Test terminal-based applications automatically with Rational Functional Tester

How to use the Extension for Terminal-based Applications

Shi Ming Qu

September 13, 2011

Many software applications that support multiple platforms provide console interfaces. A quality assurance engineer or software tester always needs to use the console by typing on a terminal emulator keyboard. The IBM® Rational® Functional Tester Extension for Terminal-based Applications helps you create test scripts to automate host application test cases. This article explains how to use this tool, so you can make terminal-based application testing not only easier but automatic.

Why terminal-based applications are still important

Software applications are continually becoming more intricate in their user interfaces. But there still are lots of applications that are terminal-based, and these applications play an important role in the software industry, because they are always are used on servers.

A terminal-based application is often designed for platforms other than Microsoft Windows, such as Linux, IBM® AIX®, and Solaris. It has no user interface, so the user interacts with the application by entering a command, which returns text output directly. Compared to GUIs, these applications are lightweight, require fewer system resources, and launch faster.

Terminal-based applications can be launched locally or remotely. Therefore, you can log onto a system remotely, using a terminal emulator, and then launch the application by entering commands in the terminal. The application is very easy to control remotely as long as a terminal emulator is installed on the system.

Overview of the Terminal-based Extension

IBM® Rational® Functional Tester is an object-oriented automated functional testing tool. It is well-known for GUI automation testing. Version 8.0 provides a new Terminal-Based Extension tool that can help you test terminal-based applications. It supports functional testing for these types of terminals:
This tool is like a terminal emulator, in that it can act as a client to connect to the host. It’s embedded in the testing software and based on Java, so it is very easy to automate testing with Rational Functional Tester.

**Launch the Terminal-based Extension tool**

There are three ways to launch the extension tool.

**Use the icon**

You could click the launch icon on the toolbar, as shown in Figure 1.

**Figure 1. Launch the extension tool, example 1**

![Figure 1](image1)

**Use the menus**

You could use the menus to launch the Application Configuration Tool:

1. Click **Configure** in the menu bar, and then select **Configure Applications for Testing** from the drop-down menu (Figure 2).
2. Select and run the extension by selecting **Extension for Terminal Applications** from the list of applications, and then clicking **Run** (Figure 3).

**Figure 2. Launch the extension tool, example 2**

![Figure 2](image2)
Use the automation code

You could launch the extension tool in your automation code. You can add the code in Listing 1 in your automation to launch the tool during your automated test.

**Listing 1. Launch the extension tool by using code, example 3**

```java
import com.rational.test.ft.script.RationalTestScript;
RationalTestScript.startApp("Extension for Terminal Applications");
```

This code calls `startApp` of the `RationalTestScript` class. The parameter is the name of the tool. Notice that the name is case-sensitive.

**Tip:**
If you want to develop your automation to test console applications, this is the best choice for launching the extension.

Use the Rational Functional Tester extension

After the extension tool is launched, you can see the window shown in Figure 4.
The top pane in this window is the connection configuration information. The terminal view is in the bottom pane. The function keys are displayed as buttons along the bottom. You can enter the host address, select the terminal type, specify a connection port, select a code page, and choose screen size from the top pane. When you connected to a host, the information will show in the black console view of the bottom pane, and the bottom pane will expand automatically, as Figure 5 shows.
The most common use of this tool is to connect to a host. You can store, load, and share common host configurations using a property file, and you can use scripts to load the connection configuration. You also can work with multiple terminals at the same time.

**Control the extension tool automatically**

Rational Functional Tester is a very good tool for developing and testing GUI automation, especially for Java-based applications. It can recognize all of the objects in UI. You can get the test objects by using the code in Listing 2.

**Listing 2. Code to get a test objects**

```java
RootTestObject root = RationalTestScript.getRootTestObject();
TestObject[] o = root.find(SubitemFactory.atDescendant( property, value ));
```

The test object map in Rational Functional Tester is tree structure, and it is singleton pattern. This means that there is only one RootTestObject, which is the parent of all other TestObjects. Therefore, you need to get the RootTestObject, and then use it to get the descendent TestObjects.

The first line of code in Listing 2 calls getRootTestObject. All TestObjects have properties and values so that you can identify each TestObject by its specific properties and values. The second line in Listing 2 calls the `find` method to get the TestObjects by property and value.
However, a TestObject can have lots of properties, so you need to choose one or more properties to uniquely describe the object in order to find it. Rational Functional Tester provides a tool to get the properties and values of TestObjects.

In the screen capture in Figure 6, you can see that the tool gets the properties and the values listed in a table. You might wonder whether all the properties are listed. The answer is "most of the time," but not all the properties are displayed by this tool.

**Figure 6. Getting TestObject properties**

![Image of TestObject properties](image)

If you want to see all properties, you could print all of the properties and values by using code, as the example in Listing 3 shows.

**Listing 3. Code to print all properties and values of a TestObject**

```java
TestObject o = getTheObject();
Object[] keys = o.getProperties().keySet().toArray();
for (Object key : keys) {
    System.out.println(key.toString() + " : " + o.getProperty(key.toString()));
}
```

In Listing 3, the code calls getTheObject, returning the instance of the TestObject, and then it gets all keys and converts them to an array. In the `for` loop, it prints all of the properties and values. By analyzing the results, you can decide which properties best describe the TestObject, so you can use the `find` method to get the test object.

Now that you can get the TestObjects, you need to simulate keyboard or mouse actions on these TestObjects. In Rational Functional Tester, all TestObject interaction methods are available from...
the GuiTestObject class, so you can simulate keyboard actions or mouse actions by using this class. After you get the TestObject you can create the GuiTestObject, as shown in Listing 4.

**Listing 4. Clicking on a TestObject**

```java
TestObject o  = getTheTestObject();
new GuiTestObject(o).click();
```

As you know, there is no GUI for console applications. Users interact with terminal-based applications by inputting text and then reading the text feedback from the console. If you want to interact with console applications in automation code, you need to make that code input commands to the terminal and get the text feedback from the terminal.

As mentioned previously, GuiTestObject can simulate keyboard actions. So if you have found the TestObject, you can create the GuiTestObject, which then allows you to input commands to the terminal. It's very easy to get.

Now let's see how to get the feedback from the terminal. Although you might need to get each line of the output, by default, you can get only the last line of the output in terminal. To get the output of every single line in console, in the `ivory.properties` file, which is in the `Rational Functional Tester\install directory\FunctionalTester\bin` location, you need to set this flag:

```java
rational.test.ft.fte.playback.vt_row_vp = true
```

By default, this flag is set to `false`. If you set the flag to `true`, you can get the TestObject of every single line. Then you can get the characters of each line by the row object's `char` property, as shown in Figure 7.
Figure 7. Getting the terminal output of one row

Tips for interacting with multiple terminals

To interact with multiple terminals, the key point is to be able to identify each terminal window. No matter how you launched the extension terminal, all of the properties of the terminal window are the same. However, the extension tool can save and open connections in a file. If you open a connection from a file, the title of the terminal dialog will end with the file name, as shown in Figure 8.
Figure 8. Opening a terminal connection by using a file

There is a property named captionText for the terminal dialog. By using this property, you can indicate in your code which console dialog you want to use for the test.

Other ways to use the extension tool

In this article, you learned how to use the Rational Functional Tester Extension for Terminal-based Applications to automate testing applications with a console UI. This tool can be widely used, not only for the automation testing but also as a terminal tool or for automated configuration of operation systems.
Related topics

- Find out more about Rational Host Access Transformation Services, Version 8:
  - Visit the product page.
  - Browse the HATS HotSpot for links to technical articles and many related resources.
  - See the Rational Host Access Transformation Services V8.0 - Detailed System Requirements page For detailed information about prerequisites and operating environments.
- Get the free trial download for Rational Host Access Transformation Services.
- Evaluate IBM software in the way that suits you best.
- Get more details about the Rational Functional Tester Extension for Terminal-based Applications.
- Visit the Rational Functional Tester area on developerWorks for introductory to in-depth information.
- Explore the Rational Functional Tester Information Center, where you can also take a short video tour.
- Investigate Rational Functional Tester Plus, which is a software application testing bundle.
- Improve your skills. Check the Rational training and certification catalog, which includes many types of courses on a wide range of topics. You can take some of them anywhere, any time, and many of the "Getting Started" ones are free.
- Try Rational Functional Tester free, requires registration.

© Copyright IBM Corporation 2011
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)