Chapter 14

Trouble Shooting Guide

➢ **In this chapter…**

14.1 Trouble Shooting Guide Overview

The trouble shooting guide provides solutions for common errors and problems, offering solutions that help the user improve and streamline JSystem operations and functionality.

14.1 JRunner

The following section details common problems encountered when using the JRunner application.

Long JRunner Start up	
Operating System	All
Symptoms	1. Launching the JRunner takes a long time; the JRunner appears to be stuck. 2. Sometimes after a while an "**OutOfMemoryError**" is dumped to the JRunner console.
Solution	In most cases the reason for the problem is that the "**tests.src**" and "**tests.dir**" properties in the "**jsystem.properties**" file point to an invalid path. Delete the properties and reload the JRunner. When the JRunner starts, it will ask the user to choose a tests project again.

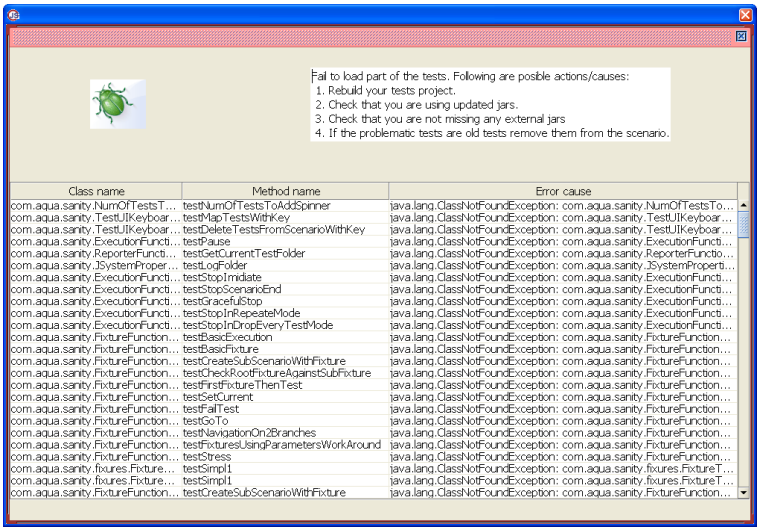
Table 1: Long JRunner Start up

Slow File Browse when Selecting a Scenario	
Operating System	Windows
Symptoms	The file browser component works very slow when selecting a scenario or creating a new scenario.
Solution	Add "**regsvr32 /u %windir%\system32\zipfldr.dll**" to your "**run.bat**" file It will disable the "**zipfldr**" dll of windows which causes this problem. ***Note: When the dll is being disabled - a popup message appears, approve it in order to run the JRunner.***

Table 2: Slow File Browse when Selecting a Scenario

SUT Load Error when Starting the JRunner	
Operating System	All
Symptoms	When opening the JRunner, the following error message appears in the command line console: **WARNING: Unable to init sut with file: windowsGeneral.xml The element type "InterprocessManager" must be terminated by the matching end-tag"</InterprocessManager>"**
Solution	1. The error message in the console shows the SUT file name. Open the SUT file with any text/xml editor and correct it. 2. The "**sutFile**" property in the jsystem properties points to an invalid SUT file. Delete the sutFile property and re-launch the JReporter.

Table 3: SUT Load Error when Starting the JRunner

Test Load Problem	
Operating System	All
Symptoms	The following error dialog is shown when launching the JRunner or refreshing it 

Solution	The reason for this error dialog is that the JRunner fails to load test classes using java reflection. This problem occurs when renaming the name of a class or test method in the class. The scenario retains the old name, when the scenario loads it tries to create an instance of the test with the old name and fails. To solve the problem, re-add the test to the scenario. Another common reason for this problem is that the "**classes**" folder of the automation project is empty. This usually happens when "**Eclipse**" cleans the project when it is being compiled. In order to solve this problem open Eclipse, select Project→Clean, select the automation project and the "**build immediately**" check box and then apply the operation. Eclipse compiles the tests and classes, the folder will be filled with compiled classes.

Table 4: Test Load Problem

ClassNotFound Exception During Test Runs	
Operating System	All
Symptoms	When running a scenario, the "**ClassNotFound**" exception is dumped to the JRunner console.
Solution	If tests were loaded correctly by the JRunner and the problem occurs when running the scenario, the reason is that Eclipse cleaned the project during work with JRunner. In order to solve the problem, re-compile the tests, refresh the JRunner and run the project again.

Table 5: ClassNotFound Exception during Test Runs

14.2 Development Environment

The following section details common errors and problems that occur when using the 3rd party development environment.

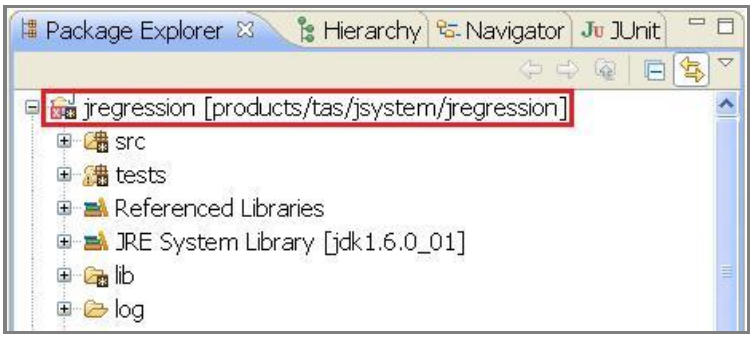
Project Error in Eclipse	
Operating System	All
Symptoms	Eclipse marks a project in red: 
Solution	This problem is caused by one of the following reasons: 1. Class path problems - In order to correct this problem, right click on the project and then select Properties→Java Build Path. The build path dialog shows the "**problematic jar**". If you do not see an obvious problem, select one jar, and remove it, apply the operation, reopen the Java Build path dialog, re-add the jar and again apply the operation. 2. Eclipse is attempting to build the project but fails to do so because the JRunner retains one of the files. In order to solve this problem, close the JRunner, and go to Project→Clean and re-build the project.

Table 6: Project Error in Eclipse

Problems with Build Fails due to JavaDoc Compiler	
Operating System	All
Symptoms	Building system object using the ant fails due to problems in activating javadoc compiler.

Solution	Javadoc compiler has to be in the path of the operating system.

Table 7: Problems with Build Fails due to JavaDoc Compiler

14.3 Problems with the Initiation of System Objects

1. First, verify that the text that is being identified by the system object in the "**system.getSystemObject**" method is identical to the name of the system object entity in the SUT file.

   Important Note: Make sure you trim white spaces.

   Code Example

```
"(MyStation)system.getSystemObject("my_station")
```

   Table 8: System Object Initiation Code Example

   SUT File

```
<sut>
  <my_station>
    <class>com.jsystem.quickstart.MyStation</class>
    <host>10.0.0.2</host>
    <userName>simpleuser</userName>
    <password>simpleuser</password>
  </my_station>
</sut>
```

   Table 9: System Object Initiation SUT Code Example

   If the user calls the system, a system object called "**my_station**" makes sure that in the SUT file there is an entry under the <sut> entity called "**my_station**"

2. Make sure the name of the class is accurate.
   SUT File Code Example.

```
<sut>
  <my_station>
    <class>org.jsystem.quickstart.MyStation</class>
    <host>10.0.0.2</host>
    <userName>simpleuser</userName>
    <password>simpleuser</password>
  </my_station>
</sut>
```

   Table 10: Accurate SUT File Code Example

The framework will look for a class "org.jsystem.quickstart.MyStation". The full name of a class is composed of package name ("org.jsystem.quickstart") and class name ("MyStation") Make sure that the full name of the class is accurately written and that the class indeed exists, that is the class "MyStation" exists under the package "org.jsystem.quickstart"

3. If the user has implemented a constructor that gets arguments, make sure to add a default constructor. (Constructor that does not get any argument). For example, this is a constructor that gets arguments:

```
01 package org.jsystem.quickstart;
02 import jsystem.framework.system.SystemObjectImpl;
03 public class MyStation extends SystemObjectImpl {
04             public MyStation(String host,String user,String
 password){
05              …
06             }
07 }
```

Table 11: Get Constructor Argument

If a constructor has been added to the system object make sure to add a default constructor.

```
01 package org.jsystem.quickstart;
02 import jsystem.framework.system.SystemObjectImpl;
03 public class MyStation extends SystemObjectImpl {
04             public MyStation(){
05              …
06             }
07             public MyStation(String host,String user,String password
){
08              …
09             }
10 }
```

Table 12: Adding Constructors