[Skip to content](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html" \l "container)

* [*Safari*](https://www.safaribooksonline.com/home/)
* [Recommended](https://www.safaribooksonline.com/r/)
* [Queue](https://www.safaribooksonline.com/s/)
* [Search](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)
* [Expand Nav](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)
  + [Recent](https://www.safaribooksonline.com/recent/)
  + [Topics](https://www.safaribooksonline.com/t/)
  + [Highlights](https://www.safaribooksonline.com/u/p253751/)
  + [Settings](https://www.safaribooksonline.com/u/)
  + [Feedback](mailto:feedback@safaribooksonline.com)
  + [Sign Out](https://www.safaribooksonline.com/accounts/logout/)
  + [Settings](https://www.safaribooksonline.com/u/)**10** days left in your trial. [Subscribe](https://www.safaribooksonline.com/subscribe/).
  + [Feedback](mailto:feedback@safaribooksonline.com)
  + [Sign Out](https://www.safaribooksonline.com/accounts/logout/)

[Software Test Engineering with IBM Rational Functional Tester: The Definitive Resource](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)

* [Search in book...](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)

Top of Form

Bottom of Form

* [Toggle Font Controls](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)
* [Share this](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)
  + [Twitter](https://twitter.com/share?url=http://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch04.html&text=Software%20Test%20Engineering%20with%20IBM%20Rational%20Functional%20Tester%3A%20The%20Definitive%20Resource&via=safari)
  + [Facebook](https://www.facebook.com/sharer/sharer.php?u=http://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch04.html)
  + [Google Plus](https://plus.google.com/share?url=http://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch04.html)
  + [Email](mailto:?subject=Safari:%20Chapter%204.%20XML%20and%20Rational%20Functional%20Tester&body=http://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch04.html%0D%0Afrom%20Software%20Test%20Engineering%20with%20IBM%20Rational%20Functional%20Tester%3A%20The%20Definitive%20Resource%0D%0A)

[Prev](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch10.html)

[Chapter 10. Advanced Scripting with Rational Functional Tester TestObjects](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch10.html)

[Next](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch12.html)

[Chapter 12. Extending Rational Functional Tester with External Libraries](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch12.html)

**Chapter 11. Testing Specialized Applications**

**Fariz Saracevic, Lee B. Thomas**

*Rational Functional Tester is capable of testing a large variety of application types. In most cases, you use Rational Functional Tester in exactly the same manner, even though the applications can be built using different technologies. However, in some cases, you need to provide a lot of capability out of the box (for example, Swing, SWT, HTML, .NET, and so on) because there are instances in which capabilities need to be extended. This chapter helps you understand how to take advantage of these extended capabilities to test applications that are terminal-based or are written using the SAP, Siebel, or Adobe Flex technologies.*

**Why Rational Functional Tester Won’t Test Every Application**

Just like other automated testing tools, Rational Functional Tester is somewhat at the mercy of the application. If the application is built in such a way that the internals of the application are exposed, then Rational Functional Tester can interact with it in a rich way. If, on the other hand, the internals are hidden, then Rational Functional Tester can do little when recording its interactions with the application, and playback is likely to fail. In such a case, playback is likely to fail even when the ScriptAssure™ settings are adjusted because the original recording might have ambiguous references to objects.

By way of analogy, imagine a visually impaired person trying to use a completely screen-based, self-service checkout at a store. It might be possible for the person to learn where the buttons are on the screen, carefully pressing at x, y coordinates, starting from some particular corner of the screen. However, if the application is changed and the location and size of the buttons change, the person needs to learn all over again. That’s the equivalent of throwing out your recorded Rational Functional Tester script and rerecording.

Contrast this with that same person using a standard 10-key keypad with a raised bump on the number 5. Even if the machine is replaced with a new model and the entire keypad is moved, the fact that the buttons are raised, the number and position of buttons are unchanged, and the raised bump contributes to help the person interact successfully with the machine.

In a similar way, Rational Functional Tester needs a special interface for some classes of applications. Otherwise Rational Functional tester would need to learn how to interact with these types of applications (for example, SAP, Siebel, and so on) by means of x, y coordinates. In all the cases in this chapter, an additional component is used along with Rational Functional Tester, and in all but the first one of the cases, something must be installed and configured prior to making a recording with Rational Functional Tester.

The technologies covered in this chapter are terminal-based applications (sometimes referred to as mainframe applications), SAP applications, Siebel applications, and applications built on Adobe Flex technology.

**Extension for Terminal-Based Applications**

Terminal-based applications are built on application frameworks that assume the end user has a terminal on his desk, not a desktop or laptop computer. The terminal is connected by a special cable to a mainframe or midrange computer. The terminal would have a limited set of display capabilities and a specialized set of input functions controlled by the keyboard. The display area is divided into a grid of rows and columns, with cells of the grid combined into fields and attributes (such as read-only or foreground and background colors) applied to those fields. Some keystrokes can be entered into any writable field, whereas other keystrokes clear the field of characters, or even send the data. In many applications, the user presses the Tab key to navigate from one field to the next and presses the Enter key to send the data. The application is in a locked state, refusing input, until the response is prepared and transmitted, at which time, the application is unlocked. All transmissions occur over specialized data lines, using a unique protocol that supports the locking and unlocking actions.

However, today most users have replaced their terminals with computers. To connect with terminal-based applications without rewriting them, users use a *terminal emulator* program. This program interacts with the application just as a terminal would and also behaves like a native application on the user’s desktop.

To control and verify the application, Rational Functional Tester must interact with the terminal emulator. In turn, the terminal emulator must expose the attributes of the application to RFT and must also allow Rational Functional Tester to send any specialized input characters to the application. Because the terminal emulator program does not alter the way the application works, RFT’s Extension for Terminal-based Applications installs its own emulator. This enables the interactions between Rational Functional Tester and the terminal emulator to be optimal.

**Prerequisites**

The Extension for Terminal-based Applications is supported only for the Microsoft Windows platform for both recording and playback. It is not possible to record or play back these types of scripts on Linux. To use Rational Functional Tester to test a terminal-based application, you must satisfy the following prerequisites:

**1.** Install and license of Rational Functional Tester version 7.0 or newer.

**2.** Purchase a license for the Extension for Terminal-based Applications for Rational Functional Tester versions older than the 8.0 version.

You should also collect the following connection settings that your terminal emulator uses to connect to the application:

• The Host name (the name of the mainframe) where the application runs. You can also use the IP address, which is typically in the form of four sets of numbers separated by dots, as in 192.168.1.1.

• The Port used to connect to the application. This is a number of typically two to four digits, with a default of 23.

• The Terminal type, such as TN3270, TN3270E (where the E stands for Extended), TN5250, or VT100.

• For a Terminal type of TN3270E, also collect the Logical Unit (LU) or pool name.

• For a Terminal type of TN5250, also collect the Workstation ID.

• The Code page, which controls the types of characters that the terminal accepts and displays. For United States English, this is the default of 037.

• The Screen size, which is the number of rows and columns of characters. This defaults to 24×80, ignoring the mode line.

You might need settings in addition to these, depending on your Terminal type.

**Setup Instructions**

To use the Extension for Terminal-based Applications, you must create a connection. Refer to the data you gathered:

**1.** Launch the terminal emulator by clicking on the Extension for Terminal-based Applications icon.

**2.** In the Connection Configuration dialog box, enter the following information you collected earlier:

**Required:**

• A name for the configuration in the Connection Configurations dropdown box.

• The name or IP address of the mainframe in the Host name box.

• The type of terminal from the Terminal type dropdown.

Your choice of Terminal Type might cause one of the two grayed-out fields to become active. For TN3270E, you might specify the LU or pool name. For TN5250, you might specify the Workstation ID.

**Possibly Required:**

• The Port, if it is different from the default of 23.

• The **Code page**, if it is different from the default of 037 United States.

• The **Screen size**, if it is different from the default of 24×80.

• Click the **Advanced** tab, and investigate the Advanced Settings area to see if any of the settings need to be changed.

**3.** On the toolbar of the Connection Configuration dialog box, click on the **Connect** icon imageto attempt to connect to the mainframe.

**4.** If you see a successful connection in the Terminal Area, click the **OK** button to save this configuration. If you do not see a successful connection, check the settings you entered above to see if you need to enter additional data or change some of the data you entered.

**5.** Click the **Disconnect** icon imageto disconnect from the mainframe.

**6.** Close the Extension for Terminal-based Applications window by clicking the **X** in the top-right corner.

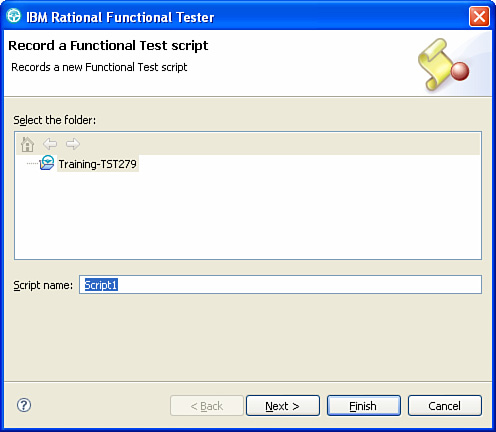
**Recording the Test Script**

You are ready to start recording the test script against your mainframe application. To start recording the Rational Functional Tester script, follow these steps:

**1.** Click the **Record a Functional Test Script** button or **File > New > Functional Test Script Using Recorder**.

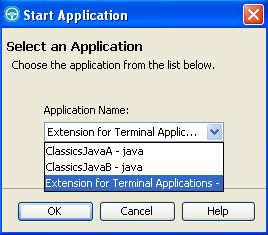
**2.** Enter the script name, select the script folder location, and click **Finish** (see [Figure 11.1](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig01)).

**Figure 11.1** Rational Functional Tester Record window



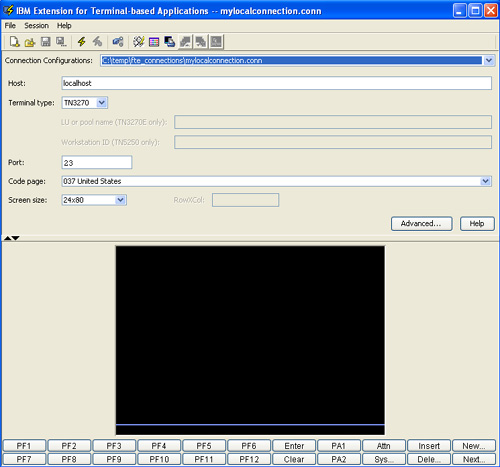
**3.** Click **Start Application** in the recording window to select the **Extension for Terminal-based Applications** option (see [Figure 11.2](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig02)).

**Figure 11.2** Start Application window



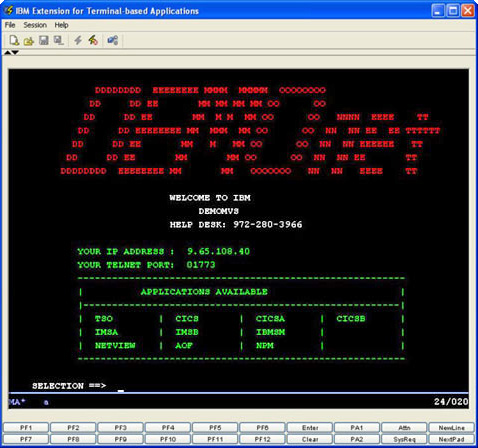
**4.** When the Extension for Terminal-based Applications window opens, select the configuration you defined previously via the Connection, and then click the **OK** button (see [Figure 11.3](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig03)).

**Figure 11.3** Extension for Terminal-based Applications window



**5.** You can now perform the actions to connect to the mainframe, start your application, and interact with your application (see [Figure 11.4](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig04)).

**Figure 11.4** Extension for Terminal-based Applications connected to an application



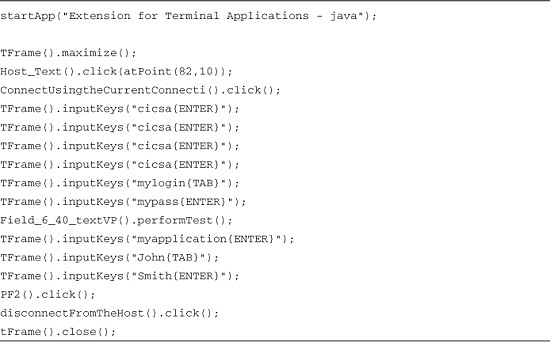
**6.** When you have finished interacting with the application and logged off the mainframe, click the **Disconnect** button.

**7.** Close the Extension for Terminal-based Applications window by clicking the **X** in the top-right corner.

**8.** Click the **Stop Recording** button on the toolbar.

You have now completed this test scenario. When you stop the recording, the test script is generated. [Listing 11.1](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11ex01) shows a sample recorded script, where you can see confirmation of recognition of the terminal objects with their corresponding events. You can also perform property and data verification points on the terminal, using the Rational Functional Tester data pool framework.

**Listing 11.1** Sample recorded script for a terminal-based application



You are ready to play back the test script. The procedure is the same as you would use to play back a Rational Functional Tester script against an application that is not terminal-based.

**SAP**

Rational Functional Tester helps deliver successful functional testing on an application’s build on the SAP framework. SAP is designed to help organizations quickly and confidently deliver expected application functionality and reliability to meet the business needs of the on-demand organization.

Still Using Rational Functional Tester Version 7.x?

If you are using Rational Functional Tester version 7, refer to the “Test Terminal-based Applications with Rational Functional Tester” article on developerWorks (<http://www.ibm.com/developerworks/edu/r-dw-r-termbased.html>).

There are two SAP access implementations supported by Rational Functional Tester: SAP GUI for Windows and HTML (mySAP). The supported versions of SAP GUI are 7.1, 6.40, and 6.20 with patch level 52 or more. SAP GUI has its own controls that Rational Functional Tester needs to understand. mySAP is accessed via standard browsers (Internet Explorer, Firefox), and special mySAP objects are added to the application. These objects have many dynamically changing properties. Therefore, Rational Functional Tester needs to provide a mechanism to handle the value of dynamically changing objects.

Rational Functional Tester enables you to record and play back test scripts against the SAP UI with reliable recognition against SAP controls, including customized data verification of SAP controls. Rational Functional Tester support is built on top of SAP’s GUI scripting framework, exposing all scripting capabilities provided by SAP and adding significant value through the inherent capabilities of Rational Functional Tester.

**Prerequisites**

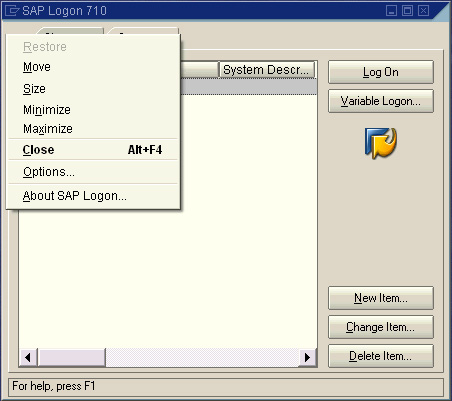
To test an application built on the SAP framework, you must satisfy the following prerequisites:

**1.** Install Rational Functional Tester version 8.0.

**2.** Support the version of the SAP GUI and if needed, support the patch level.

You can check the patch level by clicking on the top-left corner of the SAP Logon dialog box and selecting **About SAP Logon** (see [Figure 11.5](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig05)).

**Figure 11.5** Checking the patch level



**3.** SAP client-side and server-side environments need to be configured for the object recognition to work. Nothing extra needs to be installed; however, a SAP Administrator needs to enable some settings on the server-side. There are two approaches:

• You can enable scripting temporarily from the SAP client. The value set using this procedure is lost when the system is restarted. The following procedure is used to enable scripting temporarily from the SAP client:

**a.** Start the SAP Logon program and log in to the SAP server.

**b.** Start a RZ11 transaction.

**c.** Type sapgui/user\_scripting in the Maintain Profile Parameters window.

**d.** Click Display.

**e.** Click Change value in the Display Profile Parameter Attributes window.

**f.** Type TRUE in the New value field.

**g.** Save the settings and log off the SAP GUI.

**h.** Exit the SAP Logon program.

• If the administrator edits the application server profile of the SAP sever to include sapgui/user\_scripting = TRUE, scripting is enabled by default when the server is restarted. After the server side is enabled, the SAP GUI (client) needs to be restarted. Rational Functional Tester enables client-side scripting automatically when the recorder starts up.

Note

See the “Using Rational Functional Tester to Test SAP applications” technote for additional details ([www-01.ibm.com/support/docview.wss?uid=swg21255820](http://www-01.ibm.com/support/docview.wss?uid=swg21255820)).

**Setup Instructions**

When you test a SAP application, you first need to define the SAP system that SAP GUI client connects to. This example uses SAP GUI version 7.1.

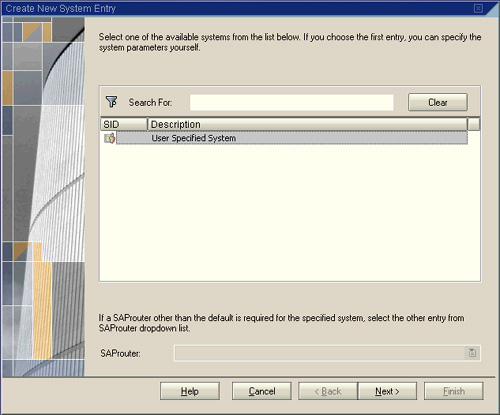
**Defining the SAP System**

To define the SAP, follow these steps:

**1.** Open the SAP Logon view and select the **Systems** tab.

**2.** Click **New Item** and you should get a window like the one shown in [Figure 11.6](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig06).

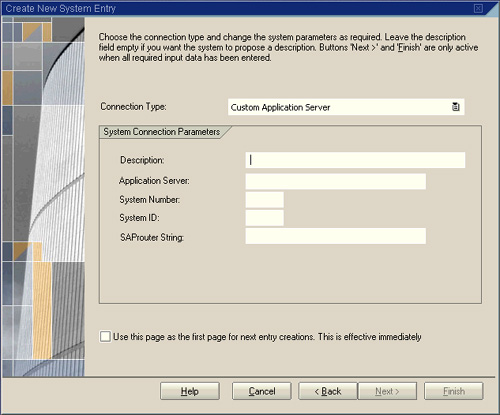
**Figure 11.6** Create New System entry



**3.** Select the system that you are looking for or specify parameters in the Search to find it. If you are not able to find your system, but you have system details, click the **Next** button.

**4.** You need to populate the Description, Application Server, System Number, and System ID fields (see [Figure 11.7](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig07)).

**Figure 11.7** System connection parameters



**5.** Click Finish and test that you can connect to the SAP system.

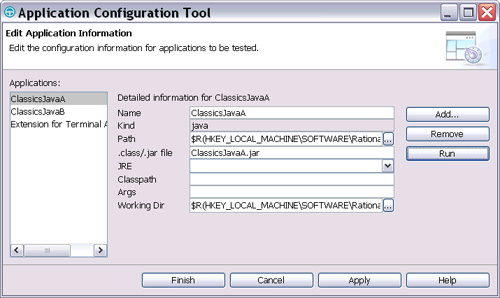
**Configuring the SAP GUI for Testing**

You need to configure the SAP GUI for testing by completing the following steps:

**1.** Launch the Application Configuration Tool wizard by selecting **Configure > Configure Application for Testing** from the **Functional Tester** perspective.

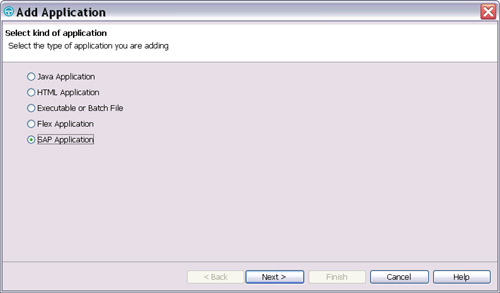
**2.** In the **Application Configuration Tool**, click **Add** button (see [Figure 11.8](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig08)).

**Figure 11.8** Application Configuration Tool



**3.** Select **SAP Application** and click **Next** (see [Figure 11.9](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig09)).

**Figure 11.9** Add Application



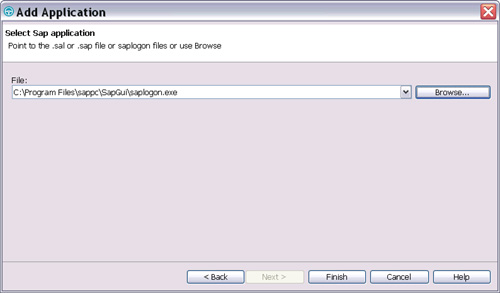
**4.** Specify .sal or .sap file or saplogon files or use the Browse button (see [Figure 11.10](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig10)).

• If you use SAP GUI 7.1, saplogon.exe is available from C:\Program Files\sappc\SapGui.

• If you use SAP GUI 6.4, saplogon.exe is available from C:\program files\sappc\SAPGUI.

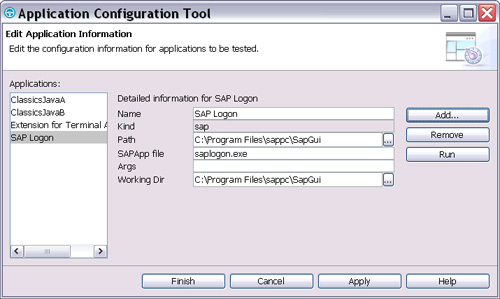
• If you use SAP GUI 6.2, saplogon.exe is available from C:\program files\SAP\FrontEnd\SAPGUI.

**Figure 11.10** Add Application: select SAP Application



**5.** Click **Finish**. Notice that **saplogon** displays in the Applications list in the Application Configuration Tool. You can provide a different name at this point. For this exercise, select SAP Logon. An example of this is shown in [Figure 11.11](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig11).

**Figure 11.11** Application Configuration Tool with the SAP Logon choice



**6.** Click either **Finish** or **Apply** to save your changes.

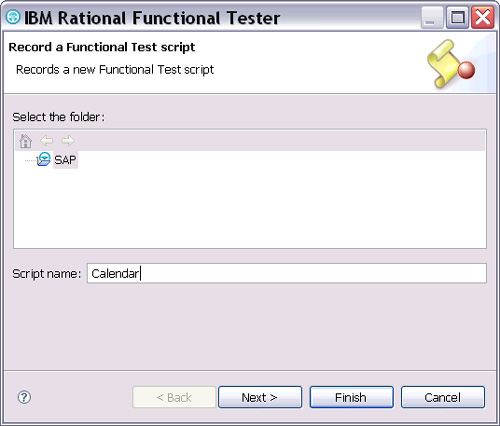
**Recording the Test Script**

You are ready to start recording the test script against your SAP application. To do so, follow these steps:

**1.** Click the **Record a Functional Test Script** button or **File > New > Functional Test Script Using Recorder**.

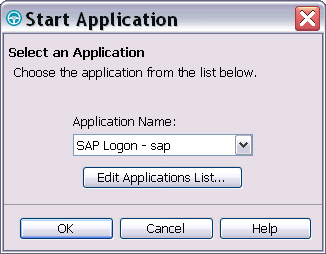
**2.** Enter the script name, select the script folder location, and then click **Finish** (see [Figure 11.12](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig12)).

**Figure 11.12** Rational Functional Tester Record window



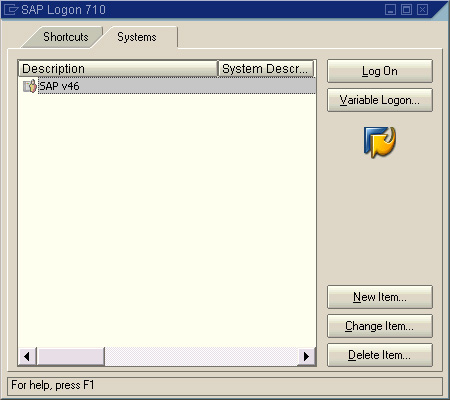
**3.** Click **Start Application** in the recording window to select the **SAP Logon** that was defined previously (see [Figure 11.13](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig13)).

**Figure 11.13** Start Application window



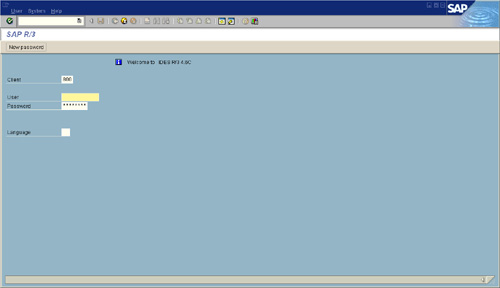
**4.** From the SAP Logon window that opens, select the system that you want to test, and then click the **Log On** button (see [Figure 11.14](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig14)).

**Figure 11.14** SAP Logon window



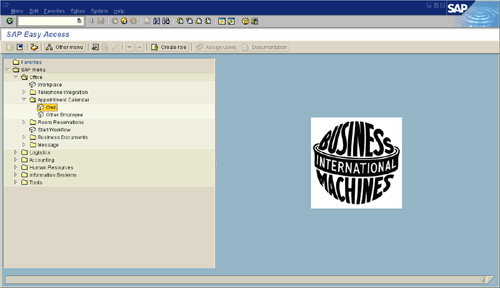
**5.** Enter your user credentials in the SAP Welcome screen (see [Figure 11.15](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig15)).

**Figure 11.15** SAP Welcome screen



**6.** After supplying your user credentials, you can perform actions against the application under test. For this demonstration, select the **Own calendar** under **Appointment Calendar** (see [Figure 11.16](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig16)).

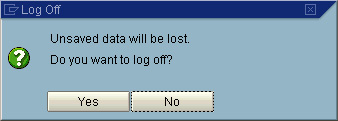
**Figure 11.16** Selections available under Appointment Calendar



**7.** Close the appointment calendar by clicking the round red icon with the X. Then close SAP by clicking the X in the top-right corner.

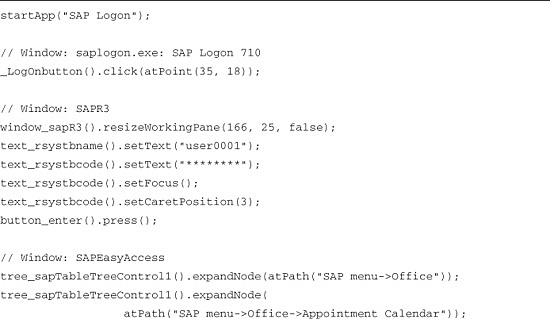
**8.** Select **Yes** in the Log Off window (see [Figure 11.17](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig17)).

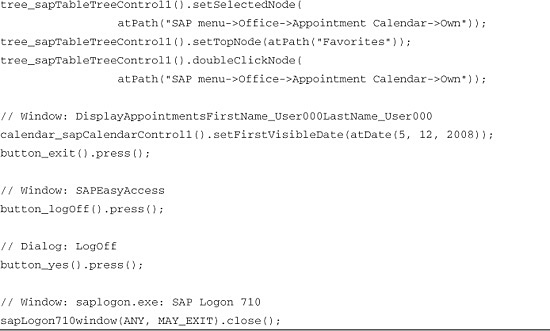
**Figure 11.17** Log Off window



You have completed this test scenario. When you stop the recording, the test script is generated. [Listing 11.2](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11ex02) shows a recorded sample test script, where you can see confirmation of recognition of the SAP GUI objects, with their corresponding events. You can also perform property and data verification points on the SAP GUI, using the Functional Tester data pool framework.

**Listing 11.2** Sample recorded script for a SAP application





As you can see, SAP password is recorded as text\_rsystbcode().setText(\*\*\*\*\*\*\*\*). Rational Functional Tester tool records the password entered as it appears in the password field, and it cannot read the encrypted value. Therefore, before playing back the script, be sure to **replace** the recorded password with the actual password in the script.

You are ready to play back the test script. The procedure is the same as when using the Rational Functional Tester against non-SAP applications.

**Enabling SAP Support for Preexisting Rational Functional Tester Projects**

You can use Rational Functional Tester SAP support for a project that was created using the earlier versions of Rational Functional Tester. You must add two new templates (one for script headers and one for script helper headers) to the project. You will also need to update the build or reference information for the project. You can use the following steps to enable SAP support for pre-existing Rational Functional Tester projects:

**1.** In the Rational Functional Test Projects view, right-click the project and select the **Properties** option.

**2.** In the Properties dialog box, select **Functional Test Script Templates** from the navigation list.

**3.** Select the template type **Script Helper: Header of the file**.

**4.** If you have not customized this template, you can upgrade it by clicking the **Restore Defaults** button.

**5.** Add the line import com.rational.test.ft.object.interfaces.SAP.\*; in the import section of the template.

**6.** After modifying the template, click the **Apply** button.

**7.** Select the template type **Script: Header of the file** and add the same line in the import section of the template.

**8.** Finally, right-click the project again in the Functional Test Projects view and click Reset Java Build Path.

Each member of your team must perform this last step, because the **Java Build Path** is local to each project on each machine.

Note

If you are using Rational Functional Tester version 7, refer to the “Effectively Use the IBM Rational Functional Tester Extension for SAP Solutions” article on developerWorks (<http://www.ibm.com/developerworks/rational/library/07/0126_saracevic/>).

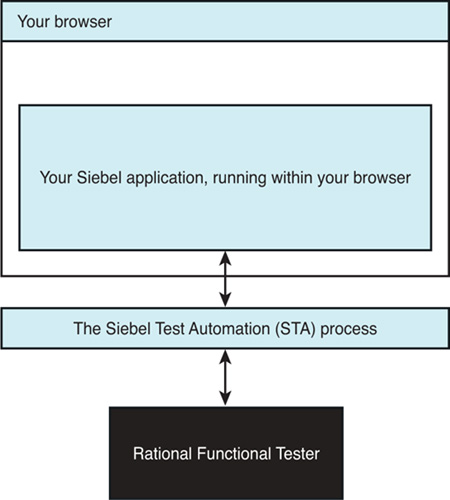
**Siebel**

Rational Functional Tester helps deliver successful functional testing on an applications build on the SAP framework. SAP is designed to help organizations quickly and confidently deliver expected application functionality and reliability to meet the business needs of the on-demand organization.

When you interact with a Siebel application, the Siebel system downloads ActiveX controls into your browser. These are special Siebel controls that do not behave like the standard HTML controls you would use to interact with any non-Siebel website. Because these controls are special, Rational Functional Tester must be assisted to work with them. Otherwise, your scripts will record successfully, but will not play back correctly.

You can obtain a package known as Siebel Test Automation which will permit Rational Functional Tester to interact with the Siebel ActiveX controls, as seen in the diagram contained in [Figure 11.18](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig18).

**Figure 11.18** Siebel Test Automation architecture



The Siebel ActiveX controls come in two types: Standard Interactivity (SI) and High Interactivity (HI) objects. The Rational Functional Tester Extension supports both types. The HI objects can be referenced in one of two ways; see the “[Recording the Test Script](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11sec1lev3)” section. HI support also requires an additional framework to be downloaded to your browser.

**Prerequisites**

To use Rational Functional Tester to test an application built on the Siebel framework, you must satisfy these prerequisites:

• Rational Functional Tester for Siebel requires Rational Functional Tester 7.0 or greater. The capability to record and play back against Siebel applications is included in Rational Functional Tester itself; you do not need to download or install an extension for Rational Functional Tester or modify your Rational Functional Tester installation in any way.

• Rational Functional Tester for Siebel only works for Rational Functional Tester running on Windows. You cannot use Rational Functional Tester on Linux to either record or play back against your Siebel application, because the Siebel Test Automation package only works on Windows.

• Using Rational Functional Tester to record a script for a Siebel application requires that you buy a license, the Siebel Test Automation license from Oracle, and correctly make the Siebel Test Automation package available on the Siebel server. Rational Functional Tester itself is not modified.

• Rational Functional Tester can test applications running under Siebel versions 7.7, 7.8, and 8.0.

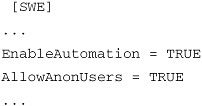
• You must have the Java Runtime Engine (JRE) installed and enabled in your browser. If this is not the case, you will receive an error when trying to record with Rational Functional Tester. However, the error dialog will instruct you in how to download and configure the correct JRE.

**Setup Instructions**

When you are testing a Siebel application, you first need to install and enable the STA on the Siebel server. You must also activate test automation on the Siebel server. Finally, you Install and Enable the STA on the Siebel Server. To do this, refer to your documentation from Oracle.

**Activating Test Automation on the Siebel Server**

To activate test automation on the Siebel server, open the **.CFG** file for the applications that you are testing, and set the **EnableAutomation** and **AllowAnonUsers** switches to TRUE in the **SWE** section:



**Configuring a New Application for Testing Within RFT**

Within RFT, configure your application for testing by following these steps:

**1.** Create a new entry from the Configure menu.

**2.** Select Configure Applications for Testing.

**3.** Add a new browser-based application.

On the end of the URL you normally use to access your application, add “?SWECmd=AutoOn.” This tells the Siebel Web Engine to generate test automation information for Siebel applications. So, if your original URL looks like this:

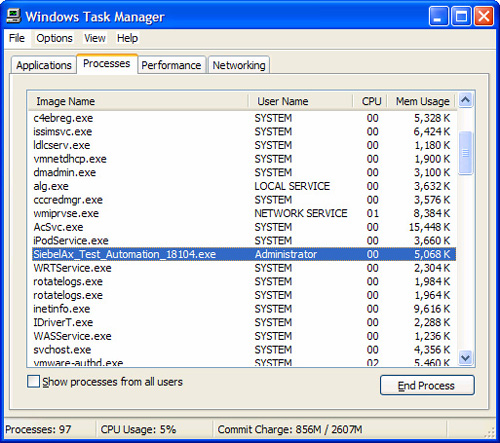
http://hostname/callcenter/start.swe

enter this as the URL in the Configure Applications for Testing dialog:

http://hostname/callcenter/start.swe?SWECmd=AutoOn

If the STA and SWE are installed correctly on your Siebel server, and if you have correctly changed the URL that Rational Functional Tester will use, then the first time you try to create a recording against your Siebel application, the SWE and STA will combine to download and install STA components from the Siebel server to the workstation where Rational Functional Tester is running. This one-time installation may take a few minutes. You can confirm that it has occurred successfully by looking in the Task Manager on the Windows system where Rational Functional Tester is installed; you should see an entry for SiebelAx\_Test\_Automation\_*nnnnn*.exe, where the value of *nnnnn* will depend on your version of the STA. It should look similar to what is displayed in [Figure 11.19](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig19).

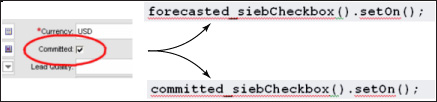
**Figure 11.19** Process table showing STA running on Rational Functional Tester workstation



**Recording the Test Script (Siebel)**

When you are ready to record, you need to make one more decision. By default, the Rational Functional Tester tool records using the repository name, but it can record Siebel HI controls by using the name of the user interface (UI). You might prefer to use the UI name because it makes your script easier to compare to the application. See [Figure 11.20](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig20).

**Figure 11.20** Repository name (“forecasted”) versus UI name (“committed’) for the Committed checkbox



If you want to have your scripts use the UI name, use regedit to create a new registry key named HKEY\_CURRENT\_USER/Software/Rational Software/Rational Test/8 of type REG\_DWORD, and set its Data value to 1.

**Security Warning Window**

When you start the Rational Functional Tester tool with your Siebel application for the first time, the Siebel High Interactivity Framework might be loaded. If a Security Warning dialog box displays, select Yes to install and run the Siebel High Interactivity Framework.

**Siebel Browser Check**

If you do not have the JRE configured to run with your browser, a window titled Siebel Browser Check displays, stating that you do not have the Sun Java Runtime Environment 1.4.1.02 or higher installed. If that happens, you need to install a supported version of the Java Runtime Engine (see the Details section in the Siebel Browser Check window).

To record the Functional Tester script, follow these steps:

**1.** Click the **Record a Functional Test Script** button or choose **File > New > Functional Test Script Using Recorder**.

**2.** Enter the script name, select the script folder location, and click **Finish**.

**3.** Click **Start Application** in the recording window to select the new application that you configured previously.

**4.** Interact with your Siebel application under test exactly as you would if you were not recording in Rational Functional Tester.

**5.** Log out of your Siebel application and close it as you normally would do.

**6.** Stop the Rational Functional Tester recording.

You have completed this test scenario. When you stop the recording, the test script is generated. [Listing 11.3](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11ex03) shows a recorded sample test script, where you can see confirmation of recognition of the Siebel GUI objects with their corresponding events. You can also perform property and data verification points on the Siebel GUI, using the Functional Tester data pool framework.

**Listing 11.3** Portion of a recorded script for a Siebel application



You are ready to play back the test script. The procedure is the same as when using the Rational Functional Tester against nonSiebel applications.

Note

If you are using Rational Functional Tester version 7, refer to the “IBM Rational Functional Tester Extension for Siebel Solutions” article on developerWorks ([www.ibm.com/developerworks/rational/library/07/0109\_saracevic/](http://www.ibm.com/developerworks/rational/library/07/0109_saracevic/)).

**Troubleshooting**

Two common troubleshooting issues are the timeout that is too short for client automation and the need to use Siebel 7.8 scripts for a Siebel 7.7 installation. These issues are addressed next.

**Configuring the Client Automation Server Timeout**

The Client Automation Server (CAS) provides a mechanism to configure a timeout on calls that can result in a stalled system. By default, this timeout is set to one second, but you can configure this in the ivory.properties file, which is found in your Rational Functional Tester installation folder, by setting the rational.test.ft.siebel.cas\_submit\_timeout to the value that you prefer.

**Reverting to Siebel 7.7 with Your Siebel 7.8 Rational Functional Tester Scripts**

Rational Functional Tester V7.0 does not recognize Siebel 7.7 controls if you revert your server installation back to Siebel 7.7 from Siebel 7.8. When you start Siebel 7.8, Microsoft Internet Explorer® installs the latest Microsoft ActiveX® objects for Siebel. Therefore, if you try to record by using the Rational Functional Tester tool on Siebel 7.7 after you have used Siebel 7.8, the software cannot recognize the Siebel 7.7 controls.

The workaround for this situation is to remove the Siebel ActiveX objects (for example, Siebel High Interactivity Framework, Siebel Test Automation) from the C:\WINDOWS\Downloaded Program Files folder, and then try recording with the Siebel 7.7 application again.

**Adobe Flex**

Adobe Flex is an application framework that enables developers to create Rich Internet Applications. Flex applications are usually written using either Adobe MXML™, the Flex markup language, or the Adobe ActionScript® language. A .mxml or .as file can be compiled, and then linked with the Flex SDK libraries to produce an Adobe Shockwave® (.swf) file, which is the executable application. This .swf file can run in the AdobeFlash player, but is typically embedded in an HTML wrapper file.

By default, Flex does not expose its controls so that they can be used by Rational Functional Tester. There are two ways to use Rational Functional Tester with a Flex application:

• **Compile-time automation support—**This method requires that a developer compile the application with the Flex SDK libraries, Flex Automation Framework libraries, and Rational Functional Tester adapter. However, multiple Flex applications that have each been compiled using this method can all be tested simultaneously; this is useful if your website uses several applications, particularly if they also communicate with one another. In fact, all the application SWF files can even be embedded into a single HTML page to avoid pagination during testing. This method works if the application is deployed on a web server machine that is different from the machine where Rational Functional Tester is executed, or if they are both on the same machine. This method has three different techniques that can be used to enable the application. The URL is unchanged. Note that the application that is tested by Rational Functional Tester is identical to the applications that are deployed into production because they are created by different builds.

• **Run-time automation support—**This method does not require any recompile of the application, but does require the one-time compilation of the RuntimeLoader.mxml file with Flex SDK libraries, Flex Automation Framework libraries, and Rational Functional Tester adapter according to the Flex SDK version. After the mxml is compiled to RuntimeLoader swf, it can be used to test any Flex application The user can test one or more Flex applications by passing the name(s) of the application(s) as a list of comma-separated .swfs in the URL. The applications must be deployed to a machine different from the one where Rational Functional Tester is installed. The method consists of a single technique. The URL must include an additional parameter. Because the applications are not rebuilt, the production builds can be tested with this method.

**Prerequisites**

You need to validate a few prerequisites to ensure that Rational Functional Tester can test an application built on the Flex framework:

• A supported version of the Adobe Flex framework is installed

• The Flex Automation Package is deployed on the web server

• You use Flex® Builder™ 2.0.1 or later

• You use Flex SDK 2.0 or later

• You have installed the Flex Automation Framework

**Setup Instructions**

Rational Functional Tester requires an enablement library to test a Flex application. You have two options:

• Recompile the application to include the enablement library

• Use a runtime version of the enablement library

If you recompile the application, you can test it without using a web server. If you do deploy to a web server, you can use the same process as you would if the application had not been recompiled. In addition, you can test multiple enabled Flex applications using a single HTML page.

If you cannot recompile the application (as is the case if you do not have access to the source) or simply prefer not to recompile it, you can use the runtime enablement library. With this method, you must deploy the application to a web server, but you must also change the deployment process to include the enablement library.

The following sections describe each option for enabling your application. There are two to recompile your application, whereas using the runtime enablement library has only a single method.

**Recompiling the Application**

You can recompile your application using one of two methods:

• Using Flex Builder

• Using the Flex command-line compiler

When you have recompiled it, you need to embed the application in an HTML wrapper (if it does not already have one). If you use a web server instead of testing locally, you need to deploy all the recompiled components. The following sections describe these actions.

**Recompiling the Application Using Flex Builder**

If you have Flex Builder, you can configure it to compile your Flex application with the enablement libraries included. To recompile your application, you need to add files to the Flex build environment:

**1.** Copy the automation\_agent.swc file from the *<flex automation install folder>*/frameworks/libs directory to the *<flex builder install folder>*/Flex SDK 2/frameworks/libs directory.

**2.** Copy the automation\_agent\_rb.swc file from the *<flex automation install folder>*/frameworks/locale/en\_US directory to the *<flex builder install folder>*/Flex SDK 2/frameworks/locale/en\_US directory.

This path is for the en\_US locale (U.S. English). If you use a different locale, replace en\_US with that locale. By default, the Flex automation install directory is C:\Program Files\Adobe\Flex Automation, and the Flex builder install directory is C:\Program Files\Adobe\Flex Builder 2.

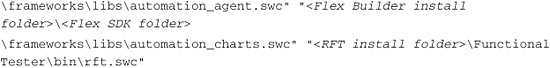
**3.** Start Flex Builder.

**4.** Create or select your Flex project.

**5.** Click **Select Project > Properties > Flex Compiler**.

**6.** Enter the following code in the Additional compiler arguments field:

image



Note

You need the automation\_charts.swc file only if the application contains charting controls.

**7.** Click **OK** to save your changes.

**8.** Click **OK** to close the Properties dialog box.

**9.** Compile your Flex sample application.

**10.** Open the application’s .mxml file in Flex Builder. (Use the example filename compiletime.mxml.)

The output files from the compilation are compiletime.swf and compiletime.htm. The .htm file is an HTML wrapper that is generated by Flex Builder in the applications directory. This HTML is ready to be tested by Rational Functional Tester.

**Recompiling the Application Using the Command Line**

If you do not have Flex Builder, you can compile the source with the Flex SDK libraries, Flex Automation libraries, and Rational Functional Tester adapter via the command line. You do this by running the batch file buildapplicationwithadaptor.bat that can be found in the RFT install folder>\FunctionalTester\Flex folder:

**1.** Copy this batch file to the same directory as your .mxml file.

**2.** In a command shell, change to that directory and run the batch file with the mxml filename as the argument. For example, if your application is compiletime.mxml, type the following command at the command prompt:

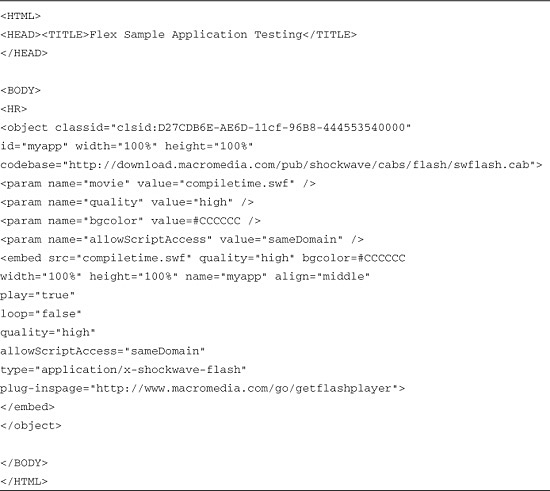
Buildapplicationwithadapter.bat compiletime.mxml

This compilation results in a .swf file. (For the example before it is compiletime.swf.)

**Embedding the Recompiled Flex Application in an HTML Wrapper**

Before you can use Rational Functional Tester to test the Flex application, the application must be embedded in an HTML page. If the page does not already exist or you do not want to modify an existing page, you can create one. For the previous compiletime.mxml application, [Listing 11.4](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11ex04) shows sample HTML code that you can save in a .html file. You can name the file compiletime.html, for example.

**Listing 11.4** HTML wrapper for Flex application



The SWF name has been passed as parameter. You can use this same HTML page to test other enabled applications simply by changing the .swf filename.

**Deploying the Recompiled Application**

You can test your application on the same machine where you compiled it. If you prefer to use a web server, you need to deploy all of the following files:

• Application file (SWF)

• HTML wrapper and wrapper files (the wrapper files can include SWF files, JavaScript files, and other files)

• Module files (SWF)

• RSL files (SWC)

• Helper files such as theme SWC files, style SWF files, and image, video, and sound files

**Using Runtime Automation Support**

If you cannot or do not want to rebuild your application’s .swf file, you can use the runtime enablement library. The runtime enablement library is available in the <RFT install folder>\FunctionalTester\Flex folder, and is named runtimeLoader.swf.

If you use a version of the Flex SDK other than the 2.0 version, you might need to recompile the runtime enablement library. The source for the runtime enablement library is named runtimeLoader.mxml and is also available in the <RFT install folder>\FunctionalTester\Flex folder. The runtimeLoader.mxml file needs to be compiled with Flex SDK libraries, Flex Automation Framework libraries, and Rational Functional Tester adapter to create the runtimeLoader.swf file, using either of the methods mentioned previously.

To use runtime automation support, you need to deploy the runtime enablement library along with the application. For example, if you use Apache for your web server, you copy the runtimeLoader.swf file to your apache\htdocs directory.

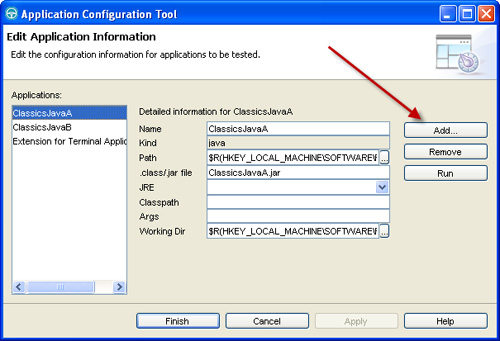
**Configuring the Application for Testing by RFT**

To configure your application for testing, follow these steps:

**1.** Select **Configure > Configure Application for Testing** from the Functional Tester perspective.

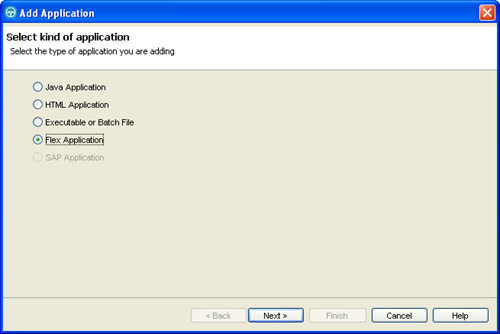
**2.** In the Application Configuration Tool, click the **Add** button. See [Figure 11.21](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig21).

**Figure 11.21** Application Configuration Tool



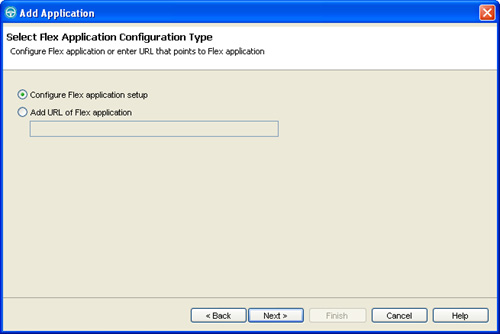
**3.** Select **Flex Application** as in [Figure 11.22](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig22), and click **Next**.

**Figure 11.22** Add application



**4.** For an application accessed via a URL, you would click the radio button for Add URL of Flex Application, enter the URL, and click **Finish**. For a locally installed application, click the radio button Configure Flex application setup, and then click **Next**. See [Figure 11.23](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig23).

**Figure 11.23** Add Application—Select the type of Flex application



**5.** In the Flex Application Configuration dialog box, enter the parameters for your application. The application name is passed to the HTML page as a query parameter:

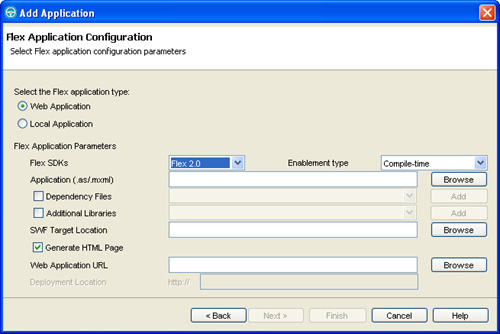
http://localhost/runtime.html?automationswfurl=runtime.swf

Multiple applications can be passed as follows:

http://localhost/runtimeLoadingTestMultipleSwfs.html?automationswfurl=runtime.swf, runtime2.swf,runtime3.swf

See [Figure 11.24](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig24).

**Figure 11.24** Application Configuration Tool with the Flex parameters



**6.** Click **Finish** to save your changes.

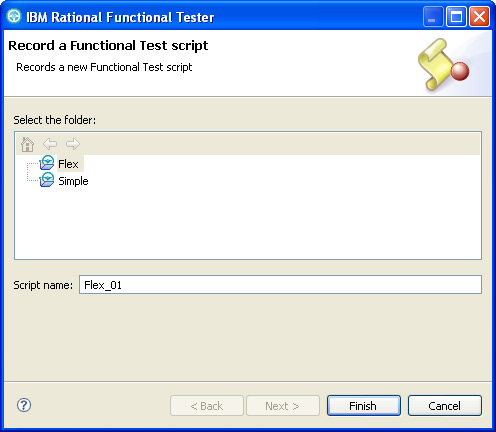
**Recording the Test Script (Flex)**

You are ready to start recording the test script against your Flex application. Follow these steps:

**1.** Click the **Record a Functional Test Script** button or **File > New > Functional Test Script Using Recorder**.

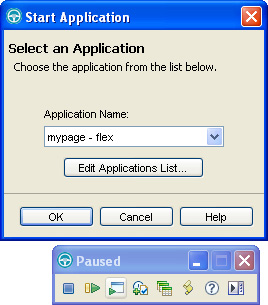
**2.** Enter the script name, select the script folder location, and then click **Finish**. See [Figure 11.25](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig25).

**Figure 11.25** Rational Functional Tester Record window



**3.** Click **Start Application** in the recording window to select the **Flex Application** that was defined previously. See [Figure 11.26](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig26).

**Figure 11.26** Start Application window



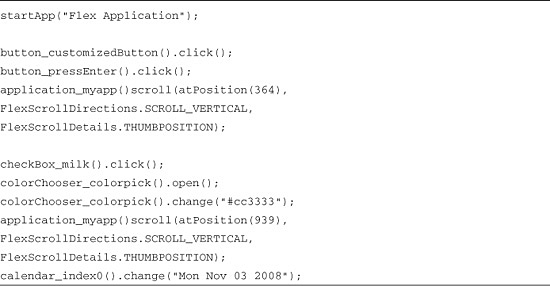
**4.** Perform actions against the application under test.

**5.** Exit the application as you normally would.

**6.** Close the browser window.

You have completed this test scenario. When you stop the recording, the test script is generated. [Listing 11.5](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11ex05) shows a recorded sample test script, where you can see confirmation of recognition of the SAP GUI objects with their corresponding events. You can also perform property and data verification points on the SAP GUI using the Functional Tester data pool framework.

**Listing 11.5** A sample recorded script for a Flex application



You are ready to play back the test script. The procedure is the same as when using the Rational Functional Tester against a nonFlex application.

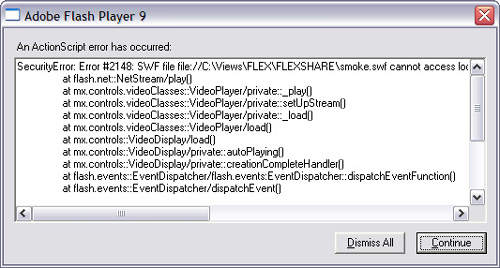
**Troubleshooting**

If any of the following errors occur, you can resolve them easily yourself.

**Security Warning**

A security violation notice comes up when the Flex application is started (see [Figure 11.27](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig27)).

**Figure 11.27** Adobe Flash error notification



Make sure that the Flex applications that you test are run from a trusted folder.

**1.** You can add the OUT directory to the Flex file in the FlashPlayerTrust folder.

**2.** You can create a FlashPlayerTrust folder in the C:\Windows\System32\Macromed\Flash\ directory:

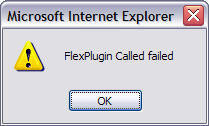
**a.** Create a .txt file and add the path of the application to be tested to this file.

**b.** Save this file as Flex with no extension. You can add any number of directories to this TXT file. For example, if the testMyApp.swf and testMyApp.html files are in the C:\FlexTutorial directory, you can add that path to the Flex file.

**Flex Plug-in Call Failed Alert**

When the application is run from the localhost/IIS server, you might get a notice saying that the plug-in call failed (see [Figure 11.28](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html#ch11fig28)).

**Figure 11.28** Adobe Flex error notification



If this happens, check the local security settings of the Internet Explorer ActiveX and plug-ins:

**1.** Open Internet Explorer.

**2.** Select **Tools > Internet Options** from the menu bar.

**3.** Click the **Security** tab.

**4.** Select the appropriate web content zone. For example, select **Local Intranet** if you have set up Apache or IIS on the local machine.

**5.** Select **Custom Level**.

**6.** From the Reset to list, select **Medium-low**.

**7.** Under Initialize and script ActiveX controls not marked as safe, click the **Enable** radio button.

**8.** Click **OK**.

Note

If you use Rational Functional Tester version 7, refer to the Testing Flex applications with the Rational Functional Tester 7.0.1 article on developerWorks ([www.ibm.com/developerworks/rational/library/08/0819\_awasthy/](http://www.ibm.com/developerworks/rational/library/08/0819_awasthy/)).

**Summary**

You now know how to use extensions to Rational Functional Tester to test terminal-based, SAP, Siebel, and Adobe Flex applications. In some instances, you needed to work with a separate user interface (for example, terminal-based applications). In other instances, it is simple a matter of configuring Rational Functional Tester a certain way (for example, SAP, Siebel, and Adobe Flex). In any case, you now have the knowledge to test these technologies, using Rational Functional Tester.

[Prev](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch10.html)

[Chapter 10. Advanced Scripting with Rational Functional Tester TestObjects](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch10.html)

[Next](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch12.html)

[Chapter 12. Extending Rational Functional Tester with External Libraries](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch12.html)

**Welcome to Safari**. Remember, your free trial will end on March 7, 2015, but you can [subscribe at any time](https://www.safaribooksonline.com/subscribe/)

* [Recommended](https://www.safaribooksonline.com/r/)
* [Queue](https://www.safaribooksonline.com/s/)
* [Recent](https://www.safaribooksonline.com/recent/)
* [Topics](https://www.safaribooksonline.com/t/)
* [Settings](https://www.safaribooksonline.com/u/)
* [Blog](http://blog.safaribooksonline.com/)
* [Support](http://support.safaribooksonline.com/?prod=flow)
* [Support](http://msupport.safaribooksonline.com/?prod=flow)
* [Feedback](mailto:feedback@safaribooksonline.com)
* [Sign Out](https://www.safaribooksonline.com/accounts/logout/)

© 2015 [Safari](http://www.safaribooksonline.com/).   
[Terms of Service](https://www.safaribooksonline.com/terms/) / [Membership Agreement](https://www.safaribooksonline.com/membership-agreement/) / [Privacy Policy](https://www.safaribooksonline.com/privacy/)

[**Make font larger Make font smaller**](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch11.html)