



**Michael Meyers
Chris Beyers
Don Weber**

An Experience Using Rational Performance Tester to Benchmark Oracle EnterpriseOne

The Power Systems™ (i) Benchmark Center used IBM® Rational® Performance Tester (RPT) to test Oracle®'s JD Edwards® EnterpriseOne application. This IBM Redpaper publication provides detailed information about executing performance tests in this specific environment. The information provided will help make testing of upgrades and deployments in this environment a seamless effort.

Introduction and overview

This paper provides an overview of how to use IBM Rational Performance Test to execute performance tests against Oracle EnterpriseOne (JD Edwards) applications that will help deliver successful upgrades and deployments critical to the business. Hints and tips that are useful in any test environment are also included. Highlights of the paper include:

- ▶ Recording strategies of JD Edwards EnterpriseOne
- ▶ Enhancement of JD Edwards EnterpriseOne test scripts including naming conventions, grouping pages, and looping strategies
- ▶ Techniques in schedule creation of a JD Edwards EnterpriseOne performance test

Audience

This paper is intended for an audience that already has a working knowledge of RPT and its terminology. We assume this background knowledge:

- ▶ General working knowledge of RPT
- ▶ General working knowledge of your Oracle EnterpriseOne environment
- ▶ Installation of the IBM Rational Performance Tester (RPT) v7.0.1.1 or later
- ▶ An Oracle EnterpriseOne v8.1.2 environment

If you would like more information and a basic understanding of applying RPT to enterprise application testing, refer to *Using Rational Performance Tester Version 7*, SG24-7931¹.

Project background

An IBM client has worked with the IBM Power Systems (i) benchmark center to roll out Oracle's JD Edwards EnterpriseOne Version 8.12. They have been users of previous versions of JD Edwards. The decision to move to the latest architecture began January 2007 and resulted in a benchmark to test the out-of-the-box functions of Version 8.12. With the successful completion of the early 2007 benchmark, they moved forward with deployment of customized JD Edwards changes to support their business. With this phase nearing completion, the client wanted to execute a second benchmark to verify that the customizations would still support their batch and interactive performance requirements.

One of the major differences between the 2007 and 2008 benchmarks was that the client engaged the Power Systems (i) Benchmark center to help create the interactive scripts. Our testing tool of choice for Web-based workloads is Rational Performance Tester.

The benchmark completed successfully and all customer objectives were met. RPT performed well during the benchmark and our center gained knowledge and experience with the product.

Recording performance test scripts

Initiating the recording of a JD Edwards EnterpriseOne test is no different from other recordings and can be started either using a right-click from the Performance Test project

¹ <http://www.redbooks.ibm.com/abstracts/sg247391.html>

from the toolbar, or from the File menu item. For JD Edwards EnterpriseOne recording, select the **HTTP Recording** recorder option, as indicated in Figure 1, and click **Next**.

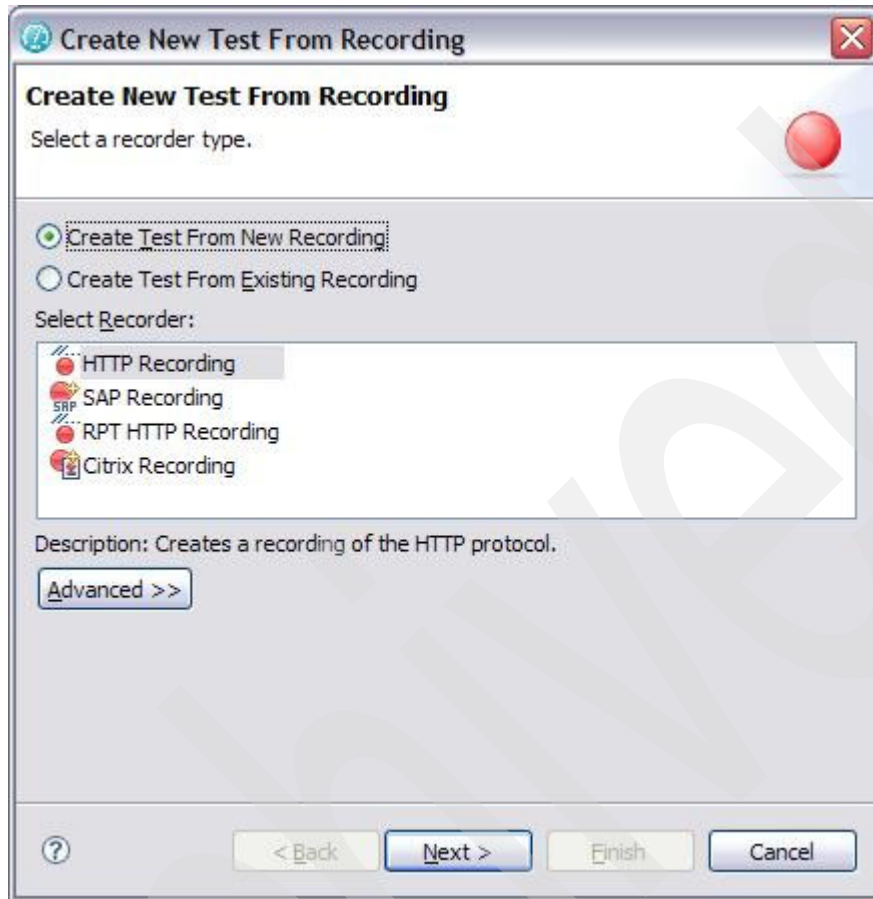


Figure 1 IBM Rational Performance Tester: Create New Test window

In the following window select the appropriate folder within the performance test project, provide a test name, and click **Finish** to continue.

Once the recording starts, the tester will perform actions on the JD Edwards EnterpriseOne Enterprise application opened in the browser. When the test activities are complete, closing the browser stops the recording and the test is generated. Figure 2 shows a recorded sample test.

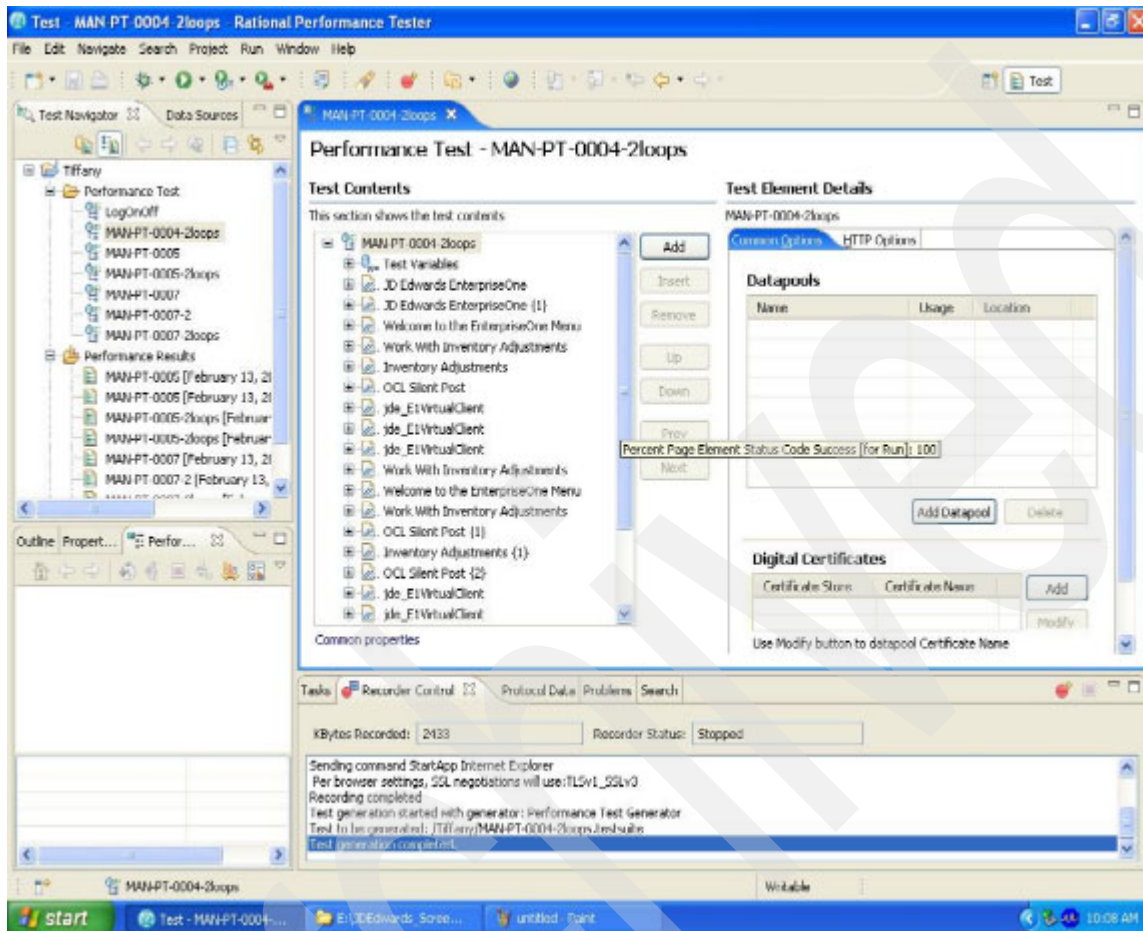


Figure 2 Recorded performance test scenario

Figure 3 shows that the login pages are isolated from the action of the test by a loop. By looping on certain test pages, the pages that are of interest can be repeated to provide a steady state of behavior. In this example, the pages of interest are executed once without a loop and once within a loop for four iterations. This allows two transactions to be evaluated. The first transaction models the scenario without data cached in the browser, and the second transaction models the same scenario with data cached in the browser. This provides two measurements of behavior for the same test scenario with different input criteria.

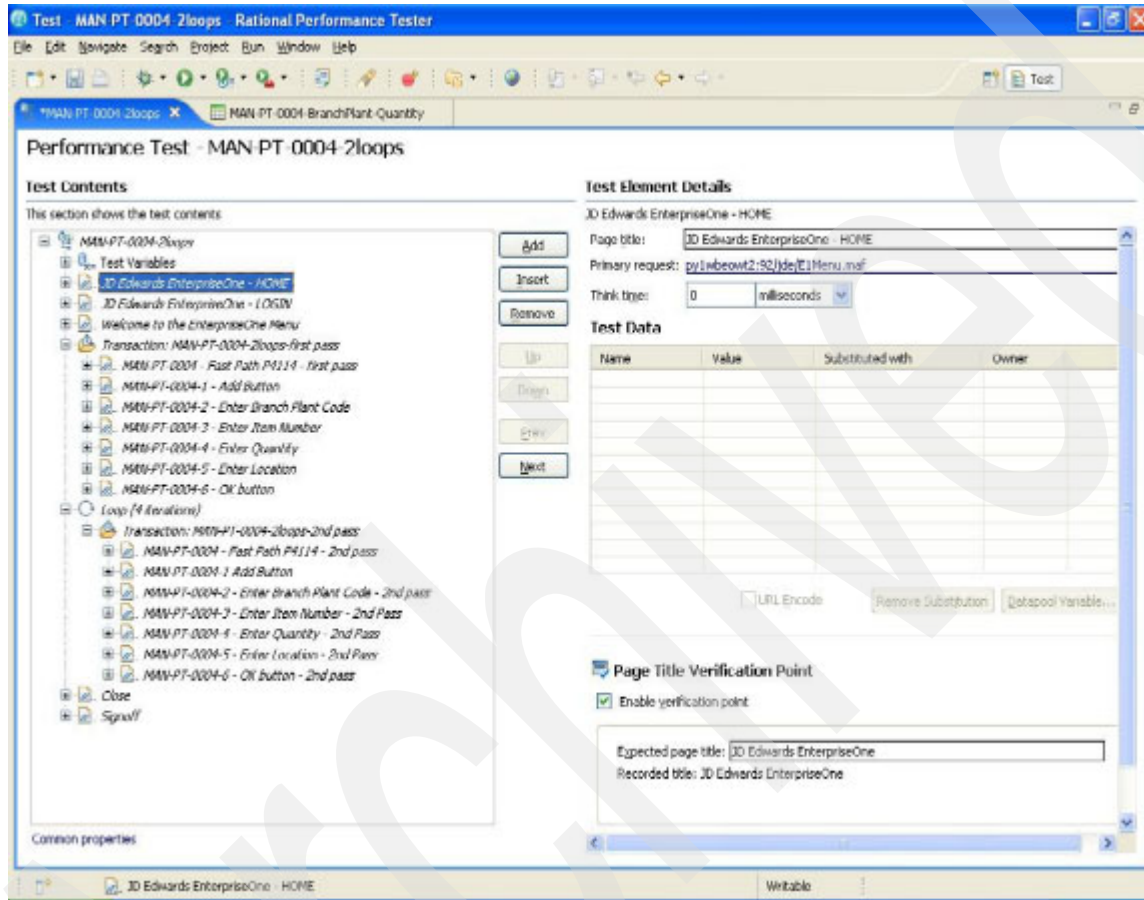


Figure 3 Loop usage in the test scenario

Rational Performance Tester allows accurate workload emulation using schedules. Hence, after you finish editing the test, you create a schedule. You add user groups to the schedule, then you add appropriate tests to each group to emulate a task. In addition, execution can be distributed over several machines. In our scenario, we have specified to execute User Group 4 from our test agent, called TestSrv1.

When recording a test, it was found that a single page will sometimes be recorded as one page, but other times as two. Also, sometimes two pages are recorded as one. A method was found that seemed to improve the recording accuracy when it comes to grouping page elements correctly under the page itself. Navigate to **Window** → **Preferences** and from the menu select **Test** → **Performance Test Generation**. Under the Protocol tab, you will see Generate new page if delay between requests. Make a note of this number (Figure 4).

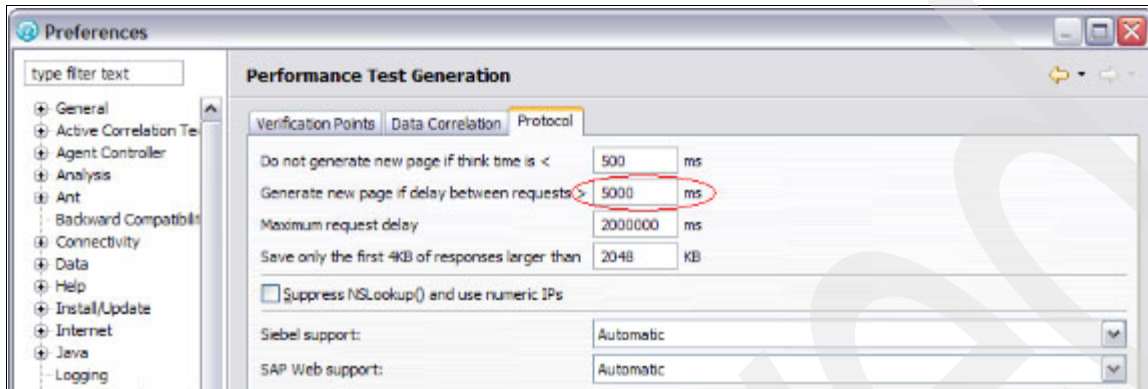


Figure 4 Performance test generation settings

Now, while recording, watch the Kbytes recorded in RPT (Figure 5). When the value stops incrementing, wait for the previously noted delay before continuing to navigate the application. This method seemed to greatly improve the accuracy of the recorded page groupings.

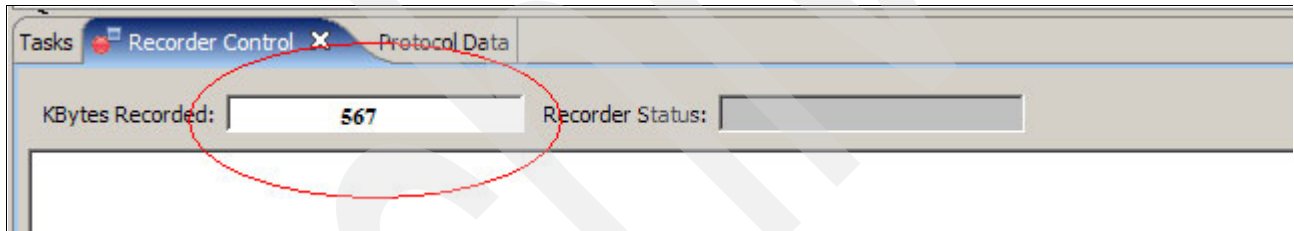


Figure 5 Kbytes recorded

A complexity within the Oracle Application that may be related to recording issues is the fact that the browser activity icon and status bar often indicate that a page is completely loaded (Figure 6). Sometimes the application indicates that a user can continue working while applications load (see Figure 7 on page 8). At other times, similar messages have very different characteristics regarding browser activity (Figure 8 on page 8). These different page elements, although often viewed as a single page by the user, may be viewed as multiple pages or single pages by RPT. Often how RPT views them is simply due to the pacing at which the user navigates the application.



Figure 6 Browser activity

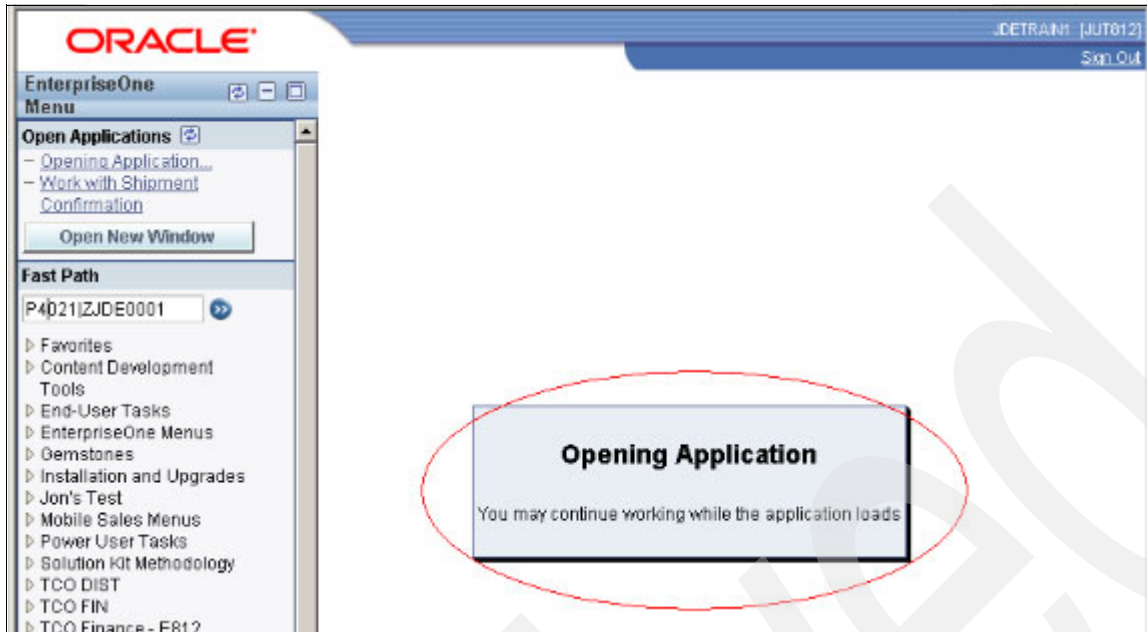


Figure 7 Opening application while continuing to work

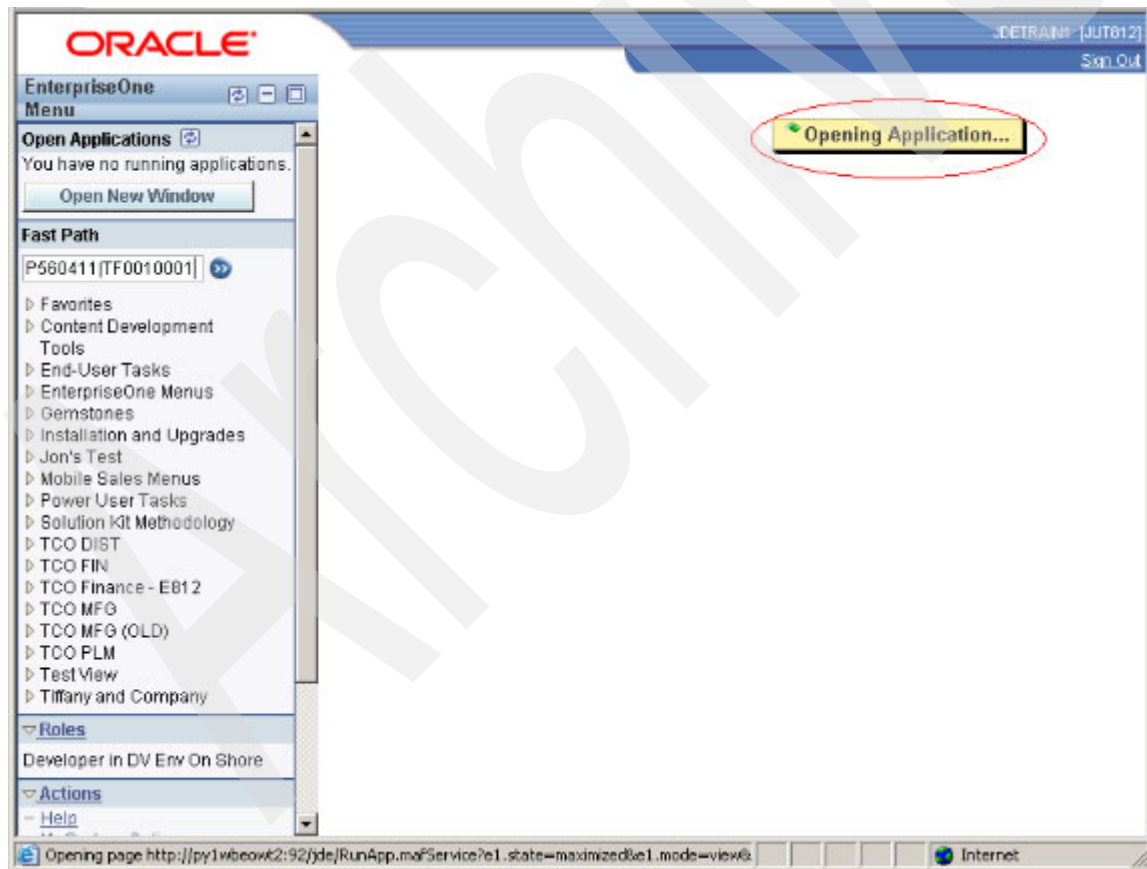


Figure 8 Opening application in a different situation

Modifying and enhancing performance test scripts

At this point you should have standard test scripts created that showcase primary business transactions. The next step is to modify these test scripts to achieve desirable yet realistic results by focusing on some RPT performance enhancements and JDE EnterpriseOne issues.

Helpful naming conventions for page titles

After a recording, you will see generic page names generated by RPT that usually require a change to a more accurate description of the page. It is important is to increment your pages with a unique naming convention so that you can easily group and sort your transactions in the results, as seen here:

- ▶ JD Edwards EnterpriseOne - Home
- ▶ JD Edwards EnterpriseOne - Login
- ▶ FIN-PT-0001_01 - Step1
- ▶ FIN-PT-0001_02 - Step1

Note: Sorting by label helps with analysis.

Removal of page element delays

RPT records the time for each page element request to be satisfied. If there are delays on the server side that lead to delays in the page elements requests, these delays may adversely affect the evaluation of a performance test. In Version 7.0.2, the page element delays may be edited globally using an option at the test level under the HTTP Options tab, as shown in Figure 9.

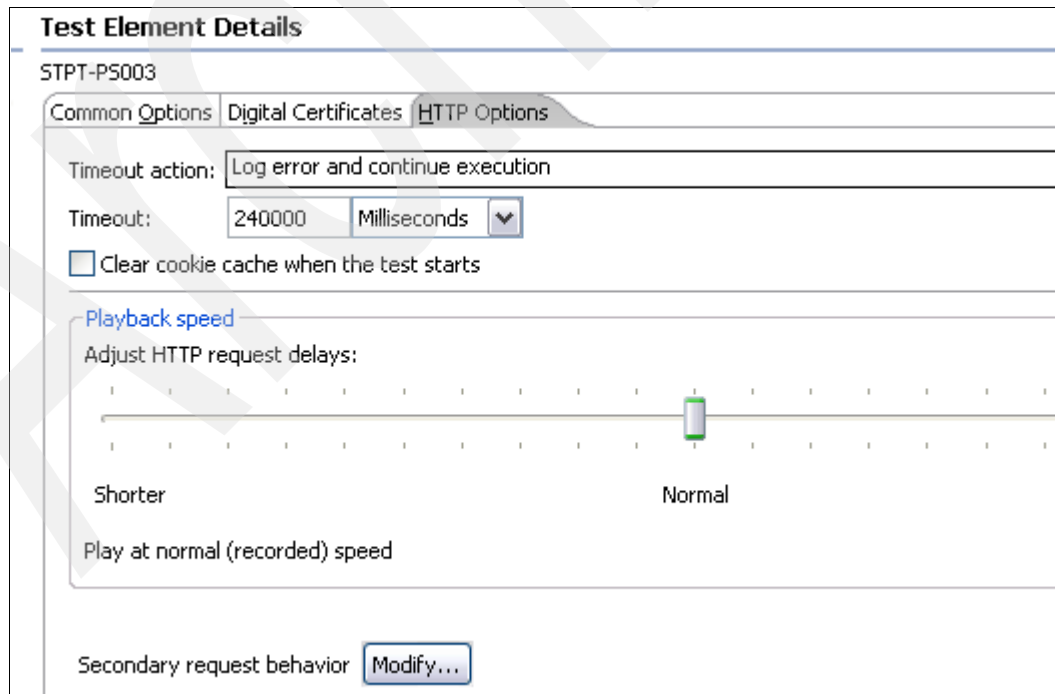


Figure 9 Editing page element delays

It is important to understand why the page element delays are present, and when and why you may choose to reduce or remove the page element delays. The slider bar allows you to adjust the delays to determine whether there is any effect on the playback of the test scripts.

Figure 10 shows the version number highlighted in the test data. The fast path would be the applDversion.

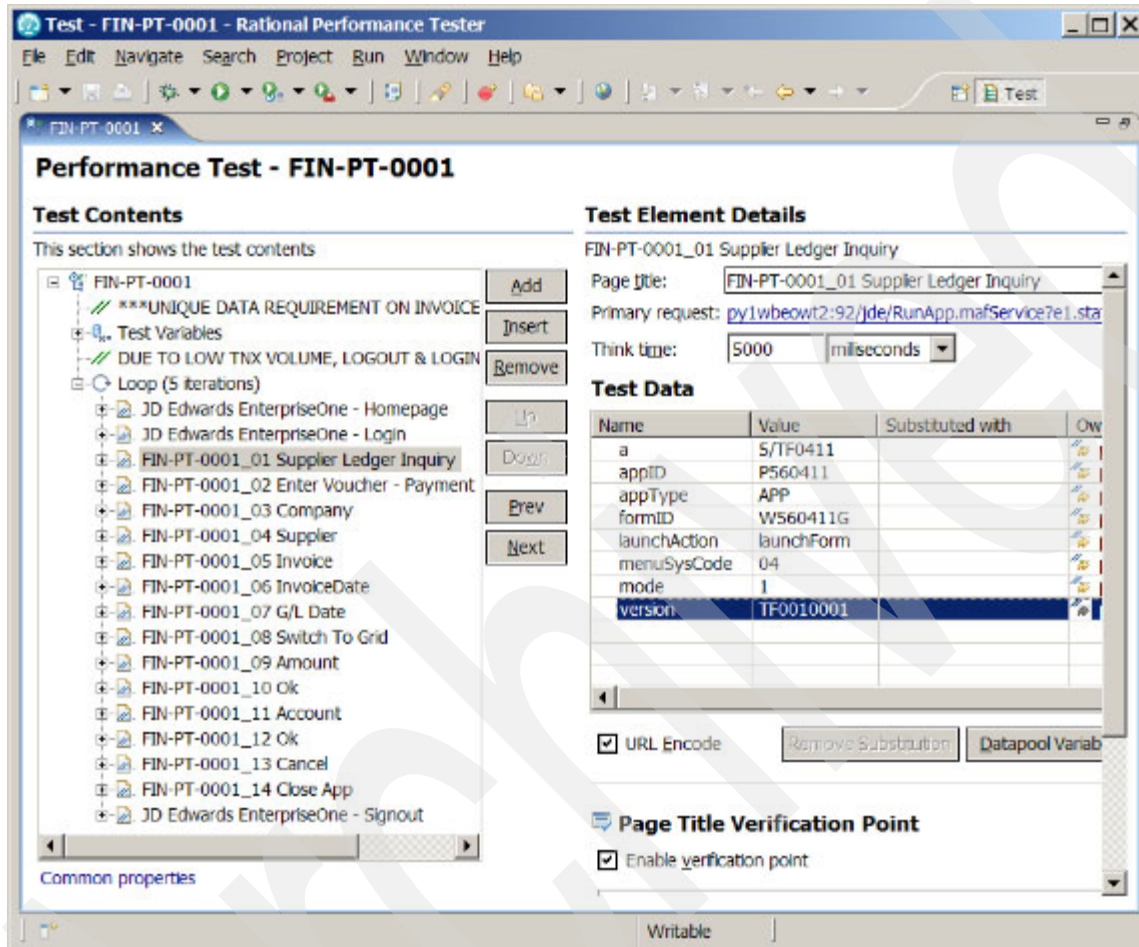


Figure 10 Version number

To merge the pages, press Ctrl+ left-click to select the two pages. Then right-click and select **Merge Pages** (Figure 13).

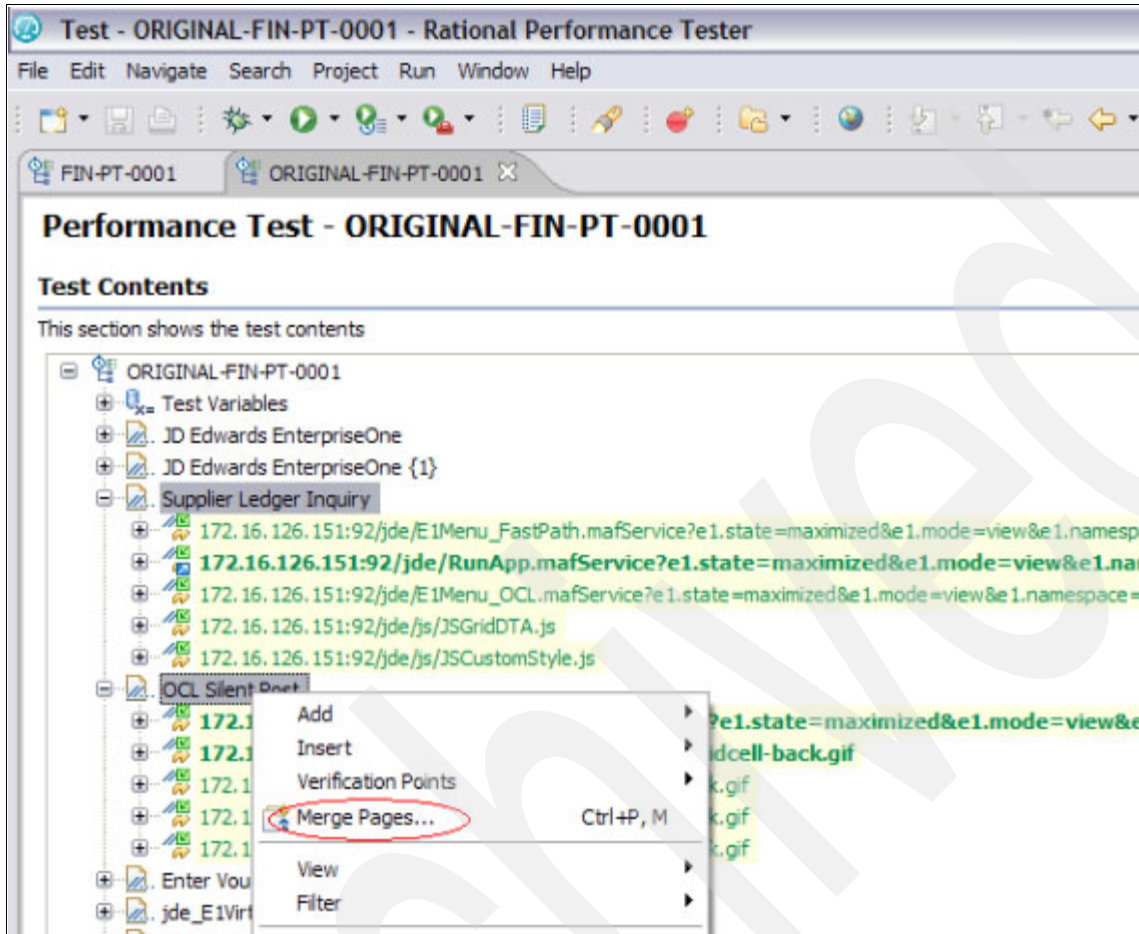


Figure 13 Merging pages

When pages are merged, if the page title has been customized, it will be reset to its originally recorded value. If you have already customized the name, make note of the name before merging the pages.

Ungrouping pages is just as easy. Highlight a page element that you would like in a new group that is the break-point, right-click, and select **Split Page Here** (Figure 14.) You are then presented with a Performance Test Editor window that describes the split that is going to take place. Verify and select **Finish**.

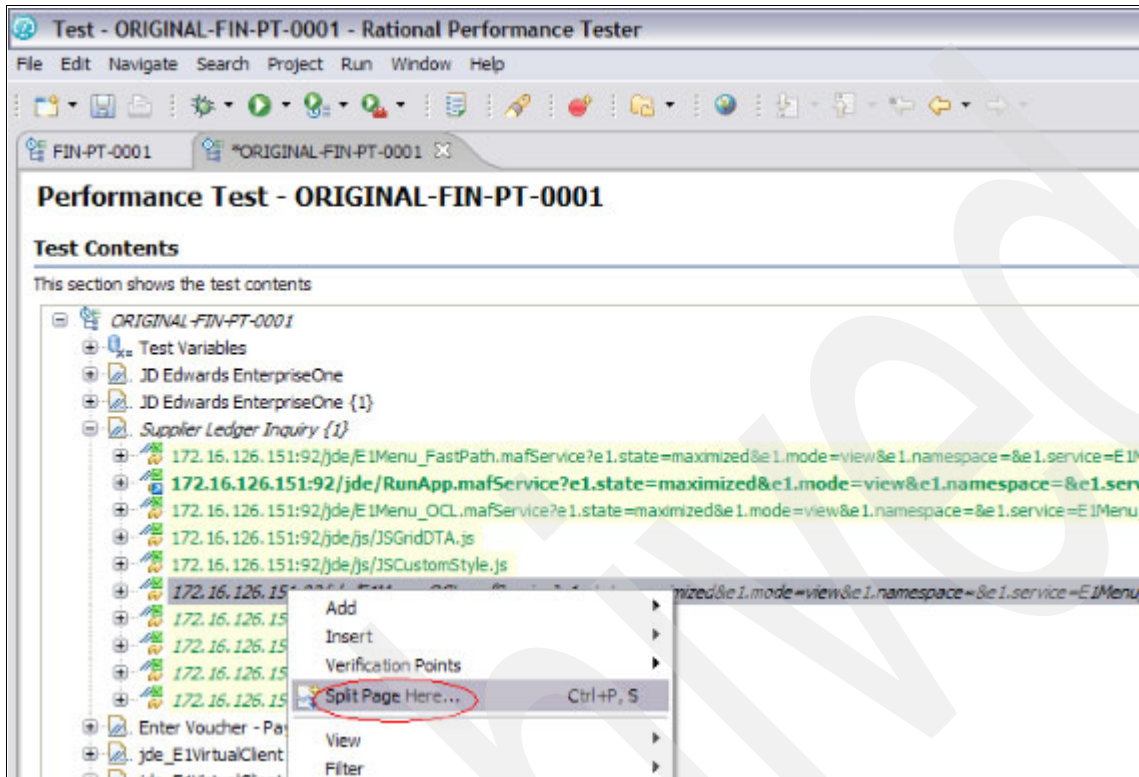


Figure 14 Split page here

Figure 15 shows the options when splitting or merging and how you can select to group the top part with the bottom or vice versa.

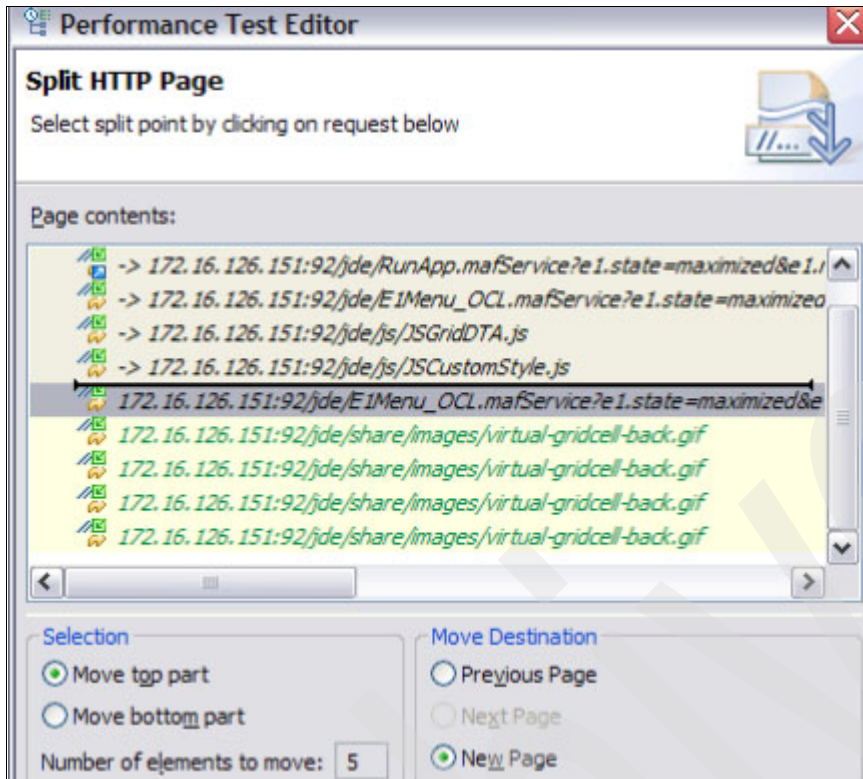


Figure 15 Split HTTP page

Timestamps

Timestamps are a unique identifier used in many applications that can be either used for the server side or for the client side. After you record, you will need to check each page element to see whether any timestamps have been included and handle them respectively. The timestamps seen in the Oracle EnterpriseOne application were constructed of 13 characters that represent the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time (UTC), according to the system clock.

Substituting a dynamic timestamp in a single instance in the recording requires a few steps:

1. Highlight a page element in which you see a timestamp in the test details (Figure 16).

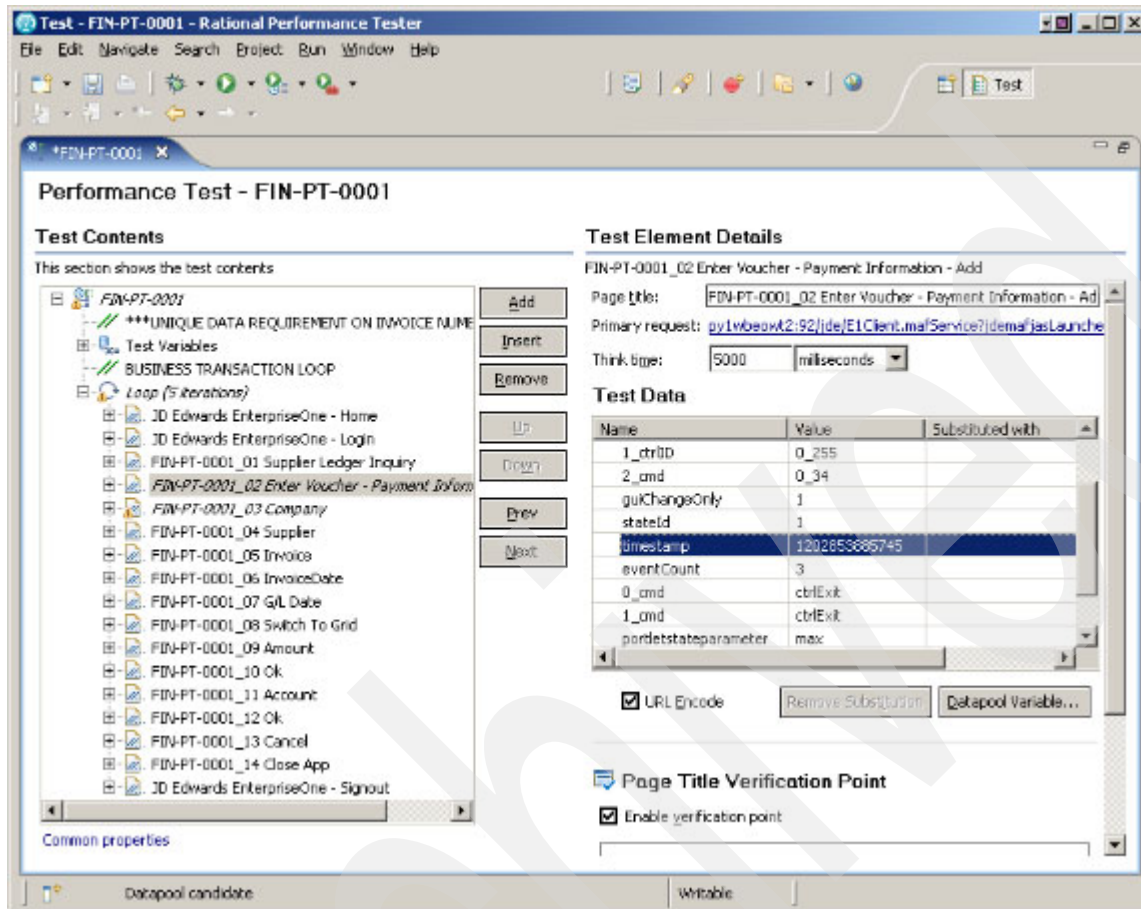


Figure 16 Page element in which there is a timestamp within the test data

- Right-click anywhere in the test data and select **Show References** to display all instances where a timestamp is used in that request (Figure 17). Notice that there may be more timestamp elements that need to be correlated than before this action was executed. If there is more than one timestamp, they will be all grouped together.

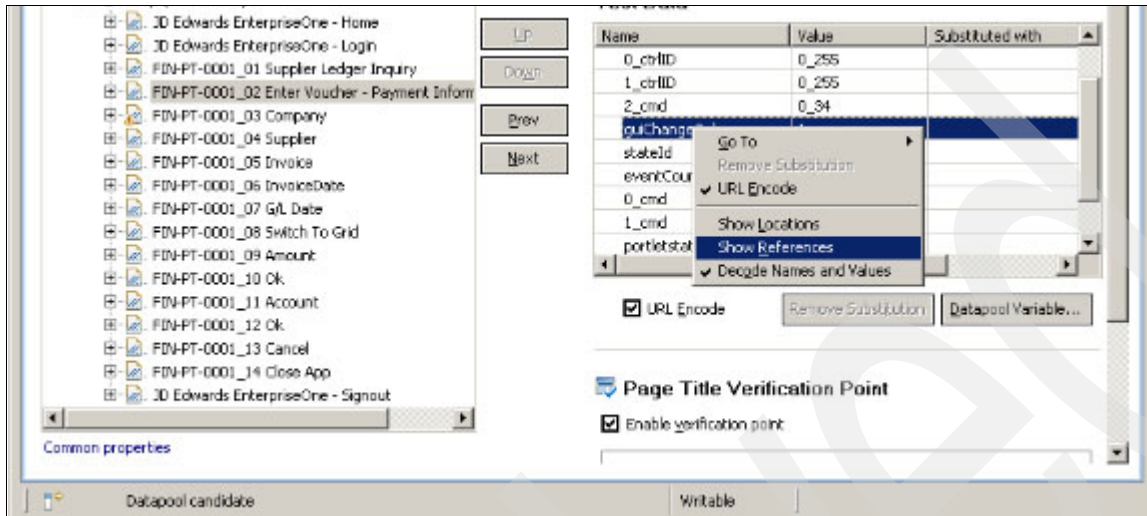


Figure 17 Show references to incorporate all instances of timestamps

- Highlight the timestamp that you want to correlate with a dynamic timestamp and click the **Substitute From** button. This opens up a drop-down list from which you must select **Built-in Datasources** (Figure 18).

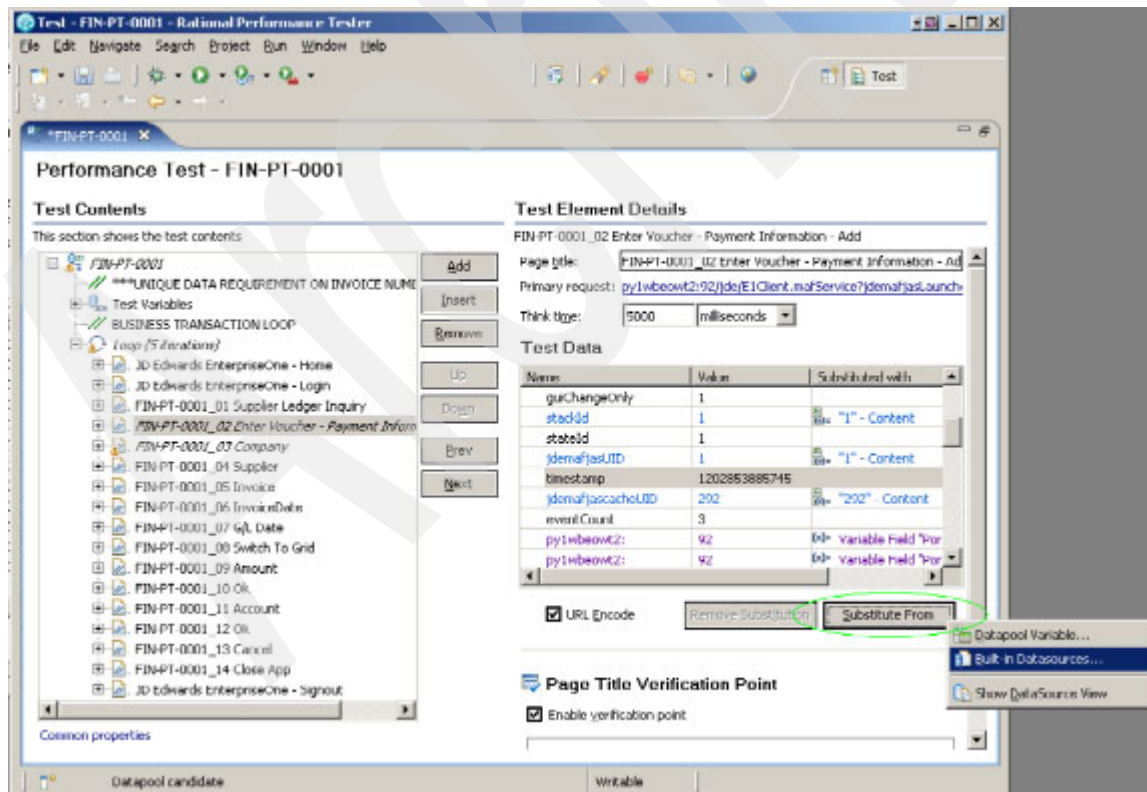


Figure 18 Substitute the timestamp with a built-in data source

- The final step is to walk through the wizard that gets initiated by selecting **Timestamp** under Generic Datasources (Figure 19). The rest of the wizard involves leaving the default and clicking the **Next** button to complete the substitution. If you want to replace a timestamp with a previously substituted timestamp, select it in the first wizard panel.

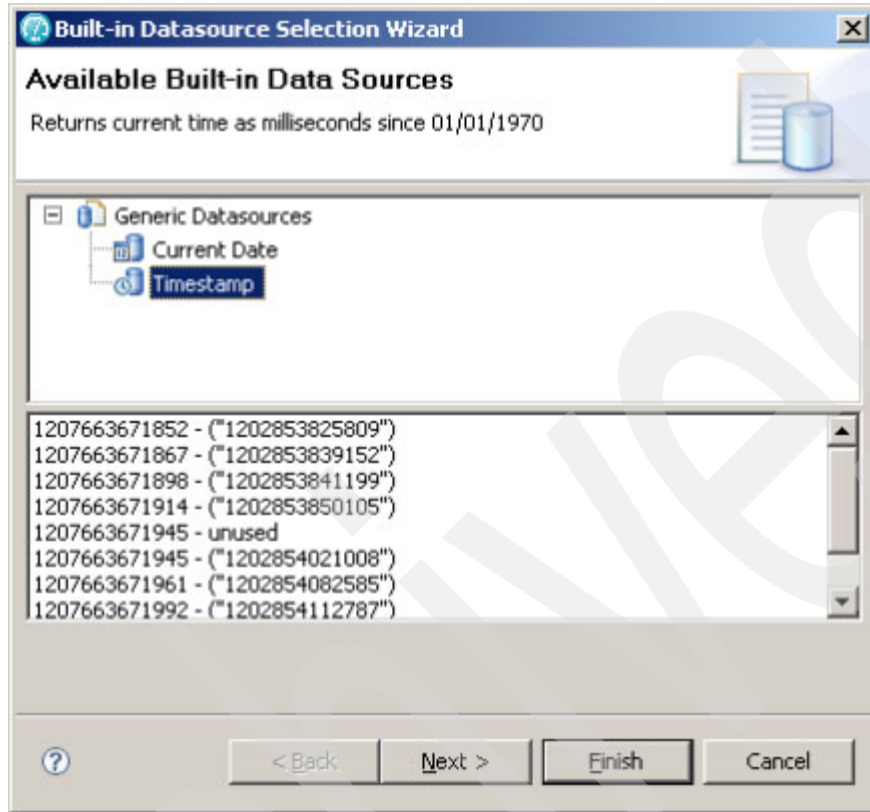


Figure 19 Timestamp wizard

Line items/multiple line items in grid data

Any time that you deal with line items or any situation in which you are inputting multiple rows (Figure 20), you must be on the lookout for incrementing counters that are not automatically correlated in RPT. For instance, in adding new line items, each element's details will have specific values that will increment with each new line item. The orange highlighted numbers (1.0.18 and 1.1.18 in Figure 21 on page 19) are values that were incrementing from the first line item to the second line item. The way that you recognize this is by recording two line items and comparing their data. These orange values are the data pieces that have already been substituted using java code.

The screenshot shows the 'Invne Transfers - Inventory Transfers' form. The form includes fields for Document Number, Document Type (IT), Transaction Date (02/18/2008), GL Date (02/18/2008), Explanation, Batch Number (2121062), From Branch/Plant (M82), and To Branch/Plant (M82). Below the form is a grid with the following data:

Item Number	Item Description	Quantity	UM	Secondary Quantity	Sec BoM	From Location	From Lot/Serial	To Location	To Lot
40173930	DIA 8MB 2.00-2.48 FG F	1.000	EA			D-FC-YH-V	200411220475	G-OP-S1-01	2004

Figure 20 New line item

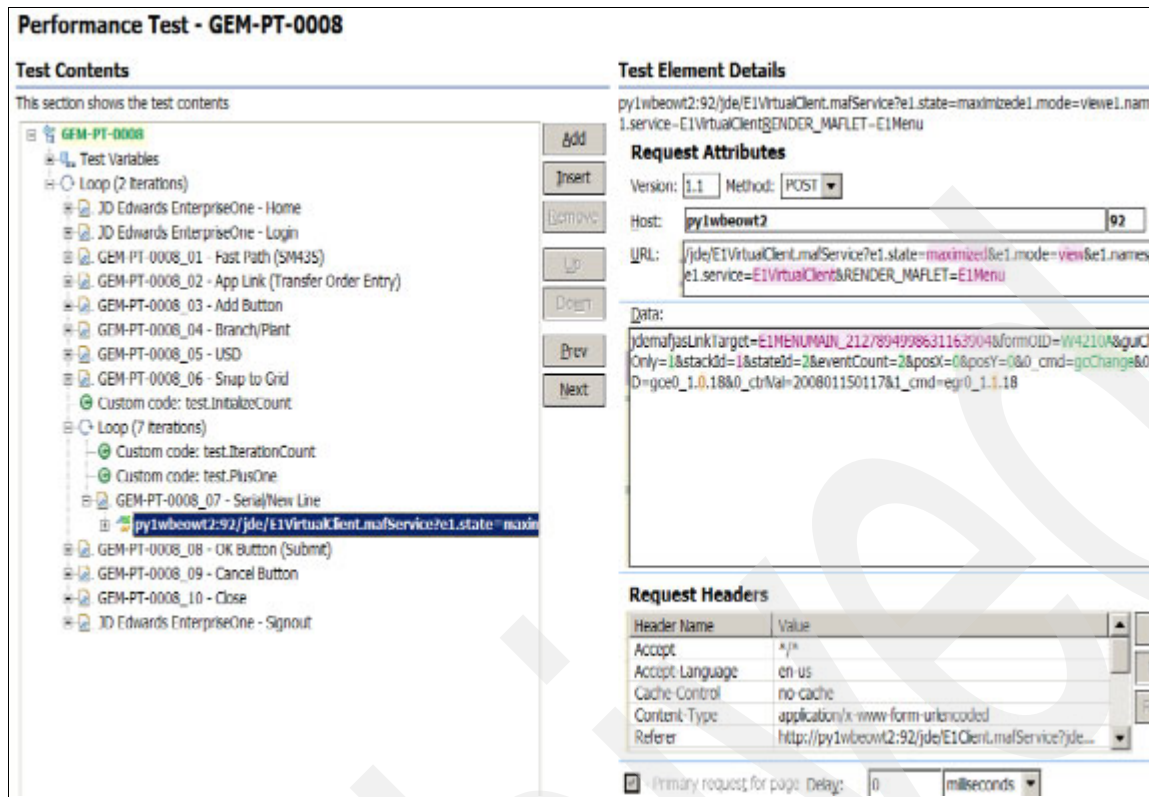


Figure 21 Test element data with incremental variables

The steps we took to get to this point are:

1. Write the Java™ code that simply returns incrementing numbers, as seen in Example 1, Example 2 on page 20, and Example 3 on page 21.
2. Highlight the piece of data in the element details that you want to substitute from left to right.
3. Right-click that highlighted section and select **Substitute From** → **Custom Code: class name**, where *class name* is the Java code that you wrote for that particular data substitution.

Example 1 Declare variable and Initialize

```
package test;

import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * @author unknown
 */
public class InitializeCount implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public InitializeCount() {
    }
}
```

```

/**
 * For javadoc of ICustomCode2 and ITestExecutionServices interfaces, select 'Help Contents'
in the
 * Help menu and select 'IBM Rational Performance Tester TES'.
 */
public String exec(ITestExecutionServices tes, String[] args) {

    Integer i = -1;
    IDataArea vda = tes.findDataArea(IDataArea.VIRTUALUSER);
    vda.put("InnerLoopCtr", i);

    return null;
}
}

```

Example 2 Get, increment, and save variable

```

package test;

import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * @author unknown
 */
public class IterationCount implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public IterationCount() {
    }

    /**
     * For javadoc of ICustomCode2 and ITestExecutionServices interfaces, select 'Help Contents'
in the
     * Help menu and select 'IBM Rational Performance Tester TES'.
     */
    public String exec(ITestExecutionServices tes, String[] args) {

        IDataArea vda = tes.findDataArea(IDataArea.VIRTUALUSER);
        Integer i = (Integer) vda.get("InnerLoopCtr");
        i++;
        vda.put("InnerLoopCtr", i);
        String result = Integer.toString(i);
    }
}

```



```

        return result;
    }
}

```

Example 3 Get, increment, no save

```

package test;

import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * @author unknown
 */
public class PlusOne implements
    com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public PlusOne() {
    }

    /**
     * For javadoc of ICustomCode2 and ITestExecutionServices interfaces, select 'Help Contents'
in the
     * Help menu and select 'IBM Rational Performance Tester TES'.
     */
    public String exec(ITestExecutionServices tes, String[] args) {

        IDataArea vda = tes.findDataArea(IDataArea.VIRTUALUSER);
        Integer i = (Integer) vda.get("InnerLoopCtr");
        i++;
        String result = Integer.toString(i);

        return result;
    }
}

```

Interdependencies of data

Understanding how the application interacts with your data is critical in making any progress with the tests that you develop. In this Oracle Enterprise application we noticed that some pre-populated fields differed based on the version of the specific program that we executed (see “Program versions” on page 23). If we initiated an older program version, we would be presented with a different flow from the one in the updated program. This produces warnings and errors that would normally never be seen by the user. By using the same version each time eliminate these problems in the creation of the tests.

Looping

Depending on the types of transactions that you are performing, modifications to the looping of the script can play a big role in defining your workload and having successful runs. Keeping in mind time outs and server response times for each of your transactions will help in creating looping strategies. Listed below are some possible situations that you could encounter and enhancements that you could make.

Home, login, and logout included where low-volume transactions occur

The nature of the application and business transactions that they represent may sometimes be low volume, but still have a significant impact on server performance. Because of this impact, it is often deemed important that these low-volume transactions still be included in the workload.

Session timeouts are one of the first considerations. For example, if we wanted to execute two transactions an hour, each taking 5 minutes, we would have on average 25 minutes of think time to achieve the correct pacing. If server session timeouts occur after 10 minutes (which it did for this case study implementation), it becomes important to log out and back in within the test's business transaction loop. Without logging out, the user would be presented with the message shown in Example 4. This in turn means that there is no active session for future transactions. In the real world, when a user is presented with this panel, he would navigate back to the home page and then log in again. We must do the same within the test.

Example 4 Session timeout

The JD Edwards EnterpriseOne session for this application has timed out. If you were in the middle of processing a transaction that transaction will need to be reentered.

Verify that timeout value is high enough for very long response times

Because of the tremendous amount of back-end process that may occur for some of the business transactions and queries, it is important that RPT can accommodate long response times. Although production-level response times were good, while recording tests against a test environment/partition, on occasion we had transactions that would take many minutes to complete.

To accommodate large test response times, the timeout value was incremented (Figure 22). This can be found by selecting the **HTTP Options** tab of the Test Element Details section. The timeout value was incremented from 240000 milliseconds (4 minutes) to 1800000 milliseconds (30 minutes).

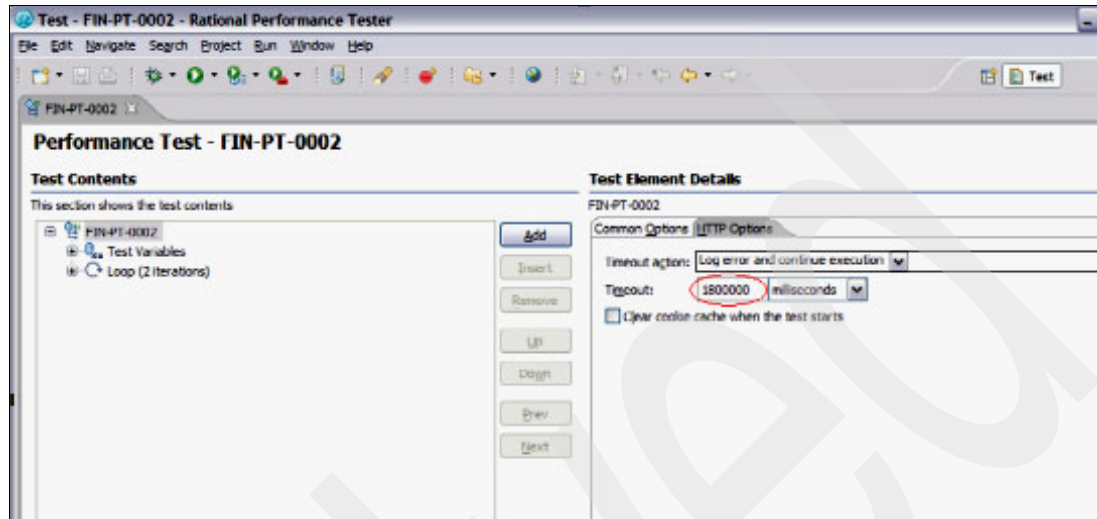


Figure 22 HTTP timeout value

Program versions

One thing to be aware of in Oracle EnterpriseOne is the application versions that you are running and scripting. Since some Oracle EnterpriseOne applications can be customized, you will have to verify which version is being accessed by checking the Fast Path page request (see the circled items in Figure 23).

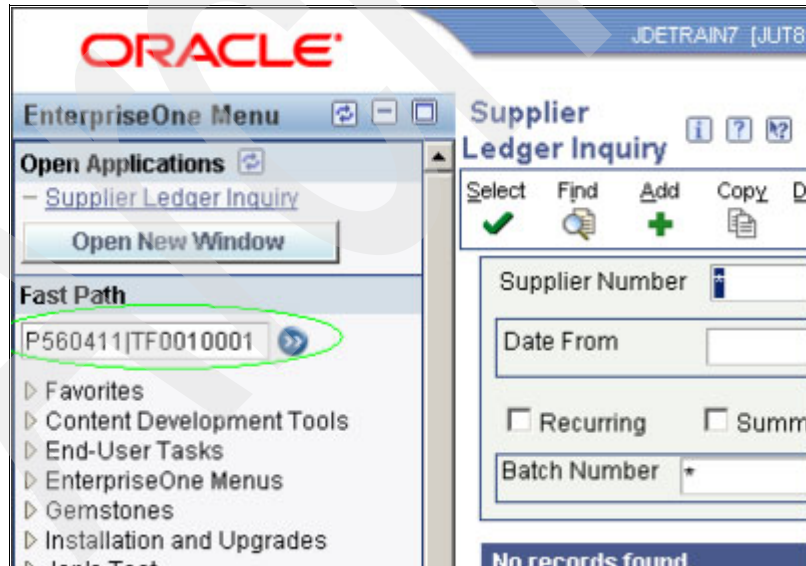


Figure 23 The Fast Path application number and version number all in one request

In some cases, the version does not appear, which usually means that in your recording you went with the default version. The difference in default application versions from customized versions is the panel flow that is presented to the user. To assure ourselves that the correct version is being accessed and to avoid any defaults that could potentially cause problems, we

inputted the version number along with the application number. The Version number is highlighted in the test data. The fast path would be the appIDversion (Figure 24).

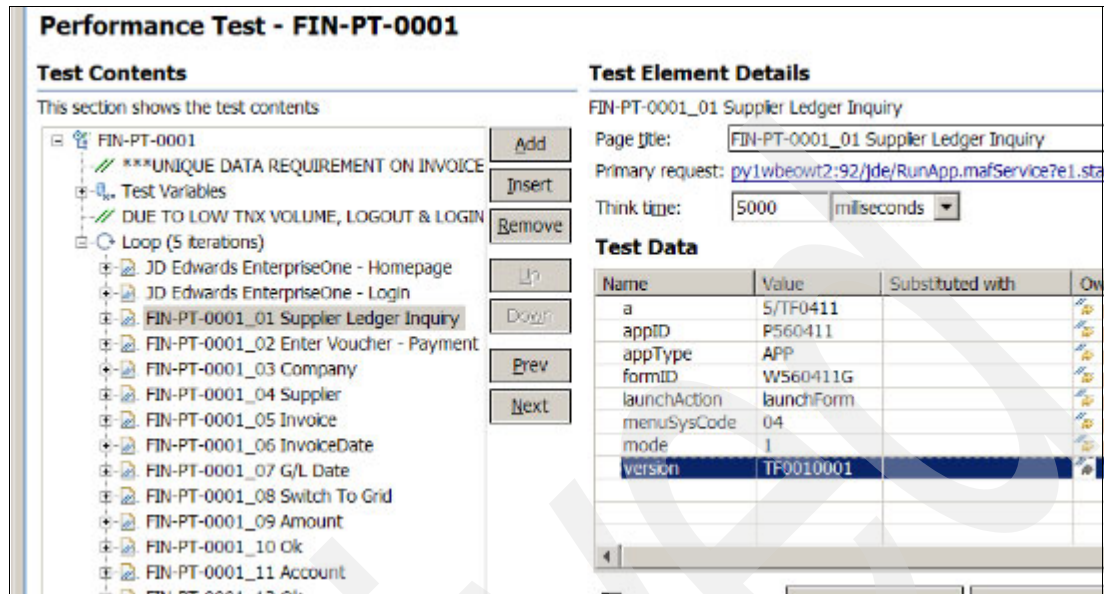


Figure 24 Version number

Enabling verification points (content/response)

To ensure accurate completion of the tests, a number of verification points are available to use. They can be enabled on the entire test or on a specific page. Most commonly used are response code verification points. They verify that status code captured during the recording of the test did not change on the execution. Content verification points are powerful tools to verify whether a specified string does or does not appear in response content (Figure 25).

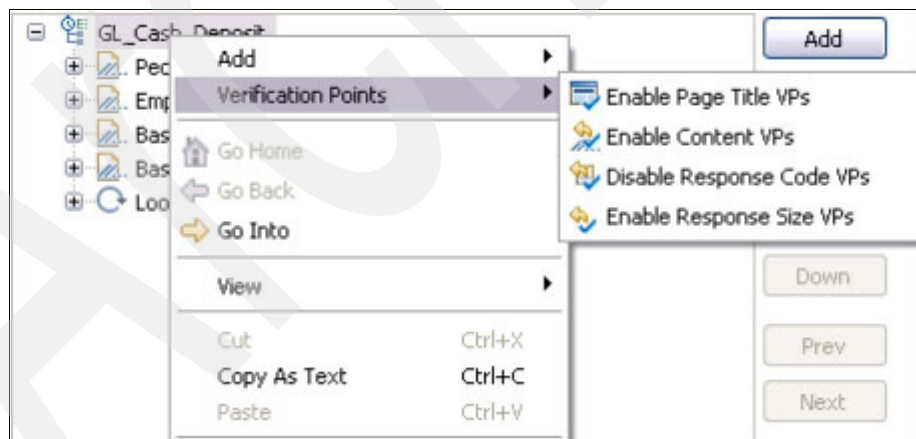


Figure 25 Enabling verification points

Page title VPs

In order for your scripts to flow properly, you will need some verification in your page titles so that the script knows when to perform a given transaction on the expected panel. This is done by simply clicking a test element at the highest level and selecting the check box **Enable verification point** under Page Title Verification Point (Figure 26).

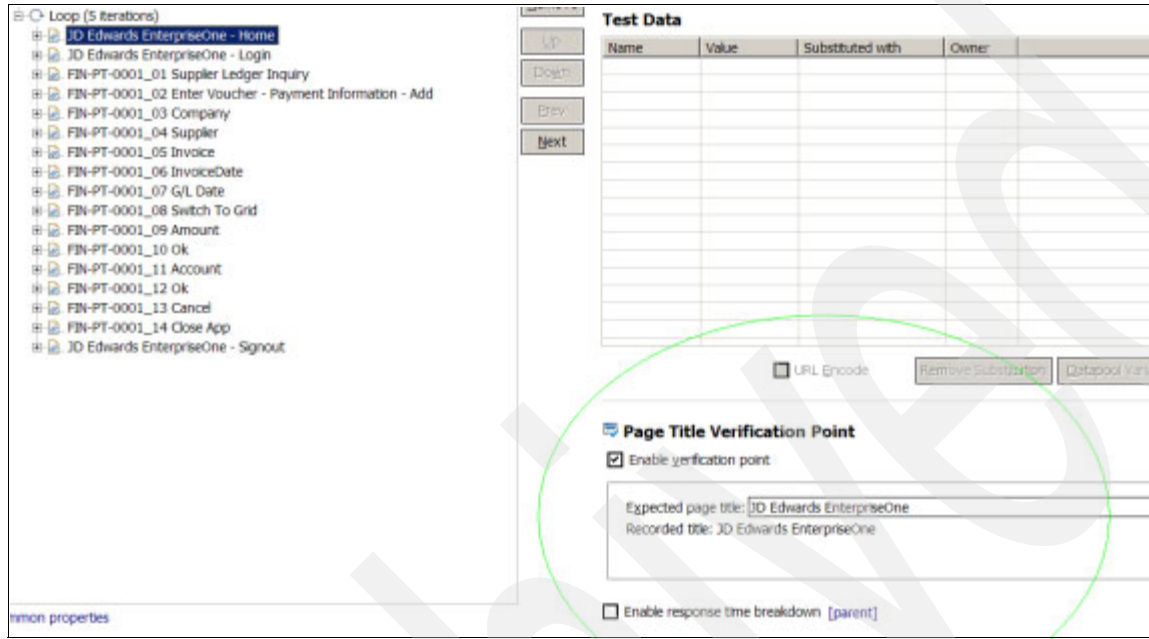


Figure 26 Changing page title verification point

The expected page title will default to whatever the script is that was recognized at recording time and is usually accurate. However, in some cases you will have to customize the page title to fit the particular transaction. If you happen to see a blank expected page title, there is a chance that some transactions were separated and need to be merged.

Playback and debugging performance test scripts - HTTP bad request recorded - results in long response time

In some cases, the tool will record bad response codes that occurred during recording and will replay those bad requests, resulting in undesirable response times. Any request other than 200, 206, 301, 302, or 304 will be an indication that it is a bad request. This will cause your scripts to fail because it is simulating those bad requests. To avoid this, simply delete these bad requests from the test.

Oracle provided a fix for the particular error that was being seen in the response codes, but not until after the recording was complete. In this case, we had to manually delete these bad requests prior to the fix.

Performance schedule creation

Once all tests have been modified accordingly, you are now ready to create and modify the schedule to achieve your desired transaction rates by focusing on looping techniques and think time.

Pacing test iterations

In order to achieve a realistic workload, we need to control the pacing of our tests (or scripts) individually. Different tests (or virtual users) run at different pacing in order to achieve the correct transactions per hour for each. In RPT, this is easily set and controlled through test element details within the script. Select the loop within the script. Navigate to the test element details. Check the box to control the rate of the iterations. Input the transactions per hour that you wish to achieve. In this example, we also want to randomly vary the delay between iterations and to delay before the first iteration. This allows for a realistic workload, with some variability in the transaction arrival times, and still achieves the correct number of transactions per hour (transaction arrival rate). An important note is that we set this pacing value within the test, not within the schedule. See Figure 27.

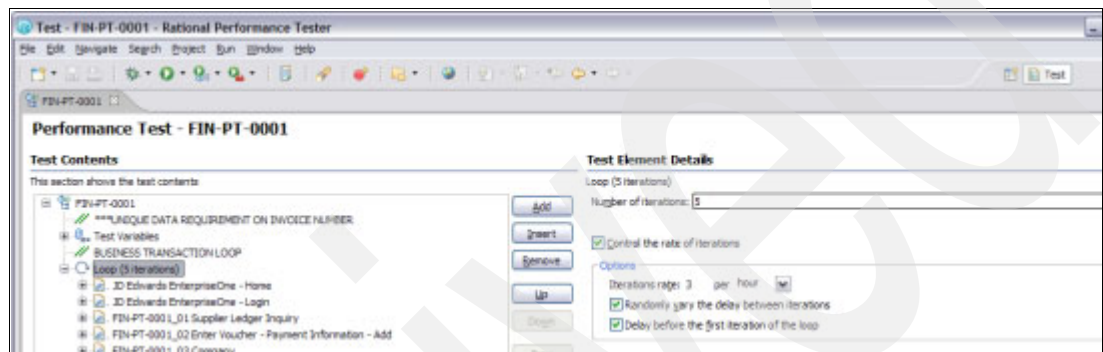


Figure 27 Loop pacing in a script

Ramping up the workload

Often in load testing we come across the necessity to ramp-up the workload over some period of time. This is often necessary due to the heavy load that can be generated by login, authentication, and application startup.

Previously, we saw how to vary the individual tests to achieve our realistic peak workload. Now we want to slow down and spread out the rate at which transactions start entering the system. After this ramp-up period, our peak workload begins to execute.

In Figure 28, we have a schedule created with individual groups created that contain our tests. We place a loop within the group immediately before the test itself. Although the loop shows a warning, that is OK. We do not actually place anything within the loop. We are using it to implement a random delay. In our example here, we want to ramp-up over a 10-minute period. We set the number of iterations to 1. We select to control the rate of iterations. Our iteration rate is 6 transactions per hour (10 minutes), which we randomly vary and we also delay before the first iteration. What we just accomplished is inserting a random delay of approximately 10 minutes before the test begins executing, in essence, allowing an extra 10 minutes for all the virtual users to get logged on and started into their transactions.

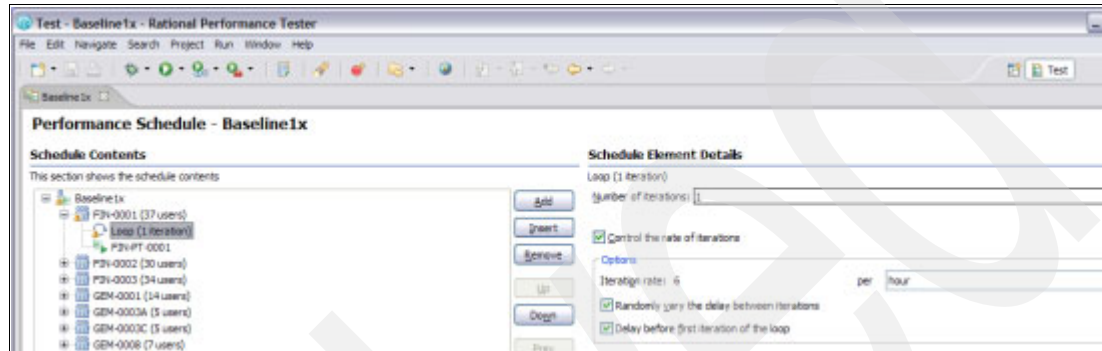


Figure 28 Inserting a loop in the schedule

Page requests and global think time changes

Page requests in your test will also include think times, which account for the amount of time that a user waits to perform the next transaction. For debugging purposes, you would rather have playback occur as fast as possible so that you are not spending unnecessary time waiting. You can achieve this in two ways:

- ▶ In the test itself, you can find each think time field by clicking the page request and editing the value, as shown in Figure 29. The disadvantage is that you have to maintain and edit all think time instances within the test.

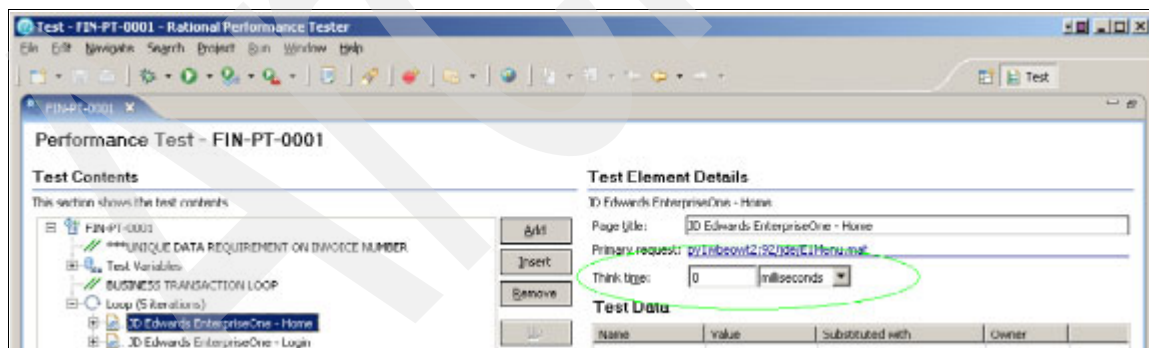


Figure 29 The circled field is where you edit the think time

- ▶ In the schedule, there is a quick way to override all think times that were recorded in your test by highlighting the schedule name and clicking the **Think Time** tab. Here you can select **Specify a Fixed Think Time** from the drop-down list (Figure 30).

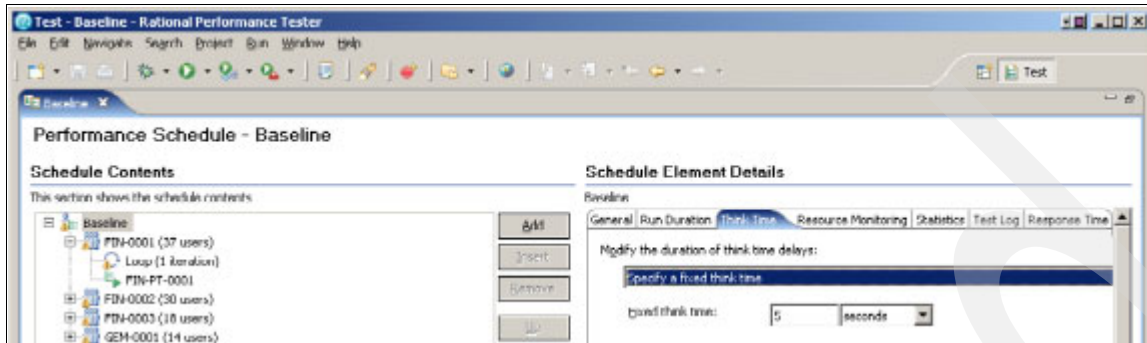


Figure 30 Version number

Execution and analysis

We can view the execution of a performance schedule that applies the load on the JD Edwards EnterpriseOne system by using a number of real-time reports. Let us look at a few of them. The *overall report* (Figure 31) provides the percentage of successful page loads, page element load, and verification points.

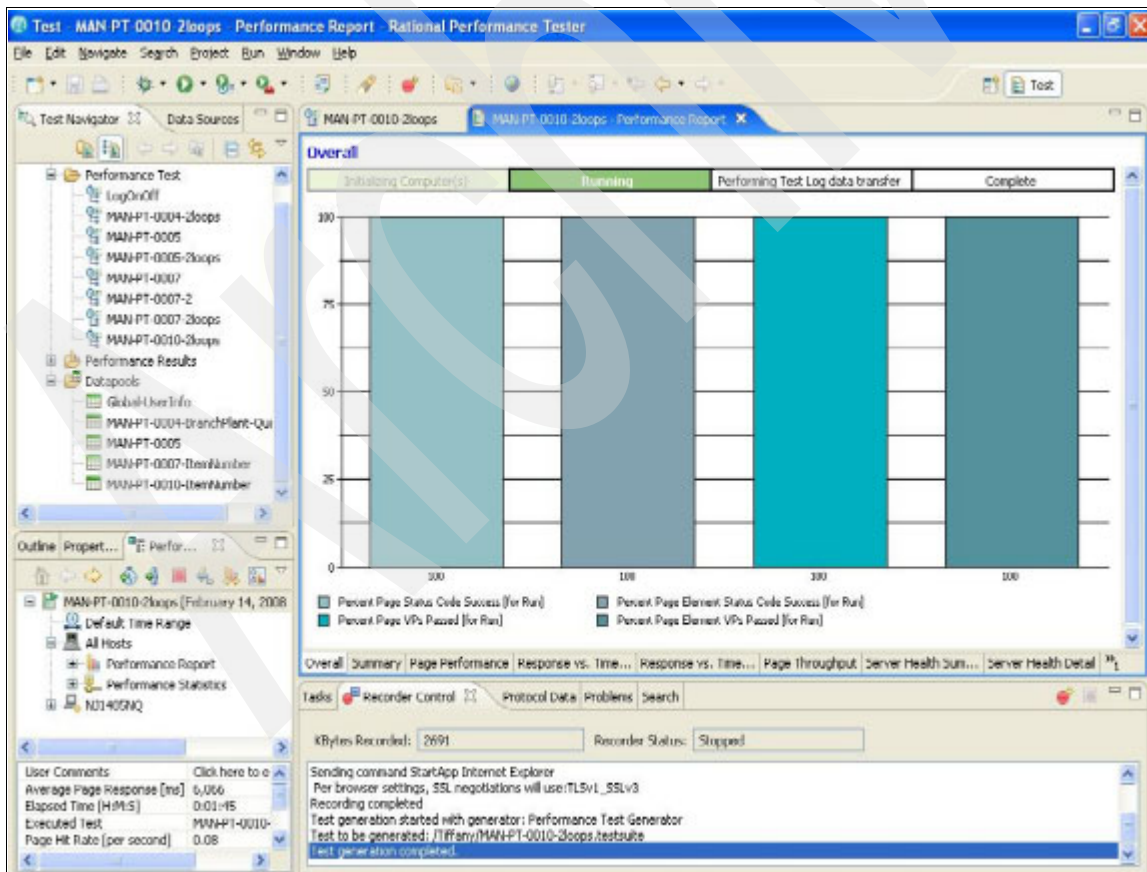


Figure 31 Overall performance report

The *response versus time summary* (Figure 32) indicates how the page response behaves while the system is under load. The steady state behavior in the time range of 1,500 to 3,000 seconds shows page response averaging around 1.5 seconds per page and 0.1 seconds per page element.

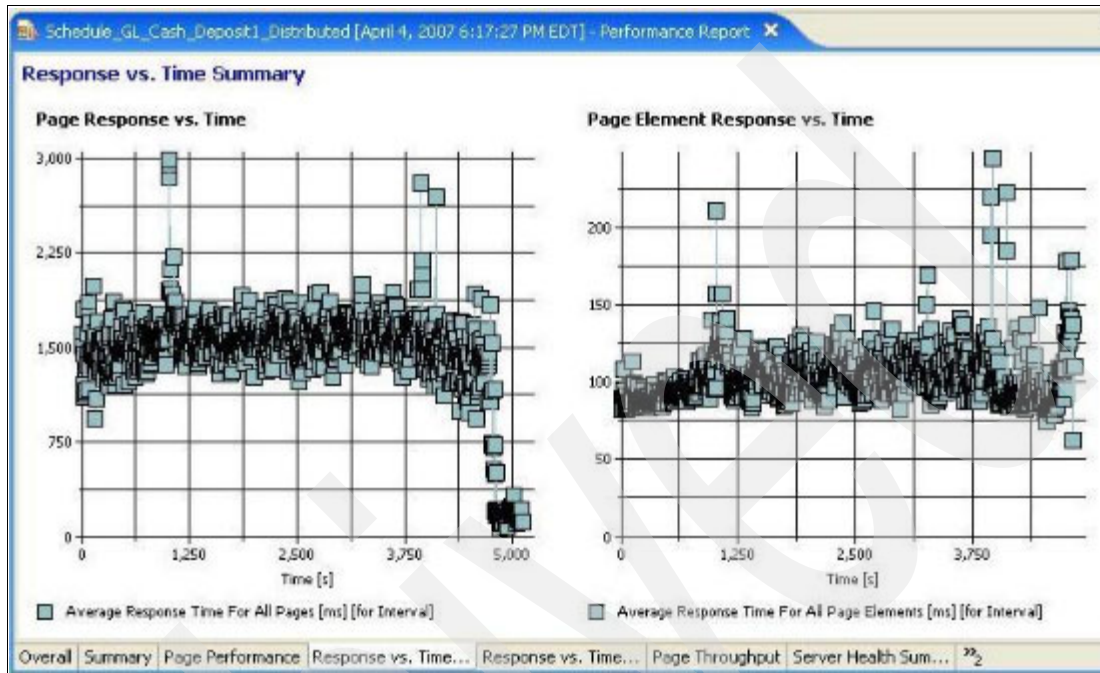


Figure 32 Response versus time summary performance report

The page throughput reports the page hit rate over time (see the left side of the graph in Figure 33) and the user load (see the right side of the graph in Figure 33). In this example, the user load is ramped up to 1,000 users and a steady state load exists from 1,500 seconds to 3,000 seconds of the elapsed time.

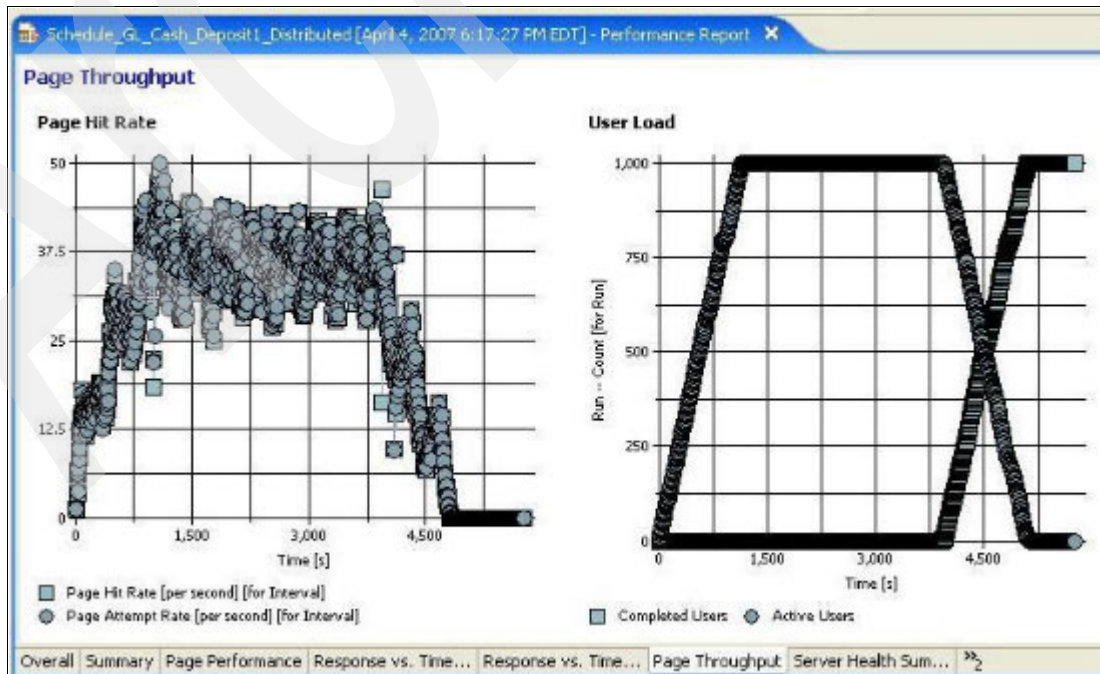


Figure 33 Page throughput performance report

Appendix

This appendix discusses some of the products and resources that were used. Figure 34 is a diagram of the resources and products that we used during our tests.

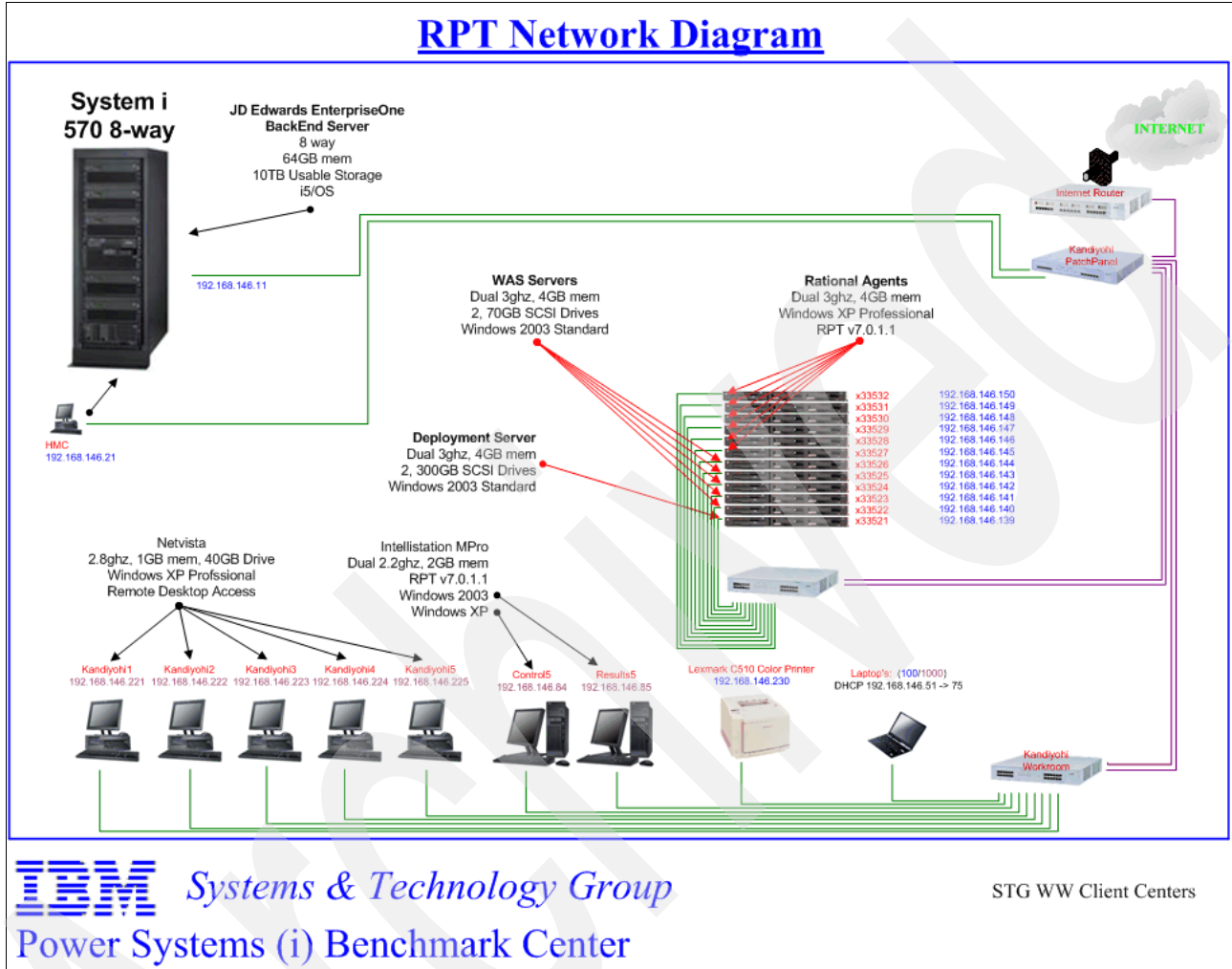


Figure 34 Test environment

IBM Rational Performance Tester

Performance Tester is a multi-user load testing and performance testing tool for validating Web application scalability. It contains these functions:

- ▶ Creates, executes, and analyzes tests to validate the reliability of complex e-business applications
- ▶ Provides no code testing, point and click wizards, report readability, usability, and customization
- ▶ Delivers both high-level and detailed views of tests with a rich, tree-based text editor
- ▶ Performs capacity planning tests to ensure optimal investment in hardware and IT infrastructure
- ▶ Delivers automatic identification and support for dynamic server responses

- ▶ Enables large, multi-user tests with minimal hardware resources
- ▶ Diagnoses the root cause of performance bottlenecks by quickly identifying slow performing lines of code and integrates with Tivoli® composite application management solutions to identify the source of production performance problems

IBM Power Systems (i) Benchmark Center

The Power Systems (i) Benchmark Center in Rochester, Minnesota, provides world-class benchmarking-related skills and facilities to the worldwide and Power Systems (i) community. Our facility is staffed with Power Systems (i) experts with more than 75 years of collective experience in performance and testing methodology. We are able to provide any System i® configured to your exact specification so that you can stress, tune, and test your application, measure performance, and determine workload capacity. The result of your benchmarking experience will provide you with the information needed to make sound business and computing decisions.

Oracle's JD Edwards EnterpriseOne

JD Edwards EnterpriseOne is an integrated applications suite of comprehensive ERP software that combines business value, standards-based technology, and deep industry experience into a business solution with a low total cost of ownership (TCO).

- ▶ Only Oracle's JD Edwards EnterpriseOne offers you a choice of databases, operating systems, and hardware so that you can build and expand your IT solution to meet your business requirements.
- ▶ Only Oracle offers 70 JD Edwards EnterpriseOne applications modules to support a diverse set of business operations.

For more information see:

<http://www.oracle.com/applications/jdedwards-enterprise-one.html>

IBM System i 570

IBM System i 570 is designed for the challenges faced by mid-sized and large enterprises that need a powerful and highly versatile system for enterprise business processes and applications. Based on scalable POWER6™ processor technology and featuring the highly efficient i5/OS® operating environment, the System i 570 platform is ideally suited for the deployment of your most critical applications. It offers exceptional business resiliency, high security, and low operations costs, a combination that can help you focus your IT staff on supporting new business initiatives and growth. The i570 also features advanced virtualization technologies designed to promote high systems utilization and efficient power usage. For more information see:

<http://www-03.ibm.com/systems/i/hardware/570/index.html>

IBM WebSphere Application Server

IBM WebSphere® Application Server V6.1 is the foundation of the IBM WebSphere software platform, and a key building block for a service-architecture (SOA). As the premier Java 2 Enterprise Edition (J2EE™) and Web services application platform, WebSphere Application Server V6.1 delivers a high-performance transaction engine that can help you build, run, integrate, and manage dynamic, on demand business applications.

As the core configuration of the WebSphere Application Server family, WebSphere Application Server is optimized to ease administration in a scalable, single-server deployment environment. This configuration is recommended for organizations that need to build and deploy stand-alone, departmental applications and Web services, but do not require failure bypass or workload-distribution options. WebSphere Application Server supports an unparalleled range of platforms and can be deployed on servers of any size. New and enhanced features deliver the flexible, open, resilient application infrastructure that you need for an SOA. For more information see:

<http://www-306.ibm.com/software/webservers/appserv/was/features/>

The team that wrote this IBM Redpaper

This paper was produced by a team of specialists from around the world working with the International Technical Support Organization, Poughkeepsie Center.

Michael Meyers is an Advisory Software Engineer with 12 years of experience in the Power Systems (i) Benchmark Center located in Rochester, Minnesota. He has been involved in numerous benchmark projects with customers from around the globe. His previous experience includes a variety of testing positions, technical support, AS/400® CISC to RISC Migration, and software development.

Chris Beyers is a Software Engineer with two years of experience in the Power Systems (i) Benchmark Center located in Rochester, Minnesota. He has worked with customers on creating interactive solutions with both Rational Performance Tester and HP Loadrunner. His areas of expertise include performance testing and workload simulation across a wide range of client-side protocols.

Don Weber is a Senior IT Specialist with IBM Software Services - Rational, where he supports testing automation tools. Don has 25 years of experience as an Application Developer and Test Lead. His areas of expertise are software processes, software quality, and security. His previous experience includes working as a CAD/CAM software engineer, a process engineer, and an administration control expert.

Thanks to the following people for their contributions to this project:

Mike Ebbers
International Technical Support Organization, Poughkeepsie Center

David Chadwick
IBM Software Group - Rational, Raleigh, North Carolina

Dan Daley
STG Power Systems (i) Benchmark Center, Rochester, Minnesota

John Mullin
STG WW Mainframe Benchmark Center, Poughkeepsie, New York

Helen Olson-Williams
STG Power Systems (i) Benchmark Center, Rochester, Minnesota

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4456-00 was created or updated on November 18, 2008.



Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at: ibm.com/redbooks
- ▶ Send your comments in an email to: redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400 U.S.A.



Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

J2EE, Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

AS/400®


i5/OS®

IBM®

Power Systems™

POWER6™

Rational®

Redbooks (logo) ®

System i®

Tivoli®

WebSphere®

Other company, product, or service names may be trademarks or service marks of others.