

Analyze system performance

Validate system scalability with IBM Rational Performance Tester

Skill Level: Introductory

[Brian Bryson](#)
IBM

26 May 2005

Learn how to analyze system performance with IBM Rational Performance Tester v6.1. This tutorial covers the process of test creation, execution, and results analysis. The goal is to guide you through the process of performance testing with IBM Rational Performance Tester V6.1. No prior experience with Rational Performance Tester is required.

Section 1. Introduction

Overview

This tutorial is designed to introduce you to performance testing with the recently unveiled IBM® Rational® Performance Tester V6.1. IBM Rational Performance Tester V6.1 is a brand new performance testing tool used to validate the scalability of Web-based applications.

This tutorial teaches you how to create, schedule, and execute your first performance test using Performance Tester. Along the way, you'll explore the challenges of performance testing and the solutions offered by Performance Tester to address these challenges.

Learn how to do the following:

- Record a performance test
- Create a multi-user performance schedule
- Analyze performance test results

...And all of this without typing a single line of code. Ready?

Who should take this tutorial?

This tutorial was designed for quality assurance professionals faced with the challenge of validating the scalability of Web-based applications. While the challenges of performance testing are well established, IBM Rational Performance Tester V6.1 is a brand new tool. Accordingly, this tutorial is written for any and all who want to explore the use of Performance Tester to address their performance test needs.

- Tool experience required: None
- Knowledge of performance testing required: Minimal
- Technical level of tutorial: Introductory

At present, Performance Tester supports only the HTTP protocol. Thus, from a technical standpoint, the only requirement for the tool is that you are testing a Web browser-based application. If you need to test other types of applications -- such as Citrix-based applications, or ERP applications such as Oracle or Siebel -- you can not use Performance Tester.

Software requirements

There are two software requirements for this tutorial: IBM Rational Performance Tester V6.1 and the Java™ Adventure Builder sample application. Both are available from IBM as fully functional free trial versions. Details on installing and configuring these applications can be found in [Get started](#) .

Section 2. Get started

Overview

Let's get started. You have three things to accomplish in this section of the tutorial:

1. Install Performance Tester.
2. Install the Java Adventure Builder sample application.
3. Install the Adventure Builder Workspace for this tutorial.

Let's walk through these steps now.

Install Performance Tester

If you haven't already, install Performance Tester. A fully functional trial version of [IBM Rational Performance Tester V6.1](#) is available at no charge.

The Performance Tester trial is delivered in two pieces. The first is the CR2SQML.BIN file, which contains the compressed software, and the second is an application called the extractor. Download both and run the extractor application. The extractor application will process the .BIN file, expand it into the application installation files, and launch the setup for you. On the setup wizard, select **Install IBM Rational Performance Tester V6.1**.

During the Performance Tester installation, select any installation directory and accept all defaults.

Install Java Adventure Builder sample application

The Java Adventure Builder application is a sample application from the Java BluePrints Program at Sun Microsystems. It is an online vacation travel site where users can book various exotic vacations. It will be the target of the testing. You can install the Adventure Builder application in one of two ways.

Installing Adventure Builder with WebSphere® Express 6.0

This is the simplest of the two install options. IBM WebSphere Express 6.0 includes the Adventure Builder as a sample application. To install and configure the WebSphere Express and the Adventure Builder application, take these steps:

1. Download [IBM WebSphere Express 6.0 Trial Version](#).
2. You can install the software to any location on your hard drive.
3. When prompted, perform a Full Install. There are no other options.

4. At the end of the installation you will be prompted to **Launch the First Steps Console**. Accept the default **yes** and launch the console.
5. From the Console, select **Start the Server**.
6. When the server console appears, navigate to **Installable Samples > Application Server > Applications > Java Adventure Builder**.
7. Follow the instructions to install Java Adventure Builder on your platform.
8. Once complete, stop and then restart the server.
9. Navigate to `http://localhost:9080/ab/main.screen` to launch the application.
10. Explore the application and close when done.

Once downloaded, installation and configuration should take 10 to 15 minutes.

Installing Adventure Builder on your application server.

If you already have a running J2EE application server, you can download and install the Java Application Builder from the source code at <http://java.sun.com/developer/releases/adventure/>.

Instructions for deploying the application on various application servers will vary by server and are available from the different server vendors. While this method involves less download time, application configuration time could be significant.

Install the Adventure Builder Workspace

With the tool and the sample application installed, there's only one more thing you need to do before you can get some work done. Performance tester uses a workspace and project structure to store all of their performance tests, schedules, and results. For this tutorial, you'll build off of a workspace that already contains some assets. Take these next steps to launch Performance Tester and import the "Adventure Builder Workspace" for this tutorial:

1. Expand the [Builder Workspace zip](#) file included with this tutorial. The zip file can be expanded to any location on your hard drive. Be sure to note the location where you expand the files.
2. Launch IBM Rational Performance Tester by selecting **Start > All Programs > IBM Rational > IBM Rational Performance Tester V6.1 >**

Rational Performance Tester.

3. You are prompted to specify a Workspace. Click **Browse**, and point to the Adventure Builder Workspace directory you created in Step 1. Click **OK** to confirm your selection.
4. If this is your first time launching the project, you will be greeted by the Rational Software Development Platform Welcome screen. Feel free to explore the information available off of this page. Once you are done exploring, click the arrow in the upper right-hand corner to go to the Workbench.
5. The collection and arrangement of windows upon the workbench is called a perspective. The perspective that greets you now is called the Test Perspective. Rational Performance Tester V6.1 is merely one perspective in the larger Software Development Platform. This same workbench can be used for visual modeling, code development in a variety of languages, unit testing, and functional testing, to name a few.
6. To save some workbench real estate, close the Welcome view.
7. In your Test Navigator, you should now see the projects, tests and schedules that were stored in the Adventure Builder Workspace.

You're all set. You've installed Performance Tester and the Sample Java Builder Application. You've launched Performance Tester, and you've opened the pre-created performance test workspace for this tutorial. Now, let the games begin...

Section 3. Test recording

Overview

The process of analyzing system performance has three milestones:

1. **Test recording**
2. Test scheduling
3. Test execution and results analysis

This section of the tutorial covers test recording. Use the Performance Tester recording mechanism to create the test. Once the test has been recorded, use the Performance Tester Visual Test Editor to modify the test to ensure a realistic and accurate playback.

Record a test

The Adventure Builder application offers many packaged vacations for purchase on the Web site. The first step is to record a test that places an order for the Maui Survival Vacation offered on the site. Using Performance Tester's recorder, you can create the tests without having to write any code. With the recorder engaged, creating a test involves nothing more than navigating the site as a typical user would. In the background, Performance Tester will note all Web requests and responses and convert them into a test.

1. In Performance Tester, select **File > New > Record Performance Test** to record your first test.
2. Accept the default **Tests** folder and **Order_Maui_Survival.rec** name for the test and click **Finish**. The recorder starts.
3. You are greeted by the Welcome To Performance Testing page which reminds you to remove temporary files before you record. This is a good habit to get into. In Internet Explorer, select **Tools > Internet options**.
4. In the Temporary Internet files area, click **Delete Files**.
5. Check **Delete all offline content** and click **OK**.
6. Click **OK** to close the Internet Options window.
7. Go to the Adventure Builder home page by entering the URL `http://localhost:9080/ab/main.screen` in Internet Explorer.
8. Click **Maui Survival Adventure**. Use the link under **Island Adventures** on the left, not the link beside the image on the right.
9. Click the Select Package button to accept the default package.
10. Accept all the default values for the number of people, start date, number of days and rooms and click **Set Package Options**.
11. Click **I will provide my own Transportation**.
12. Review your order and click **Checkout** to order this vacation.

13. Click **Sign In** to sign in as the default user **j2ee**.
14. On the Billing Information page, scroll to the bottom of the page and click **Submit** to accept the default Billing Information. The Checkout Complete page opens and gives you an order ID similar to the one pictured here:
15. Close the browser.
16. Recording stops, and you are brought to your test in Performance Tester. Your test looks like this:

Examine your recorded test

Performance Tester displays your test in the Visual Test Editor. Follow these steps to examine some of the information presented to you in the Visual Test Editor:

1. The Visual Test Editor represents your test as a tree view, with the name of the test as the top level branch. Under the top level branch all of the individual pages are listed.
2. Click the second branch labeled **Available Adventure Packages**. Under every page-level branch you find all of the elements that comprise that page. In this case there are five. The first is the primary page request; the next four are images that appear on that page.
3. Click the second image request, which ends with **Island_Survival.gif**.
4. In the View beneath your test, click the Protocol Data Tab. The Protocol Data tab provides detailed information about the selected branch.
5. On the Protocol Data Tab, click the Browser tab. On this tab you'll see a rendering of the capture image.
6. Feel free to click other branches to see what additional Protocol Data is available for viewing.

Enhance your test with Response Verification

While functional validation is not the primary goal of a performance test, it is only prudent to incorporate some form of content validation into a test to ensure an accurate playback. Performance Tester offers a variety of verification point mechanisms for this purpose, which can be performed against individual elements, or against the test as a whole as you'll do here. During playback these verification

points will confirm proper execution of the test:

1. Right-click the top level branch of your test that reads **Order_Maui_Survival**.
2. Note there are three verification point options available: Enable Page Title VPs, Enable Response Code VPs, and Enable Response Size VPs. Any of these will suffice to validate that the responses during playback are valid, but for your purposes today, select **Enable Response Code VPs**.
3. On the Performance Test HTTP Editor window, click **OK** to confirm that 31 response code verification points were enabled.

Enhance your test with datapooling

Datapooling is the process by which an individual piece of data captured during recording is replaced with a pool of data values during test execution. The purpose of datapooling is to ensure a realistic playback by providing unique data to every virtual tester. In the test you specified that the trip should start in the current month. At playback time, you should vary this data so that each user starts in a different month. A datapool has been previously created for this purpose. It contains the numbers 1 through 12, numerically representing each month. Using the Visual Test Editor, you can easily link the test to the datapool and have Performance Tester use the values in the datapool at playback time:

1. Click the Adventure Package Details page branch.
2. In the Test Data area, you'll see some of the information you entered when you recorded the test. Specifically, note the start_month field. Here you see the start_month is **4** representing April.
3. The value **4** was captured during script recording. To ensure a more realistic playback, you're going to datapool this value. Datapooling this value ensures that different virtual testers will supply different starting months to the system. This forces the server to do more work to validate vacations starting in different months. If you did not do this, the server could potentially cache information about April availability, providing faster responses than it would if it had to check multiple months.
4. Highlight the start_month row.
5. Click the Datapool Variable button.
6. On the Select datapool column window, click **Add Datapool**.

7. On the Import Datapool window, click **start_month.datapool** in the matching resources section and click **Select**. The start_month datapool is a datapool containing the numbers 1 through 12 representing the twelve months. To save time, it has been previously imported into the tool for the purposes of this tutorial.
8. On the Select Datapool column window, click the cell, **Column: start_month** and click **Use column**.
9. You should now see that in the Test Data view the start_month variable is substituted with the value from the datapool, as follows:

Realistic data is a key element of a good performance test. Performance Tester has an import facility that enables you to use data from any source such as databases, text files, or spreadsheet data. This powerful feature requires no coding, enabling even the first time tester to create realistic test scenarios.

Test recording summary

You've done a lot of work, yet you didn't have to write a single line of code. You've captured a performance test using the Performance Tester recorder. Your test was displayed in the Visual Test Editor as a tree view where you enhanced it in two ways. First you added a verification point to ensure accurate playback. During playback, should any of the pages not return the same response code that was captured during recording, the test will fail and you'll get a notification in the event log. Second, you datapooped the starting month input value. This will ensure a more realistic playback where different virtual testers enter different starting months for their vacations. Nice work!

With test recording complete, it's time to move on to scheduling and execution. The lion's share of the work is done, but you're not finished yet.

Section 4. Test scheduling

Overview

The process of analyzing system performance has three milestones:

- Test recording
- **Test scheduling**
- Test execution and results analysis

This section of the tutorial covers test scheduling. You'll use the Visual Test Scheduler to represent the workload you want to run against the server. While test recording takes the lion's share of time, accurate test scheduling is essential to ensure a valid load. The goal here is to accurately emulate the load real users will put on the system.

Examine the Browse and Buy schedule

At this point, if you want to, you can simply run your test with 100 users, for example, and start to analyze your system performance. This, however, would not be a very realistic test. For any system it is not realistic to assume that all users will perform exactly the same actions in exactly the same sequence. The goal, you'll recall, is to accurately emulate the load real users will put on the system. Accordingly, you'll need to think a little bit about the types of users on the system and the actions these users will take.

1. In the Test Navigator, double-click the **Browse and Buy** schedule in the Schedules folder. The Browse and Buy schedule has been created for this tutorial as a starting point for your scheduling work.
2. The Browse and Buy schedule has defined two user groups. *User groups* are a mechanism to represent a user profile on the system. For this schedule, the two types of users that have been defined are *Browsers* and *Buyers*. When the test runs, 80% of the virtual testers are assigned to the Browsers group, and 20% to the Buyers group. Put another way, 80% of the users on the system will browse the site and 20% of the users will place orders for vacations on the site.
3. Fully expand all branches of the **Browsers** group. The Browsers group uses a random selector. As the name indicates, a random selector is used to have users randomly select between alternative courses of action. In this case, there's a 70% chance that a user will run the **Browse_Jungle_Adventure** vacation and a 30% chance that a user will run the **Browse_Mountain_Adventure** test.
4. Fully expand all branches of the **Buyers** group. The Buyers group uses a loop. The loop is used to make virtual testers repeat a series of actions. In the Buyers case, the virtual testers will run the **Order_Urban_Cowboy_Adventure** test, wait two seconds and then

repeat that sequence a second time.

5. Right-click the loop element in the Buyers group. Select **Add > Test** and select the **Order_Maui_Survival** test you recorded in the first section of this tutorial. Your schedule should now look like this:

Set schedule options

You've defined your workload. You've specified the types of users on your system, and the actions they will perform. Now, take the following steps to specify a few schedule-level options before you run your schedule:

1. Notice that up to this point you have not specified the number of users you wish to run for this test. That's because the schedule is independent of the number of virtual testers. This is a great feature which enables us to use the same schedule for different numbers of testers. You can use the same schedule for 50, 500, or 5,000 users without modification, as you try to find the system breaking point.
2. Click the top level of the schedule on **Browse and Buy**.
3. In the User Load area on the right, specify 5 for the Number of Users, the maximum number of users available for the trial version. Use 10 if you are using a licensed version of Performance Tester.
4. For a more realistic start up, check **Add a delay between starting each user** and specify a delay of 500 milliseconds. This ensures that not all users arrive at the site at the same time.
5. **Think Time** represents the time between page requests where a user spends absorbing the contents of a page. As you recorded your test, the time you spent looking at each page was recorded. During playback, you can have all users use exactly this time, or you can vary it. The best approach here is to add some variability, as no two users take the same amount of time to analyze any given page. In the Modify the duration of think time delays list, select **Vary the think time by a random percentage**. Set a lower limit of 80% and an upper limit of 120%. This randomly varies think times by +/- 20%.
6. **Execution History** settings impact the level of detail that will show up in the Execution History report. Accept the default of **Page** level reporting and the number of users for data sampling at **5**. Note, when you run a larger test, it is generally more typical to sample from a percentage of users, usually around 20% to 30%.

7. The statistics section is similar to the Execution History in that it is here you set the number of users to use to sample for your reporting data. Accept the default of all users and 5 seconds for our purposes. For a larger test, you might reduce your sample size to 20% to 30% of the population.
8. Select **File > Save** to save your work.

Test scheduling summary

The goal of scheduling is to accurately emulate the load real users will put on the system. The Visual Schedule Editor makes this possible by providing a rich interface to define user profiles and user workloads. User profiles are set as relative percentages of the population, ensuring the same schedule can be used for varying workloads. Real user workloads can be accurately emulated by including loops, delays, and decision structures. Once your schedule is assembled, various options exist to ensure realistic ramp up and think time emulation. A variety of settings are also available to configure the amount of data collected for reporting.

You're almost there. The only thing left to do now is to run the schedule and analyze the results. Keep going!

Section 5. Test execution and results analysis

Overview

The process of analyzing system performance has three milestones:

1. Test recording
2. Test scheduling
3. **Test execution and results analysis**

This section of the tutorial covers test execution and results analysis. You'll run your test with 10 users and examine how to interpret the Performance Tester reports to analyze system performance.

Execute the test

Recording is complete; scheduling is complete. Let's run the test:

1. In the Test Navigator, right-click the **Browse and Buy** schedule and select **Run > Performance Schedule**. Performance Tester performs some overhead tasks and launch the test.
2. Once your test is running, the Performance Report opens and you can see live feedback about your running test. A great feature of Performance Tester is its ability to display live reports. This comes in especially handy when a test fails for some reason. It is not necessary to wait for a test to complete to realize a problem has occurred. With Performance Tester, once you spot a problem, you can cancel the test at any time, fix your problem and start over.

Analyze test health

Before you look at your response statistics, you should first ensure that you have a valid run. Take the following steps to ensure you have good data:

1. The first report that greets you is the overall report and it should look like this:
2. If you have three bars that all read 100, then congratulations, your run is healthy.
3. The first, blue bar is telling you that 100% of your page status codes returned as expected. During recording, Performance Tester noted the server response code for every page you hit. During playback, Performance Tester compares the results received by all virtual testers against these numbers. Any mismatches, such as a **500 - Server Too Busy** response instead of the typical **200 - OK** response will be reflected here.
4. The second, blue-green bar provides the same information at the page element level. Recall that the page elements include the actual page HTML as well as all of the images and other objects on the page.
5. The third bar is a summary of the results for the verification point you set back when you edited your script. This is telling you that all of your verification points have passed.

6. Typically, you'll want to see all of the bars on this report above the 90% or 95% levels. If you do not, examine the Server Health Summary and Server Health Tabs to see where your test had problems.
7. For additional problem information, right-click your result set in the Performance Test Runs view and select **Display Execution History**.

Analyze system performance: Page Throughput

Once you have ensured that you have valid data, it's time to look at that data to see what it reveals about your system performance:

1. The first report to look at is typically the Page Throughput report:
2. This report has two graphs: the User Load and the Page Hit Rate.
3. The User Load graph shows you how many users were active at any give point in time. Your first glance at this report should be just to validate that your users ran as expected. Later on, you might return to this report to see how many users were active at a given point in your test where you experienced a bottleneck.
4. An interesting thing to try while looking at this report is to add users to your run. In the Performance Test Runs view, click the third icon from the right to add users to your run. The trial version of Performance Tester limits runs to 5 users, so if you are using the trial version, you will have to wait until you get the full version to try this feature as you are already running with the maximum number of users.
5. The Page Hit Rate graph gives an idea of general server response. If you look at this report and see that for the majority of the points the **Page Attempt Rates** equal the **Page Hit Rates** then you'll know your server was quickly responding to all requests. If there are numerous mismatches, your server was having a hard time keeping up to the traffic.

Analyze system performance: Response vs. Time Summary

1. Next, move on to the Response vs. Time Summary reports:
2. This report gives an overview of general system response by providing two graphs: Page Response vs. Time and Page Element Response vs. Time.

3. The Page Response vs. Time report shows the average response time for all pages throughout the test. Times shown are in milliseconds, and in the preceding graph the page response time fluctuates in range roughly between 100 milliseconds to 600 milliseconds. Clearly the server could handle the load.
4. The Page Element Response vs. Time report shows the average response time for all elements at various points during the test. As would be expected, response times are even faster, fluctuating primarily between 75 and 150 milliseconds, with one outlier around 400 milliseconds. A page's response time is the sum of response times for all elements on the page, so it's no surprise that page elements come back much faster than the page.

Analyze system performance: Page Performance Report

1. Next, move on to the Page Performance report:
2. This is one of the most important reports available to you. It shows the average response time for all pages throughout the test.
3. In looking at the results above, the average response time for every page was less than 600 milliseconds, some as low as 98 milliseconds. While there's no hard and fast rule about maximum page response times, in general, page response anxiety starts around 3 seconds. Page response intolerance begins around 8 seconds.

Analyze system performance: Response vs. Time Detail

1. Next, move on to the Response vs. Time Detail report:
2. The previous Page Performance report can be deceiving. Average times can mask periods of abnormally fast or slow responses, especially over a long test. It is essential to look at the Response vs. Time Detail report to examine response times throughout the test.
3. This report shows how the average response time changed for the 10 slowest pages throughout the test. You can see here that while on average all pages came in under 600 milliseconds, there are several points where page responses were slightly higher. Here, nothing is greater than 800 milliseconds, so there is no great concern, but that might

not always be the case.

4. Note also a typical pattern here in that initial response times for most pages are slower than subsequent response times. This is a reflection of server caching. After a server initially serves up a page, that page will typically remain in the server's cache memory. Subsequent responses can come from this cache memory, an action which will be notably faster than the original fetch from disk. You see this pattern here as performance lines slope from high to low at the beginning of the test.

Sample performance problem

Unfortunately for us, the Adventure Builder application is a solid, well performing sample application. While it serves your purposes well for creating and running tests, you'd be hard pressed to find a performance problem with this application.

Thankfully, there's no problem the IBM engineers can't create! In our labs, we took the Adventure Builder application and sabotaged the Checkout page. We purposely inserted poorly performing code, so that you could see how a performance problem would manifest itself with Performance Tester. Take these next steps to look at the results of a performance test that was run against this disabled application:

1. In the Performance Test Runs view, right-click **Checkout Bottleneck results from April 29**, and select **Display Performance Report**.
2. Click the Page Performance Tab to view the **Page Performance** report.
3. The Page Performance report reveals that the average response time for the Checkout page is about 3.3 seconds. On its own, 3.3 seconds is not terrible, though it is significantly higher than any of the other pages on the site. Accordingly, this discrepancy warrants further investigation.
4. Open the Response vs. Time Detail report to investigate the issue further:
5. This is a very good example of why looking at the Page Performance report on its own is not enough. On the surface, the 3.3 second response times from the Page Performance report is not too alarming. However, drilling down with the Response vs. Time Detail report you can see that the Checkout page response times hit highs of over 10 seconds. Originally, the page was performing well, but somewhere around the 50-second point, response times increased significantly. This is exactly the type of performance problem you want to find before your customers do. With the problem identified, developers can now use the IBM Rational Performance Optimization Toolkit to drill down from this data into the

actual source code to find the specific code that is causing the problem.

Test execution and results analysis summary

Test results analysis requires the user to interpret the reports provided by Performance Tester both during and after the performance test run. Performance Tester provides several reports which progress in level of detail to help you determine the exact page and the exact point in time where the performance problem occurred. Reports drill down from high-level data such as the average server response time to low-level data such as the average page response times at every point during the test. Using this information Performance Tester can find problems and pass them along to developers. Development, in turn, can then obtain the free download of the IBM Rational Performance Optimization Toolkit to drill down in the source code to pinpoint the exact line of code causing the bottleneck.

Section 6. Summary

Congratulations, you've accomplished a lot. You have:

- Recorded a performance test.
- Created a multi-user performance schedule.
- Analyzed performance test results.

And all of this, as promised, without writing a single line of code. IBM Rational Performance Tester was designed for Day 1 productivity, and by completing your first full performance test, you have achieved that goal today. By working against the sample application, you have witnessed and overcome all of the primary challenges of performance testing. You now have all the knowledge you require to go forth and analyze your system's performance. Go get 'em!

Resources

Learn

- [Additional Rational tutorials](#)
- [developerWorks Rational Performance Tester page](#)
- [developerWorks blogs](#)

Get products and technologies

- [IBM Rational Performance Tester V6.1 trial download](#)
- [IBM WebSphere Express 6.0 trial version](#)
- [Java Adventure Builder](#)

Discuss

- [Participate in the discussion forum for this content.](#)

About the author

Brian Bryson



Brian Bryson joined IBM Rational in 1995 after having spent several years in software development. Since joining IBM Rational, he has held various positions supporting software quality tools from consultant to technical marketing. He is currently the automated software quality Evangelist for IBM Rational and spends his days speaking to customers, partners, and analysts on all matters pertaining to Software Quality. He has spoken at numerous conferences and published many articles, the most recent on patterns of success in test automation in the [December 2004 issue of Software Test and Performance](#).