[Skip to content](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.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/app01.html)
* [Expand Nav](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.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/app01.html)

* [Search in book...](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html)

Top of Form

Bottom of Form

* [Toggle Font Controls](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html)
* [Share this](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.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/ch16.html)

[Chapter 16. Internationalized Testing with Rational Functional Tester](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch16.html)

[Next](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app02.html)

[Appendix B. Regular Expressions in Rational Functional Tester](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app02.html)

**Appendix A. Advanced Logging Techniques**

**Daniel Gouveia**

*There are different ways to take advantage of Rational Functional Tester’s logging capabilities.* [*Chapter 9*](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch09.html#ch09)*, “*[*Advanced Rational Functional Tester Object Map Topics*](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch09.html#ch09)*,” covers the usage of open-source tools, including Log4J (verbose reporting). This appendix looks at ways you can expand on Rational Functional Tester’s logging.*

*You learn tips and techniques for emailing results to yourself, using EXtensible Stylesheet Language (XSL) to transform Rational Functional Tester’s XML log, and creating a new log type. XSL is not covered in detail. Neither is Java. It is assumed that you have some knowledge of these technologies and have worked with them. The goal is to provide you with the information to get your started so you can get the information you want—how you want it.*

*If you read this appendix topic-by-topic, you can see that you can build off of three advanced logging mechanisms. Of course, you can simply go to the topic that suits your need. It is written so that you can find the advanced logging technique that addresses your problem.*

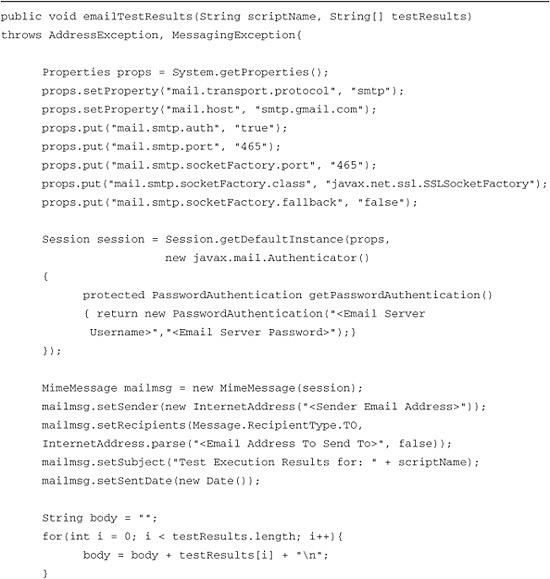
**Sending Test Results Via Email**

You can take advantage of the scripting languages Java and VB.NET to send an email, announcing test results. This is an excellent precursor to reviewing your actual execution log. You can make the email as robust or as simple as you wish. This section provides the information for accomplishing this capability using Simple Mail Transfer Protocol (SMTP).

The basic idea is the same for both languages. You need to set up the email structure (who it is being sent to, who it is coming from, its subject, body, and so on). You then implement the SMTP client, specifying the host, port, and any necessary Secure Sockets Layer (SSL) connection information. Finally, you simply send the email.

[Listing A.1](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex01) provides you with a Java implementation that sends simple pass/fail emails. The aim is to let the recipient know that execution has completed and what the results are. They are created as separate methods (functions). This enables you to place them into a Helper Superclass (Helper Base Class).

**Listing A.1** Email method for Java



Image

The preceding code example uses JavaMail version 1.4.1. It also uses JavaBeansActivation Framework version 1.0.2. You are able to obtain both of these off of Sun’s website. You download these packages as zip files and then unzip them to any directory. You just need to make sure that you have all of the .jar files listed in your Rational Functional Tester project’s Java build path. The previous code sample requires the following .jar files:

• mail.jar

• mailapi.jar

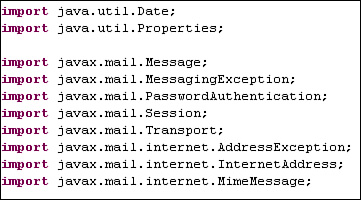
• pop3.jar

• smtp.jar

• activation.jar

You also need to make the appropriate imports into your class file, as shown in [Figure A.1](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig01).

**Figure A.1** Import statements

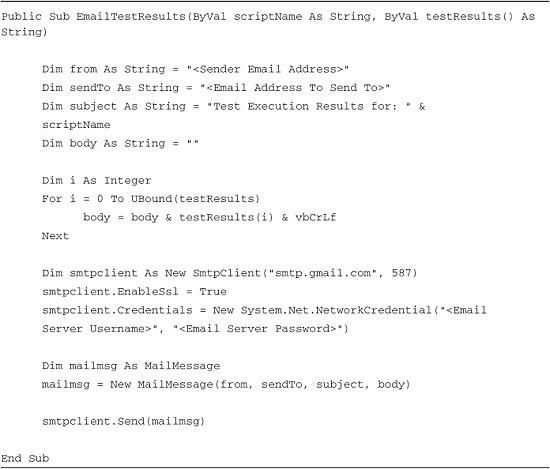


This example uses Gmail as the SMTP server. It first sets the properties that specify the mail protocol, mail (Gmail server) host, server port, and SSL settings. It then addresses the authentication mechanism that is needed. It creates a new mail session and creates the password authentication class, supplying the user credentials needed for the server. The method (emailTestResults) then creates the email message that sets the sender, send to, send date, subject, and body values. This method accepts two arguments: the name of the invoking script and an array of test results. The script name is used in the subject, letting you know which script the results are for. The method parses the results array to list the result of each subscript’s execution results (in the case of a regression script). Lastly, it sends the email.

[Listing A.2](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex02) is the VB.NET version of the prior email solution. It performs the same task as the prior Java example. One key difference is that you won’t need to download and use separate packages. You simply import the Mail namespace into your class file and use the following line:

Imports System.Net.Mail

**Listing A.2** Email function for VB.NET

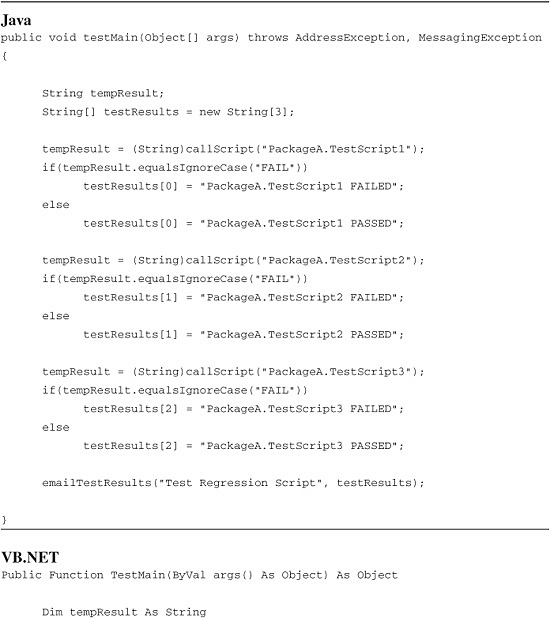


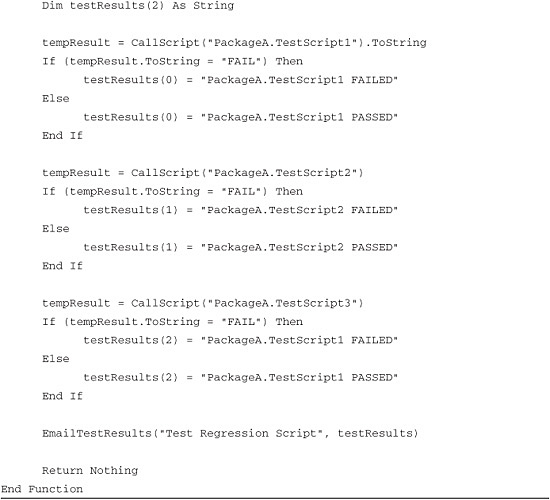
This provides access to the SmtpClient and MailMessage classes. The first thing this sample code does is set up the sent from, send to, subject, and body email fields. Like the Java code, it accepts two arguments: a script name string and test results array. The script name string is used in the subject message. The test results array is used to list the separate execution results for each script (for example, if a regression script is executed). It then creates an SMTP client, specifying a server (Gmail) and port. It sets up the SSL, providing the appropriate user credentials. The second to last thing it does is instantiate the MailMessage class, passing in the required email fields. Finally, it sends the email.

**Calling the Mail Method**

To use the method or function, you need to modify your scripts to catch the pass/fail results (in an array). You invoke the method/function and pass in the script name and test results array. The examples in [Listing A.3](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex03) show how this might be done in a regression script. The scripts, invoked by the regression script, return the overall pass/fail result for them. The regression script catches the results in an array, sending them to the email method along with the name of the regression script.

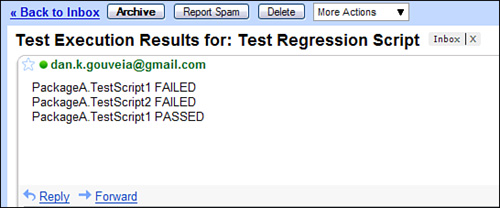
**Listing A.3** Invoking the Email method





The result of running one of the prior regression scripts is the creation and distribution of a simple email that shows the basic pass/fail log. It lets you know if each script, invoked in the regression test, succeeded or not. An example of this is shown in [Figure A.2](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig02).

**Figure A.2** Test results displayed in email



**Transforming the XML Log by Using XSL**

Rational Functional Tester offers you different types of logging. You can write to a text-based log, HTML-based log, and logs housed in different test-management tools. It also offers you the ability to log your script’s playback events to an XML-based log. This provides you with the flexibility to develop your own custom log should you desire something different than the out-of-the-box offering. You can parse it however you want and display the information that is pertinent to you.

You can use XSL to transform the XML log into an HTML-based file. If you are interested in using a different means to parse and manipulate the XML log file, review [Chapter 4](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch04.html#ch04), “[XML and Rational Functional Tester](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch04.html#ch04).” This section is not meant to be a tutorial on XSL. There are many good tutorials on the Web for that. Instead, this section shows you, through examples, how to use XSL to transform Rational Functional Tester’s XML log into HTML.

The first thing that you want to do is acquire a utility to actually carry out the transform. Microsoft offers a utility called msxsl.exe, free of charge. You can also use Apache’s Xalan-Java. Both of these work with the example in this section. Also, both are command-line utilities.

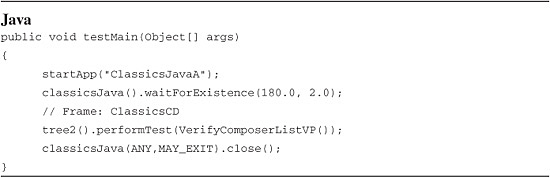
The next thing that you need is an XML file to look at. You can easily accomplish this by selecting the XML log type in your Rational Functional Tester Playback Preferences (see [Figure A.3](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig03)).

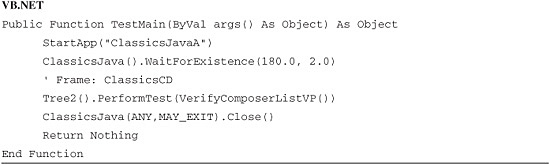
**Figure A.3** Setting the Rational Functional Tester XML log



You can play back any script, preferably with a verification point (VP) in it, after you have set XML as your logging type. [Listing A.4](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex04) (in both the Java and VB.NET version) generates the XML-based log shown in [Figure A.4](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig04).

**Listing A.4** Simple Rational Functional Tester script to generate the XML log





**Figure A.4** Rational Functional Tester XML log (same for Java or VB.NET execution)



You can use this test log as a template for building out your XSL file. Even though it is short and simple, it provides you with the hierarchal structure of Rational Functional Tester’s XML logs. The basic XML log elements that you work with are:

• <TestLog>—The root element in the log

• <Event>—The datapool and script events that Rational Functional Tester logs

Putting these together enables you to construct your XSL looping and select elements. For instance, if you wanted to loop through each event, you would use:

<xsl:for-each select="TestLog/Event/Event">

This enables the transform to look at each logged event. If you just provided TestLog/Event, you would access only the datapool iteration events, not the data for things such as pass/fail results, script command issues, warnings, and so on.

To select a particular value from the XML log, use <xsl:value-of> element. If you want to simply obtain the data from an <Event> element, use something like the following:

<xsl:value-of select="@Type"/>

This returns the type of event, such as “Script Start,” “Application Start,” “Verification Point,” and so on. If you wanted to acquire data from the child of an event (such as a <Property> element), you would place the name of the XML element in front of the @ sign.

<xsl:value-of select="Property/@name"/>

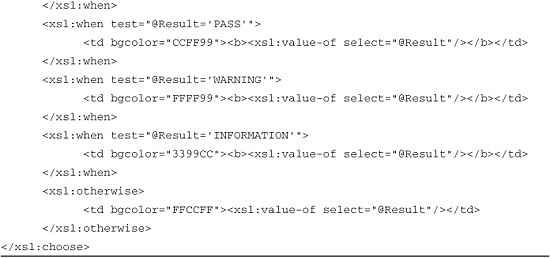
This XSL line gets the name property for an event. This is something like the name of the application you started, a verification point name, and so on. If you want to get another property ID, simply replace the value after the @ sign.

If you need to add some conditional logic, you can use the <xsl:choose>, <xsl:when>, and <xsl:otherwise> combination. This enables you to traverse different paths in your XSL transform. You might use this when you want to work with the different results found in your log (for example, Pass, Fail, Warning, and Information).

[Listing A.5](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex05) employs conditional logic to handle the different test results. It first uses the <xsl:choose> element and then the <xsl:when> and <xsl:otherwise> to make the decisions. Ultimately, the code decides what color to make the table cell’s background color (red, green, yellow, or blue), based off of the verification result. Further, if it encounters a verification point failure, it creates a link to call the specific verification comparator.

**Listing A.5** XSL conditional logic





After you create the XSL transform, you need to process it. That’s what one of the two utilities mentioned earlier assists with. If you choose to use msxsl.exe, you need to specify the XML, XSL, and resulting HTML file:

msxsl.exe <xml file> <xsl file> -o <html file>

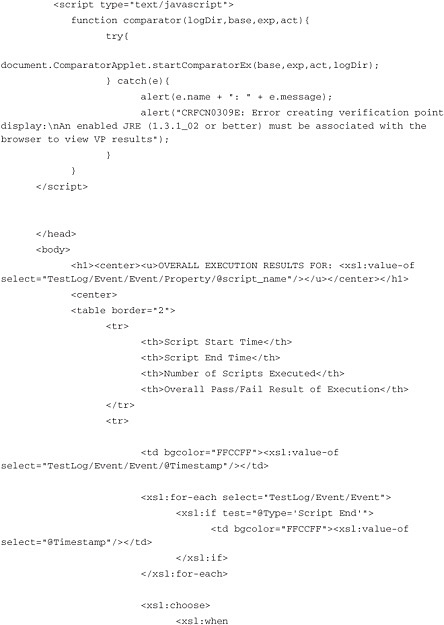
Choosing Apache’s Xalan-Java requires you to place its .jar files in the classpath. You can do this either with an environment variable or in the command-line string. Like msxsl.exe, you need to specify the XML, XSL, and HTML files:

java -classpath serializer.jar;xalan.jar;xercesImpl.jar;xml-apis.jar  
org.apache.xalan.xslt.Process -IN <xml file> -XSL <xsl file> -OUT <html  
file>

Both utilities do the same thing. They take the specified XML file and apply the XSL transform to it. The result is an HTML representation of the original XML log. To provide more context, see [Listing A.6](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex06).

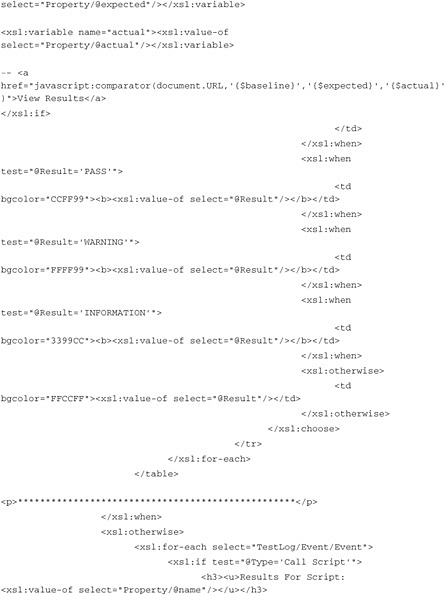
**Listing A.6** Complete XSL file for transforming the Rational Functional Tester XML log















Using Xalan-Java or msxsl.exe, you can pass this XSL transform (as a file), along with the XML log, displayed in [Figure A.4](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig04), to create a clean-looking HTML file. The command-line strings would look like the following:

For the msxsl.exe command-line string:

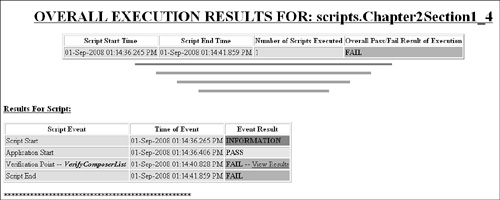
msxsl.exe rational\_ft\_log.xml XSLTransform.xsl -o HTMLLog.html

For the Xalan-Java command-line string:

Image

[Figure A.5](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig05) shows what the resulting HTML log file looks like using the XSL transform.

**Figure A.5** HTML log using XSL transform

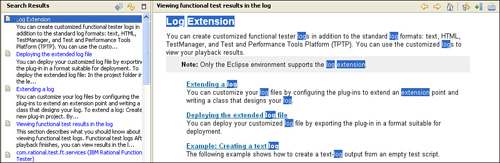


You now have a basic understanding of how to apply XSL transforms to your Rational Functional Tester XML logs. You are not limited to running command-line tools to perform the transformation. You can actually build your own utility that does this for you, transforming multiple logs at once.

**Creating Your Own RSS Log Type**

Rational Functional Tester provides you with the ability to create your own log type. This capability can be implemented only with the Eclipse version of Rational Functional Tester because you need to access the Plug-in Development perspective. You can, however, use the new log type with the .NET version. Rational Functional Tester’s Help files provide the necessary documentation to get you started. If you search on the phrase “log extension,” the section on creating your own log type should be at the top of the list (see [Figure A.6](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig06)).

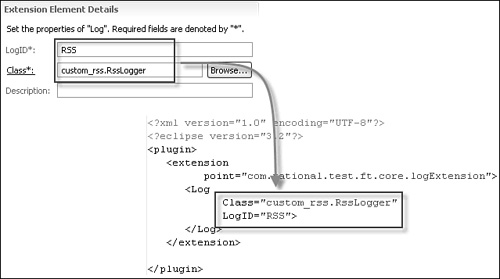
**Figure A.6** Log extension topic in Help files



The Extending a log topic discusses how to build the Eclipse plug-in project necessary for creating your own log type. Following these steps detailed in this topic, you are taken through the New Plug-In Project wizard. On the first page of the New Project wizard, you can simply provide the name for your project and click the **Next** button. You should deselect the **This plug-in will make contributions to the UI** checkbox on the second page. You can leave the default values for the Plug-in Properties. Clicking on the **Finish** button completes the wizard, creating the plug-in project for you.

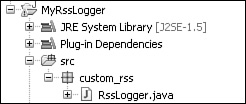
After the project is created, the help instructions continue to discuss the specification of a dependency and definition of an extension. When you get to the point of extending an extension point (steps 6 and 7 in the Help topic), you need to provide a LogID in the Extension Element Details section. This is displayed as a selection in Rational Functional Tester’s log type combo box. You also need to provide a class name, specifying any package that you place the class in. This maps to the plugin.xml file (found on the last tab of the MANIFEST.MF GUI). See [Figure A.7](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig07).

**Figure A.7** Extension element details



You now have a new plug-in project, ready for coding. The Example: Creating a text log section in the Help files provides sample code to show you what’s needed to create a new log type—in this case, a new text log. To begin, you need to enter in the .src folder of your project and expand the package that it contains. You then need to create a new class, supplying the same name that you placed in the Extension Element Details section of your MANIFEST.MF file (for example, [Figure A.7](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig07) shows RSSLogger). An example of this is shown in [Figure A.8](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig08).

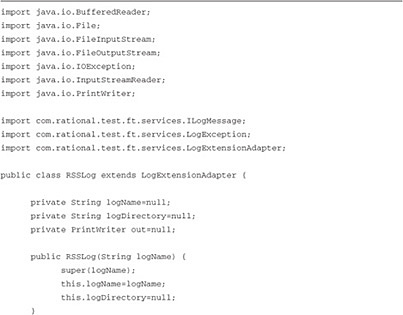
**Figure A.8** Package and class structure for new log type



You need your new class to extend the LogExtensionAdapter class. Lastly, your class needs to provide the proper code to implement the necessary constructors and methods for initializing, writing to, and closing the log. Using the code supplied in the Example: Creating a text log help section provides you with an excellent example to prepare you for developing your own custom text log type.

To implement a log type that builds out an RSS feed, replace the code you used following the “text log” example with the code shown in [Listing A.7](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex07).

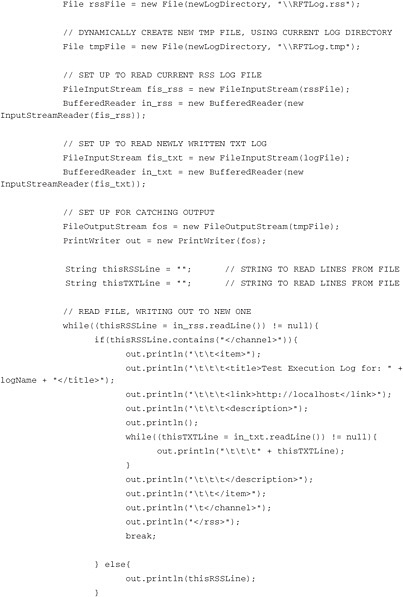
**Listing A.7** Java code for creating an RSS log

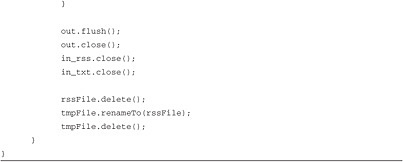






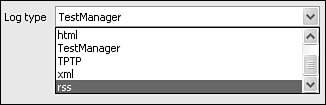






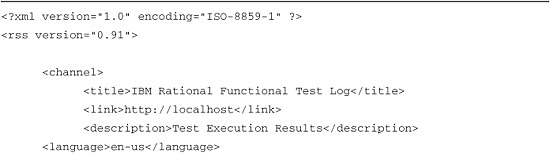
After you create the necessary pieces for your plug-in, you need to export it to the plug-ins folder. This is covered in the last section of the Extending a log Help topic, Deploying the extended log file. Keep in mind that depending on the version of Rational Functional Tester you use, your plug-ins folder path might vary. If you use version 7.x of Rational Functional Tester, your path is C:\Program Files\ibm\SDP70. The path for version 8.x is C:\Program Files\IBM\SDP. You might need to restart Rational Functional Tester for the new log type to show up as an option in your logging preferences. When completed, you see something similar to [Figure A.9](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig09).

**Figure A.9** Rational Functional Tester logging preferences for RSS



You need to do a few things before you try out your new RSS log type. You need to set up an RSS file template. You can find this on the Web, or just use the one shown in [Listing A.8](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01ex08).

**Listing A.8** RSS file template



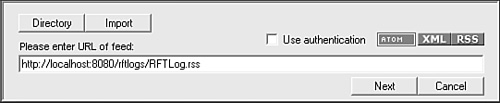


This template uses RSS version 0.91. You can also use version 1.0 or 2.0. The template is basically structured so that it provides some default Rational Functional Tester information (for example, log and description). It then is ready to accept the <item> and </item> tags provided by your custom log code. These tags are used to parse out new execution logs, displaying as new items to your feed reader. One thing to note is that the writeRSSFile() method in the XSL code looks in a rsslogs directory for the .rss file. This was manually created in the Rational Functional Tester project’s log directory. The Java code is dynamic and based off of the executing script’s information. You might want to change this, depending upon your web server’s access.

You need to set up a web server to deliver the feed. It doesn’t matter which web server you use. You need to set it up so it can access the RSS directory you have set up. This enables it to serve up the actual RSS file.

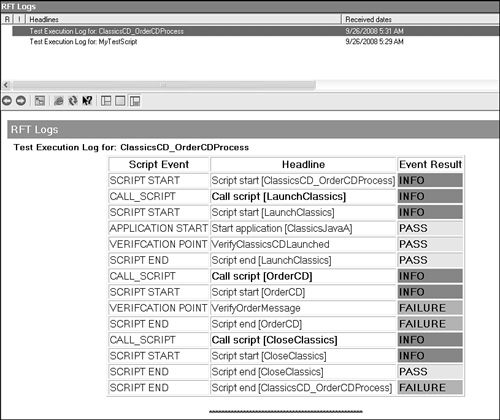
You also have to set up your favorite feed reader to access the feeds from your RSS file. For the most part, you simply specify the URL, including the .rss file. For instance, http://localhost:8080/rftlogs/RFTLog.rss (see [Figure A.10](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig10)). Some feed readers enable you to set up how often it looks for new feeds. This can be useful for acquiring script execution information soon after playback has completed.

**Figure A.10** Setting up the Feed Reader to grab the Rational Functional Tester log feed



After execution completes, you can review your log in the feed reader. Ideally, you want to have your reader set up so it notifies you when a new log has been added (for example, a new feed item has been inserted into the Rational Functional Tester RSS file). This informs you, not only of your completed scripts, but of your coworker’s completed scripts you were unaware of (see [Figure A.11](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app01.html#app01fig11)).

**Figure A.11** Rational Functional Tester RSS log



Moving ahead, you might want to change the log format. Perhaps you want to add a link to the baseline and actual verification point files (shown in the XSL code). In any case, the Java code and RSS formats provided in this section give you an idea of how to create your custom log types in Rational Functional Tester.

**Summary**

You now have the knowledge necessary for extending Functional Tester’s out-of-the-box logging and notification capabilities. These topics provide the basis for creating your own logs using either XML or an Eclipse plug-in and having Rational Functional Tester notify people when it is done executing. Not only can you acquire the information you displayed in the format that you want, but you can also get the information when it is ready. You no longer need to wait for an email from a team member. This is especially useful when your team members are dispersed across different time zones.

[Prev](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch16.html)

[Chapter 16. Internationalized Testing with Rational Functional Tester](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/ch16.html)

[Next](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app02.html)

[Appendix B. Regular Expressions in Rational Functional Tester](https://www.safaribooksonline.com/library/view/software-test-engineering/9780137036455/app02.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/app01.html)