:

```
package com.aqua.services.multiuser;
import junit.framework.TestCase4;
import org.junit.Test;
/**
 */
public class BeanParameterExample extends SystemTestCase4 {
    private SimpleBean[] simpleBeanArr;
    @Test
    public void verifyFolderCreateTime() throws Exception{
        //test code here
    }
    public SimpleBean getSimpleBean() {
        return simpleBeanArr;
    }
    @UseProvider(provider=ObjectArrayParameterProvider.class)
    public void setSimpleBean(SimpleBean simpleBean) {
        this.simpleBeanArr = simpleBean;
    }
}
```

In the JRunner:



Note that this is a dialog for managing an array of items; you can add/remove or change items position. The columns of the table are created dynamically according to the fields of the bean.



4.4.4 Custom Data Model for Bean Array Generic UI

The Array of Java Bean support for test parameters also allows defining custom data model class for the Bean. This allows custom code for fetching bean content when presented to user.

To define a custom data model class, add the *@TestBeanClass* annotation to your bean, as demonstrated in the following code example.

Bean Code:

```
package com.aqua.services.multiuser;
import jsystem.framework.TestBeanClass;
@TestBeanClass(include={"name", "description"},
               model=MyCellEditorModel.class)
public class SimpleBean {
    private String name;
    private String description;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}
```

The 'include' property notes the Bean properties that are managed by the data model, the 'model' property points to a class which implements the *jsystem.framework.scenario.ProviderDataModel* interface.



Data model example:

```
package com.aqua.services.multiuser;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import javax.swing.JTable;
import jsystem.extensions.paramproviders.BeanCellEditorModel;
import jsystem.framework.scenario.ProviderDataModel;
import jsystem.utils.beans.BeanElement;
import jsystem.utils.beans.CellEditorType;

public class MyCellEditorModel extends BeanCellEditorModel
    implements ProviderDataModel {

    private static final long serialVersionUID = 1L;

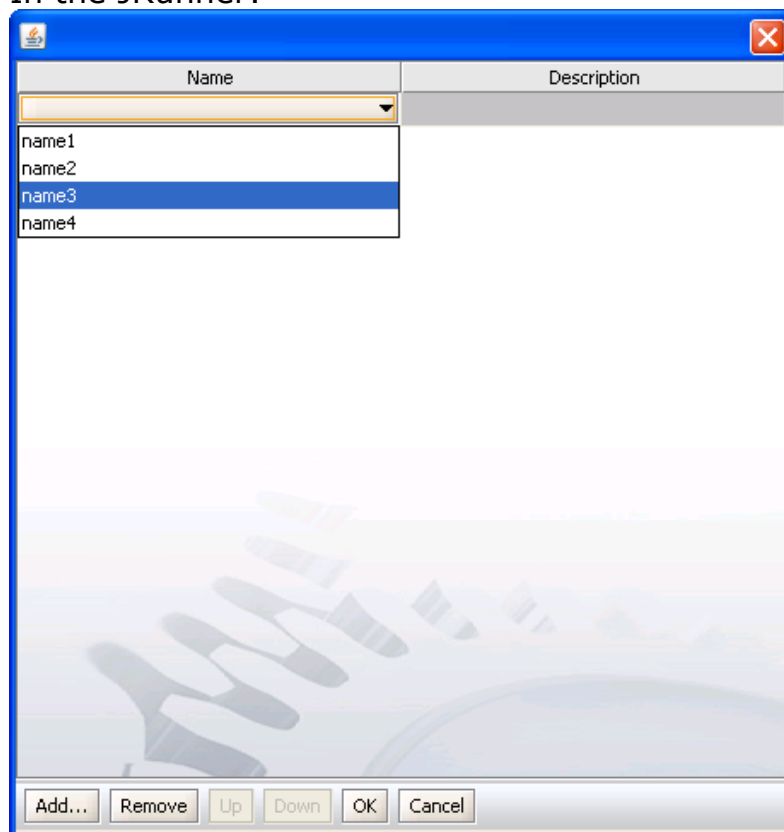
    public MyCellEditorModel(ArrayList<BeanElement>
        beanElements,
        ArrayList<LinkedHashMap<String,
            String>> multiMap) {
        super(beanElements, multiMap);
    }

    public CellEditorType getEditorType(JTable table,
        int row,
        int column) {
        String columnName = table.getColumnName(column);
        if(columnName.equals("Name")) {
            return CellEditorType.LIST;
        } else {
            return super.getEditorType(table, row,
                column);
        }
    }

    @Override
    public String[] getOptions(JTable table,
        int row,
        int column) {
        String columnName = table.getColumnName(column);
        if(columnName.equals("Name")) {
            return new
                String[] {"name1", "name2", "name3", "name4"};
        } else {
            return super.getOptions(table, row, column);
        }
    }
}
```



In the JRunner:



4.5 Writing your own UI parameter provider

Users can use the infrastructure which is used by JSystem to develop their own providers.

In order to develop a provider, do the following:

1. Implement the interface
jssystem.framework.scenario.ParameterProvider
2. Use the `@UseProvider` to annotate the setter method of the test parameter.

To learn more about implementing *ParameterProvider*, see interface documentation and the built-in implementations:

jssystem.extensions.paramproviders.StringArrayOptionsParameterProvider



5. Reporter

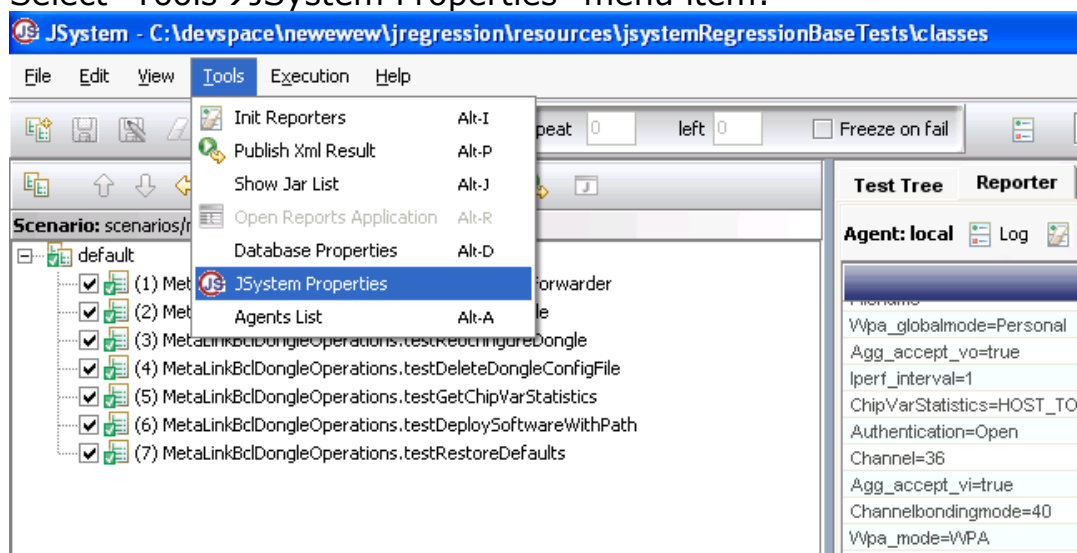
JSystem 5.4 includes some changes that make the HTML reporter more friendly and readable.

5.1 Parameters into Level

When your test includes a large number of parameters, test report in reporter tab and in the html report, is not easy to read. To mitigate this problem the *log.parameters.in.level* property was added. When this property is set to true, test report shows a link to test parameters. When browsing to link parameters are shown.

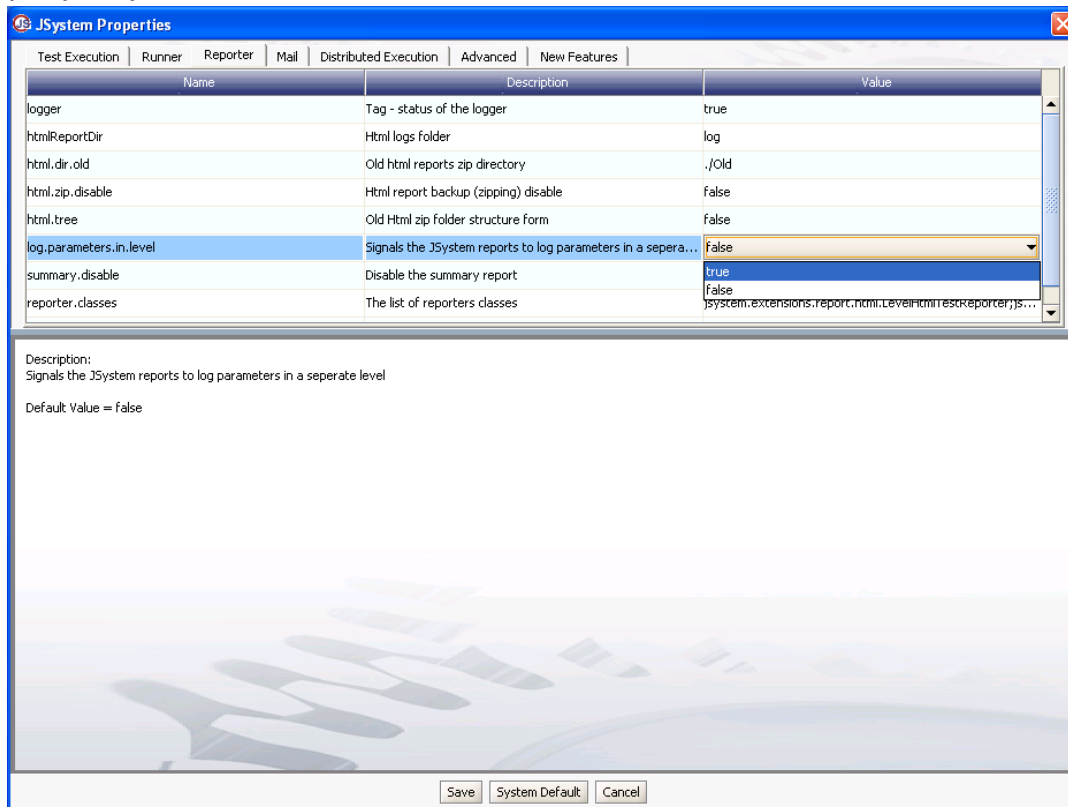
To set *log.parameters.in.level*, perform the following:

Select “Tools→JSystem Properties” menu item:

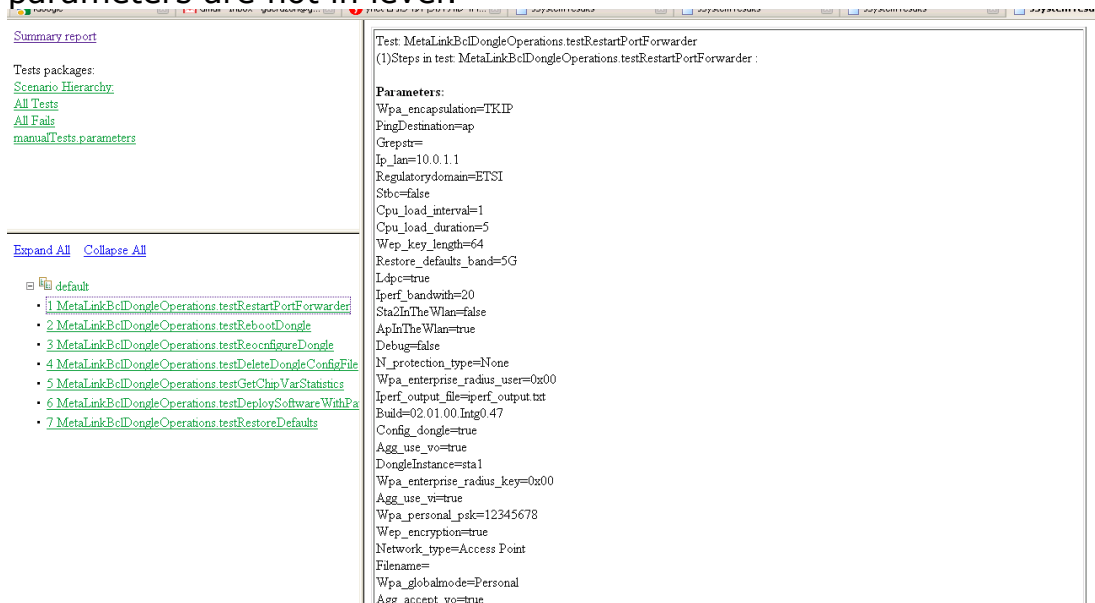




Select “Reporter” tab and change the *log.parameters.in.level* property:



Here is how the test HTML reporter looks like when the test parameters are not in level:





When test parameters are leveled:

[Summary report](#)
Tests packages:
[Scenario Hierarchy](#)
[All Tests](#)
[All Fails](#)
[manualTests.parameters](#)

[Expand All](#) [Collapse All](#)
default

- 1 [MetaLinkBcIDongleOperations.testRestartPortForwarder](#)
- 2 [MetaLinkBcIDongleOperations.testRebootDongle](#)
- 3 [MetaLinkBcIDongleOperations.testReconfigureDongle](#)
- 4 [MetaLinkBcIDongleOperations.testDeleteDongleConfigFile](#)
- 5 [MetaLinkBcIDongleOperations.testGetChipVarStatistics](#)
- 6 [MetaLinkBcIDongleOperations.testDeploySoftwareWithPa](#)
- 7 [MetaLinkBcIDongleOperations.testRestoreDefaults](#)

Test: MetaLinkBcIDongleOperations.testRestartPortForwarder
(1)Steps in test: MetaLinkBcIDongleOperations.testRestartPortForwarder :
Test parameters
Parameters end.
[test code](#)
09:55:37: setUp execution
09:55:37: tearDown execution
Start time: Mon Jan 26 09:55:37 IST 2009
End time : Mon Jan 26 09:55:37 IST 2009
Test running time: 0 sec.

When pressing on link:

[Summary report](#)
Tests packages:
[Scenario Hierarchy](#)
[All Tests](#)
[All Fails](#)
[manualTests.parameters](#)

[Expand All](#) [Collapse All](#)
default

- 1 [MetaLinkBcIDongleOperations.testRestartPortForwarder](#)
- 2 [MetaLinkBcIDongleOperations.testRebootDongle](#)
- 3 [MetaLinkBcIDongleOperations.testReconfigureDongle](#)
- 4 [MetaLinkBcIDongleOperations.testDeleteDongleConfigFile](#)
- 5 [MetaLinkBcIDongleOperations.testGetChipVarStatistics](#)
- 6 [MetaLinkBcIDongleOperations.testDeploySoftwareWithPa](#)
- 7 [MetaLinkBcIDongleOperations.testRestoreDefaults](#)

Test parameters
Parameters:
Wpa_encapsulation=TKIP
PingDestination=ap
Grepstr=
Ip_lan=10.0.1.1
Regulatorydomain=ETSI
Stbc=false
Cpu_load_interval=1
Cpu_load_duration=5
Wep_key_length=64
Restore_defaults_band=5G
Ldpc=true
Iperf_bandwidth=20
Sta2InTheWlan=false
ApInTheWlan=true
Debug=false
N_protection_type=None
Wpa_enterprise_radius_user=0x00
Iperf_output_file=iperf_output.txt
Build=02.01.00.Intg0.47
Config_dongle=true
Agg_use_vo=true
DongleInstance=sta1
Wpa_enterprise_radius_key=0x00
Agg_use_vo=true
Wpa_personal_psk=12345678
Wep_encryption=true
Network_type=Access Point
Filename=
Wpa_globalmode=Personal
Agg_accept_vo=true

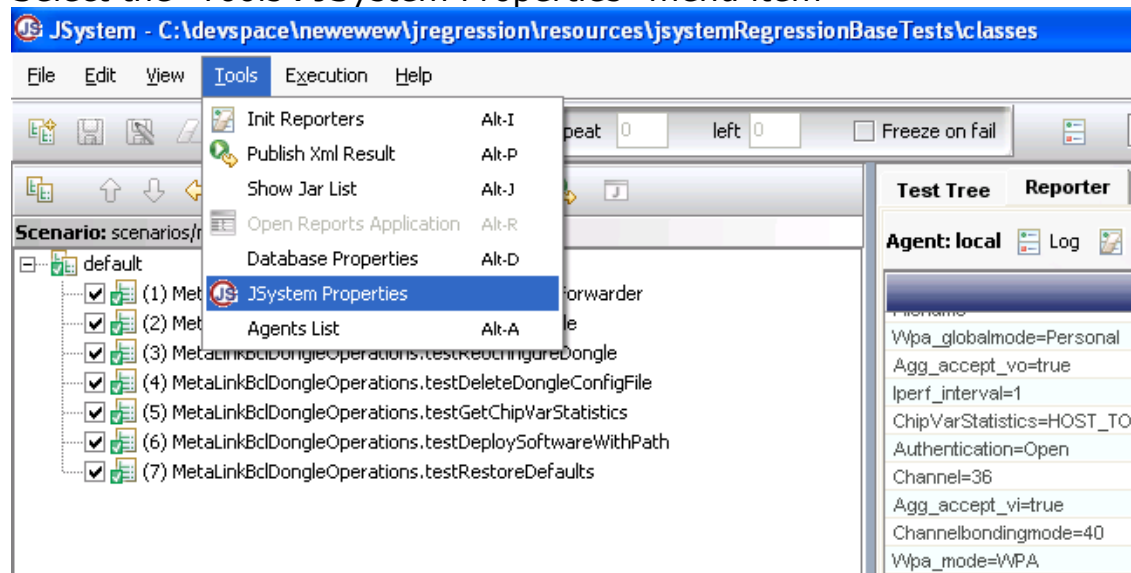


5.2 Errors HTML reporter

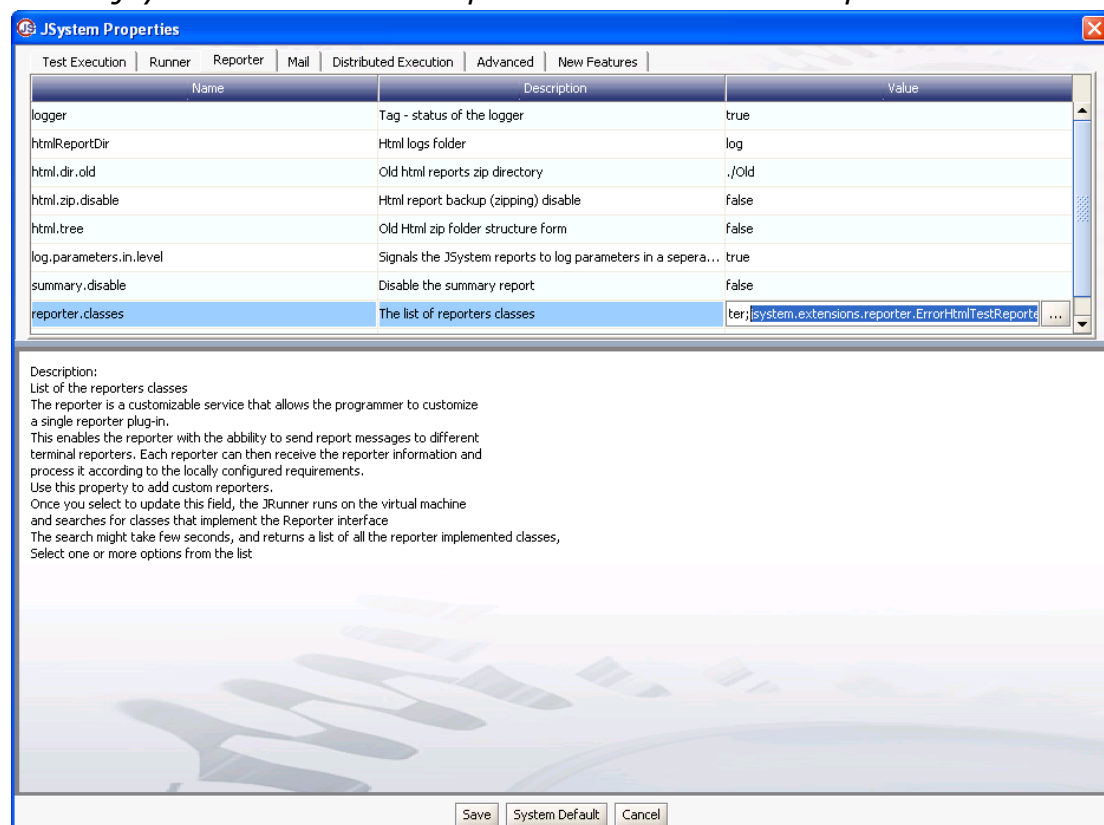
The errors html report is an add-on reporter that shows only failed tests, and in the test level, only the error messages. The errors HTML reporter add-on was added to make it easier to pinpoint the problems.

To activate the Errors HTML reporter perform the following:

Select the “Tools→JSystem Properties” menu item



Select “Reporter” tab and add to the property ‘reporter.class’ the class: *jsystem.extensions.reporter.ErrorHtmlTestReporter*





When viewing execution HTML report a new run property is added to the standard HTML report:

[Summary report](#)
Tests packages:
[Scenario Hierarchy:](#)
[All Tests](#)
[All Fails](#)
[manualTests.parameters](#)
[regression.generic](#)

[Expand All](#) [Collapse All](#)

📁 default

JSYSTEM summary report

General statistics	
Number of tests	13
Number of fails	6
Number of warnings	0
Running time	12 sec.

Run properties	
Date	Mon Jan 26 11:00:02 IST 2009
Station	10.0.0.118
Error report	Error Report
Scenario	default
Version	unknown
User	goland

Tests statistics			
test name	passed	failed	warning
MetaLinkBcdDongleOperations.testRestartPortForwarder	1	0	0
MetaLinkBcdDongleOperations.testRebootDongle	1	0	0

When pressing on the link Errors HTML report is opened:

[Tests packages:](#)
[Scenario Hierarchy:](#)
[All Fails](#)

[Expand All](#) [Collapse All](#)

📁 default

- 8 [GenericBasic.testFailWithError](#)
- 9 [GenericBasic.testFailWithError](#)
- 10 [GenericBasic.testFailWithError](#)
- 11 [GenericBasic.testFailWithError](#)
- 12 [GenericBasic.testFailWithError](#)
- 13 [GenericBasic.testFailWithError](#)

[11:00:13: Fail. Some error found](#)
[11:00:13: Failure in Test regression.generic.GenericBasic.testFailWithError](#)