An Oracle White Paper
May 2010

Load Testing Hyperion Applications Using Oracle Load Testing 9.1

Introduction

This document is intended as a guide for creating load/performance test scripts against Hyperion Financial Applications (HFM) using **Oracle Load Testing.** This guide will assist during the script creation process and enable the tester to create scripts faster and more reliably. It assumes that the person using this document has experience working with **Oracle Application Testing Suite**. This document does not necessarily cover all Hyperion transactions that have to be tested, nor does it guarantee that the parameters mentioned in this document will perfectly match your particular environment.

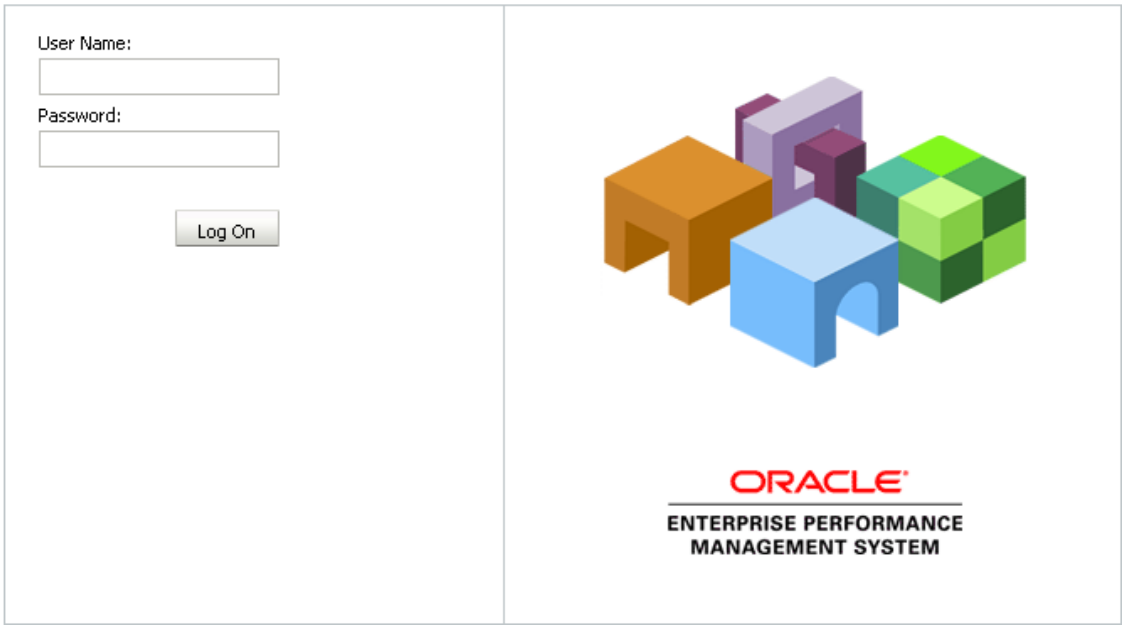

Figure 1: Sample Hyperion Login Page

Script creation

Load test scripts for Hyperion applications are created using Oracle Application Testing Suite's OpenScript scripting platform. Application Testing Suite consists of three main products:

- Oracle Functional Testing – OpenScript for automated functional testing and load test scripting
- Oracle Load Testing – for automated load testing
- Oracle Test Manager – for test process management

Dynamic parameters

One of the major challenges of performance testing Hyperion is the highly dynamic nature of the Hyperion applications. Hyperion applications make extensive use of dynamic http session parameters to track user's session state. This also makes scripting more difficult.

The key to successful scripting is finding out the parameters that need to be parameterized in order to create working load test scripts. Additionally, the script should also work 1 hour, 1 day or 1 week after the recording of the script. Scripts must be parameterized correctly to manage sessions, time stamps etc. If a script is not parameterized correctly, it may work for only a short period of time or may not cope with multiple parallel transactions across concurrent virtual users.

This is a list of some of the key dynamic HTTP session parameters for Hyperion applications:

- sso_user
- sso_password
- sso_token
- repository_token
- ssnkeystate
- Entety
- subcubes
- MODVAL
- FormPOV

Recording & Playback Configuration

OpenScript records load test scripts using an HTTP-based proxy recorder, which is configured automatically by OpenScript when scripts are recorded. If the target system (e.g. the Hyperion application) is only accessible by using a proxy server configured in your browser, the OpenScript

proxy recorder will be automatically chained. If your proxy server requires authentication, this authentication information can be entered into the OpenScript recording proxy preferences.

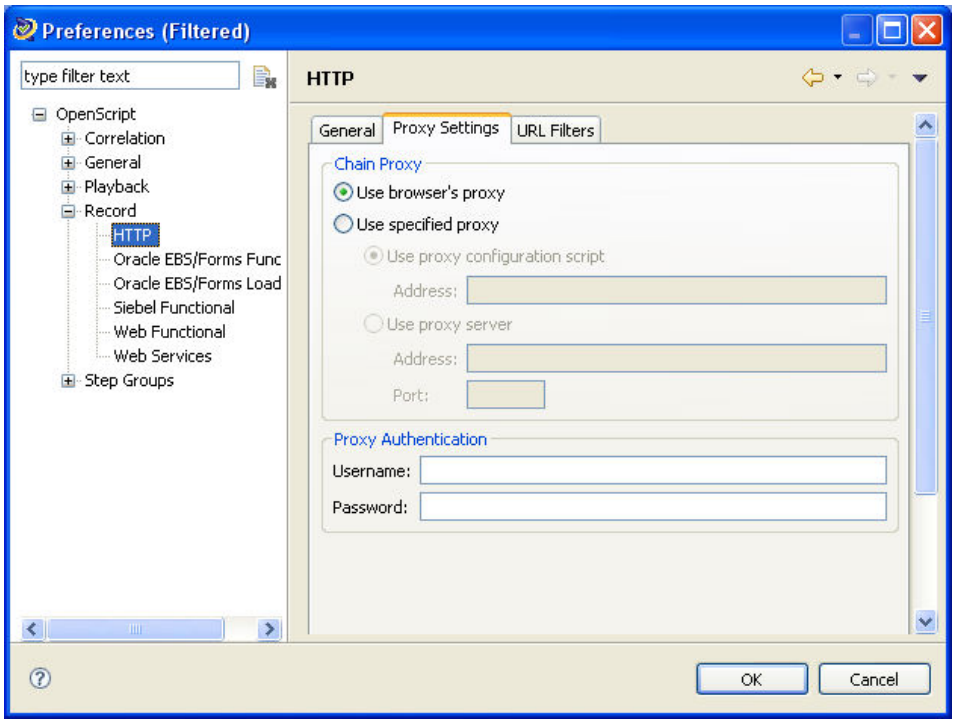

Figure 2: OpenScript recording proxy settings

If the target system (e.g. the Hyperion application) is only accessible by using a proxy server, it is also important that this proxy server is added to the OpenScript playback settings.

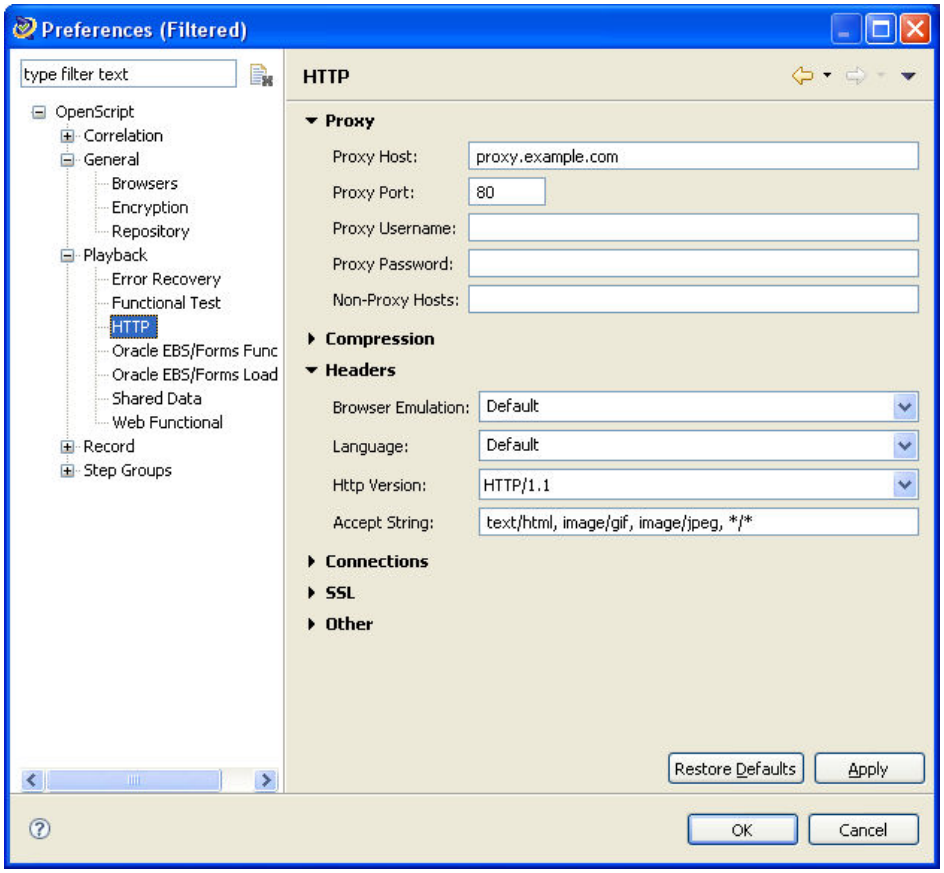

Figure 3: OpenScript playback proxy settings

Note: By default, OpenScript will use the browser's configured proxy during recording, so this may not require additional configuration.

Note: The starting point for all scripts in this document was the URL: http://myApplication.myCompany.com/workspace/

Script example: Login to Hyperion

One of the first scripts recorded will likely be a login script, where the user logs into a Hyperion application. This is an important script, since it contains the login that you will need for every scenario. It is also a good script to start with from a complexity point of view.

The following script was created by creating a new script in OpenScript (File – New), creating a new Web/HTTP load testing script, then clicking record and launching, logging in, then logging out of a Hyperion application.

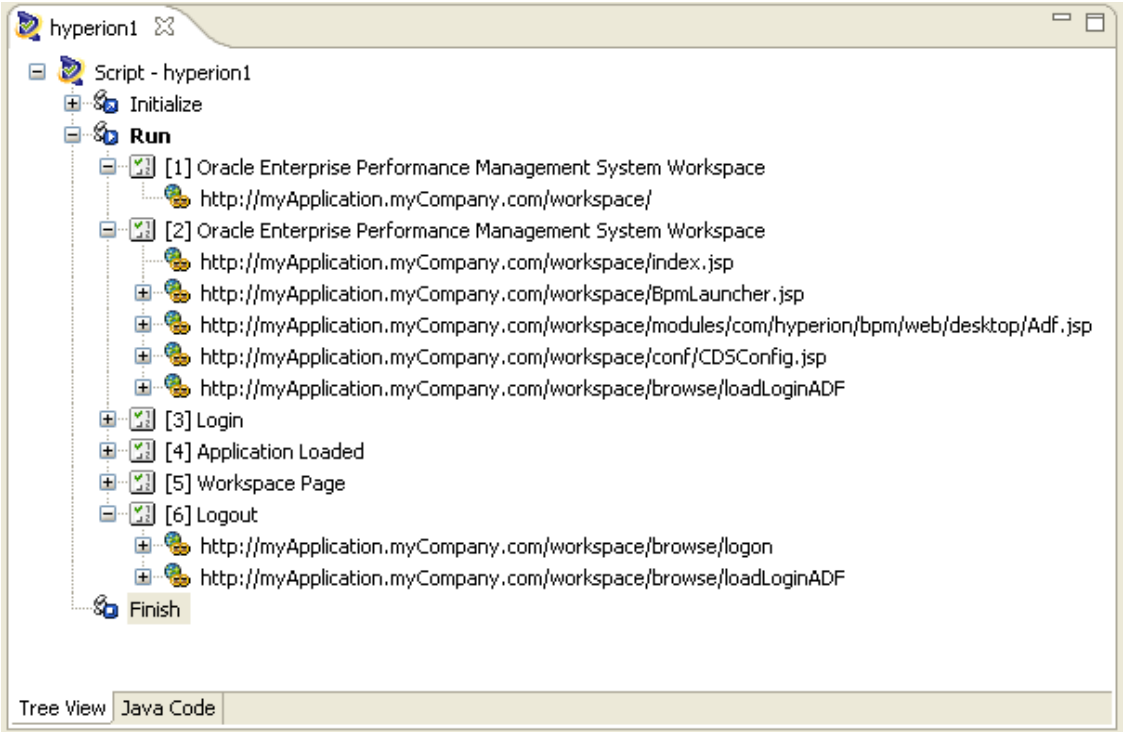

Figure 4: Example of Recorded script that does a login, application selection and finally a logout. Steps renamed to reflect step functions.

Once finished, save the script and expand the script nodes to view the HTTP requests captured. This represents the base script we will be working with. The following steps will demonstrate how to identify and work with the dynamic parameters.

Parameterizing scripts

The following describes the HTTP parameters in more detail and how to handle these in your scripts.

SSO_USERNAME / SSO_PASSWORD

Two parameters that you may want to substitute are the user credentials **sso_username** and **sso_password**. These parameters can be substituted using Databanking (e.g. data for these comes from a CSV/text based file). Databanking allows you to drive different inputs for these parameters using an external data file.

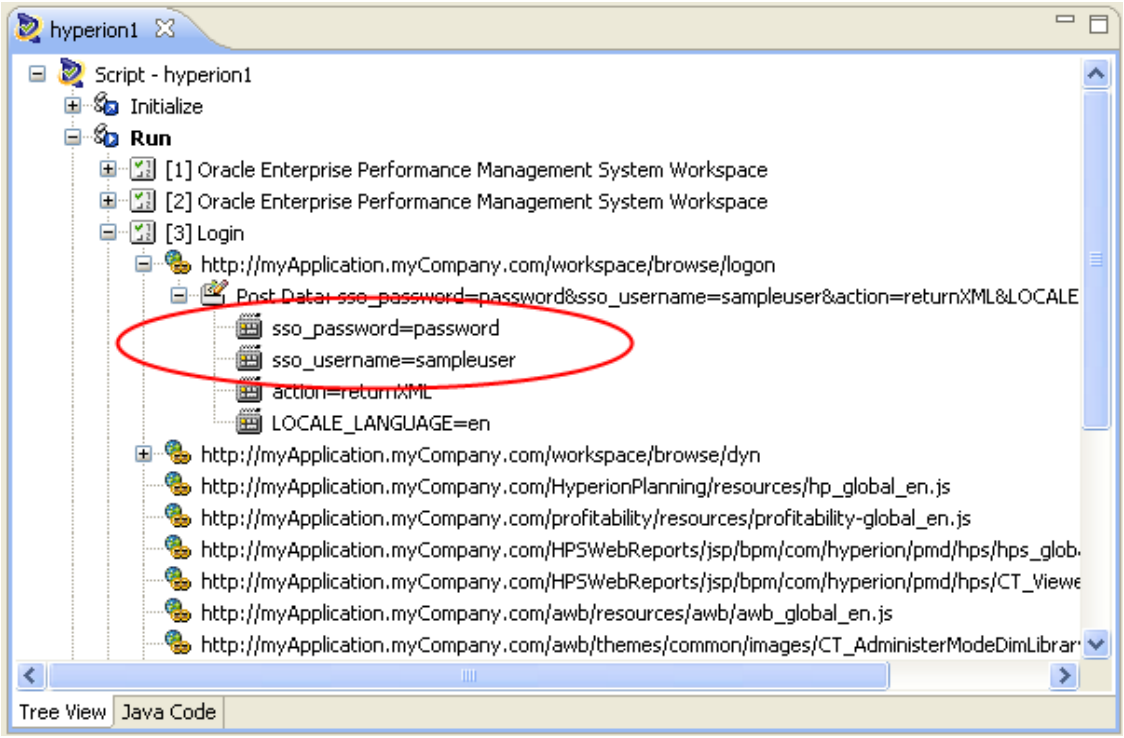

Figure 5: sso_username and sso_password parameters in the navigation PostData

SSO_TOKEN

The navigation after the login POST contains another very important value: the **sso_token**.
The **sso_token** value is a session parameter automatically generated by the Hyperion application after you login.

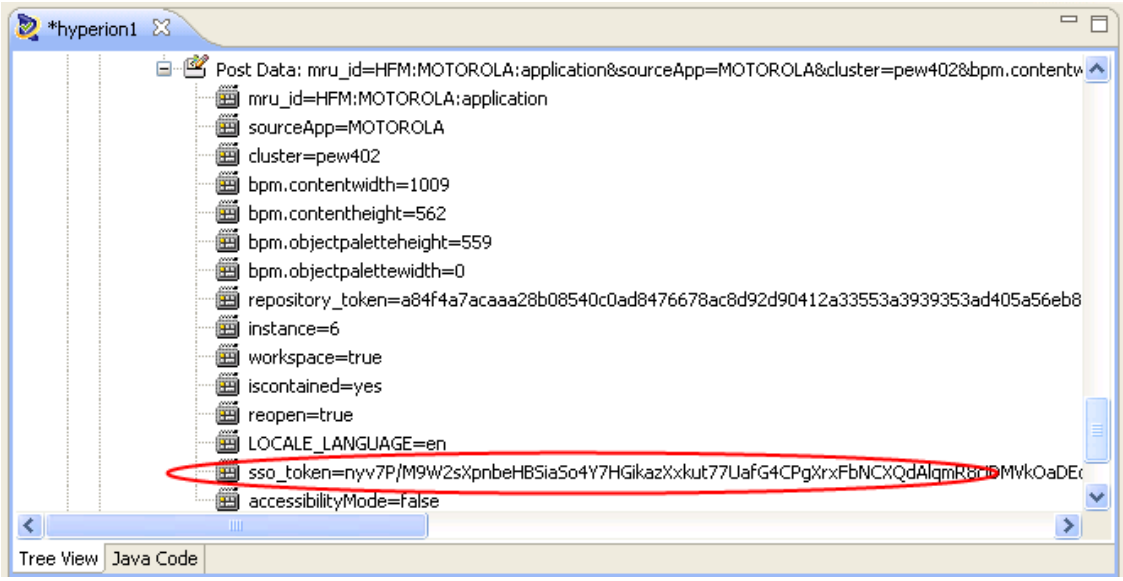

Figure 6: sso_token parameters

In order for your script to work, you need to parameterize the "sso_token" value. To do this, we need to create a variable to extract a new value when the script plays back.

In the login post navigation, right-click and select "Create Custom Variable…"

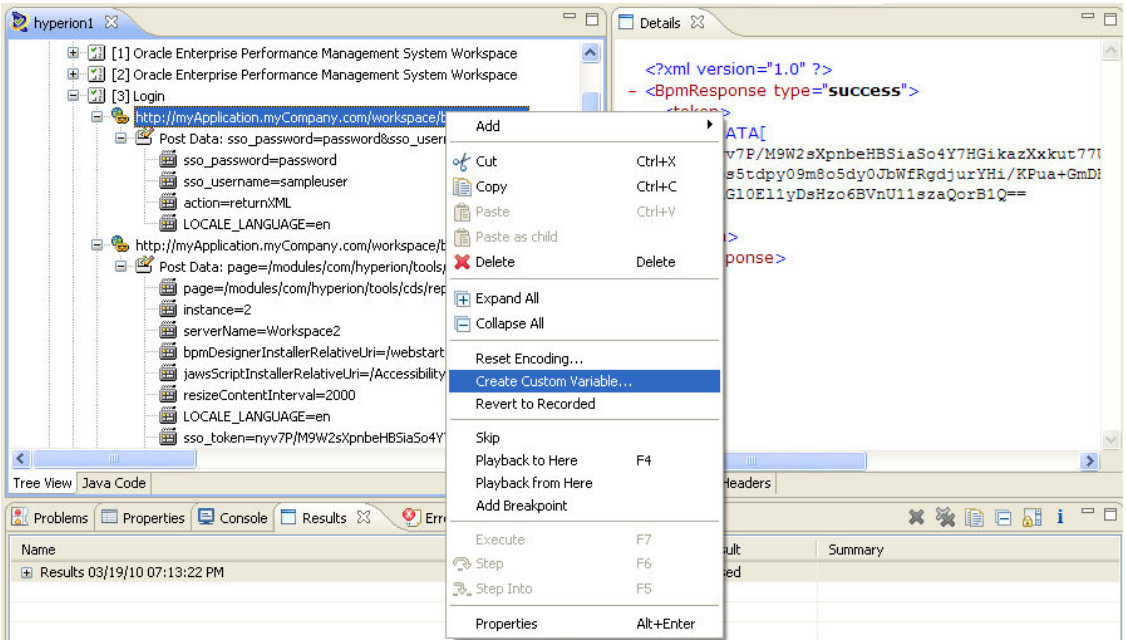

Figure 7: Create Custom Variable – sso_token – Screen 1

In the Create Custom Variable dialog, enter the following into the Regular Expression field and click "Next":

multiline<token><!\[CDATA\[(.+?)]]></token>

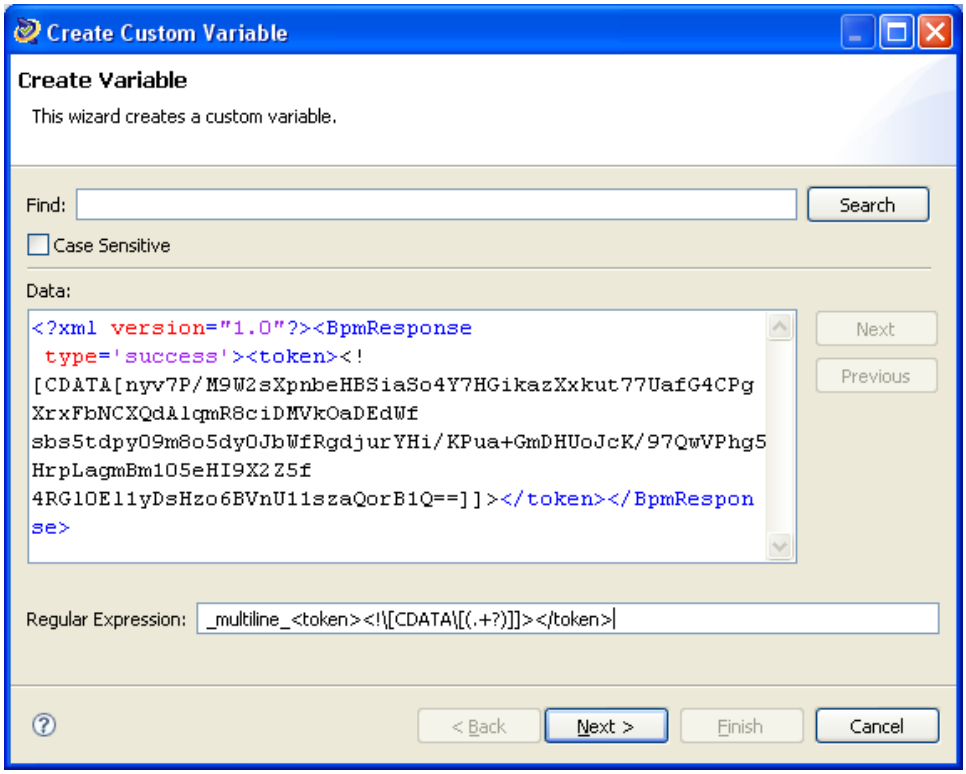

Figure 8: Create Custom Variable – sso_token – Screen 2

In the next dialog, ensure that the correct data is selected and click "Next."

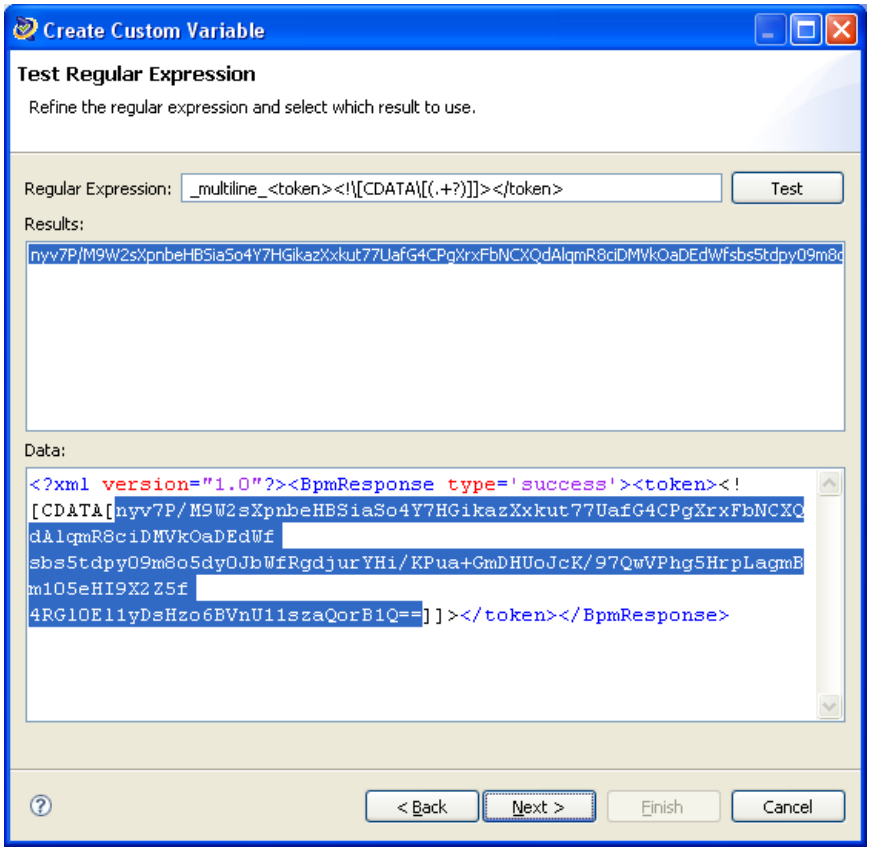

Figure 9: Create Custom Variable – sso_token – Screen 3

In the next dialog, assign a variable name ("ssotoken") and set the Encode Option to "Remove Newlines."

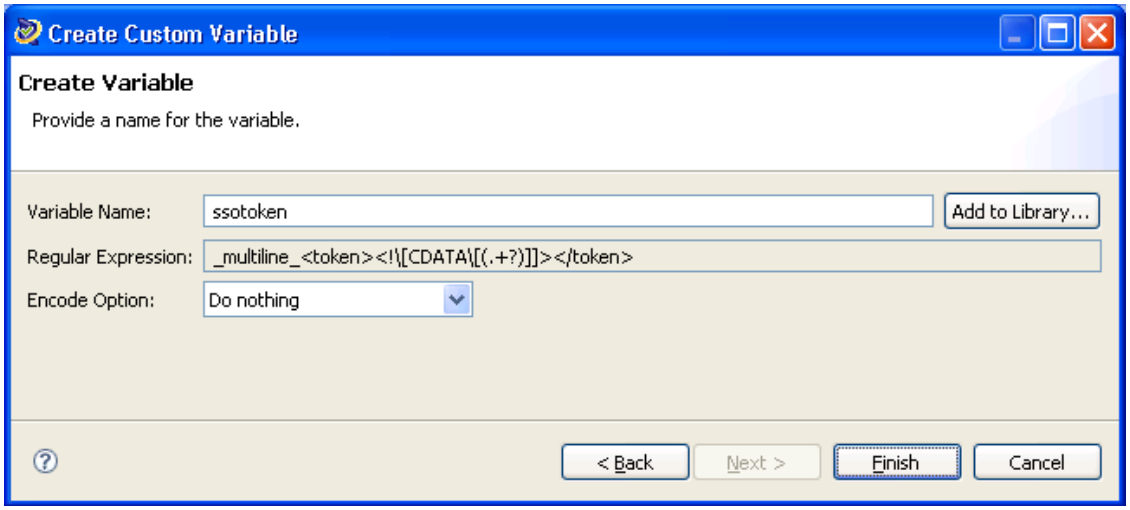

Figure 10: Create Custom Variable – sso_token – Screen 4

Before clicking finish, click "Add to Library…" to add this correlation rule to a correlation library to save for future script usage. Correlation libraries in OpenScript contain correlation rules that can be applied to all new scripts during or after recording to automatically parameterize dynamic HTTP session variables. The sso_token should be added to a Library, since it will be used for all scripts.

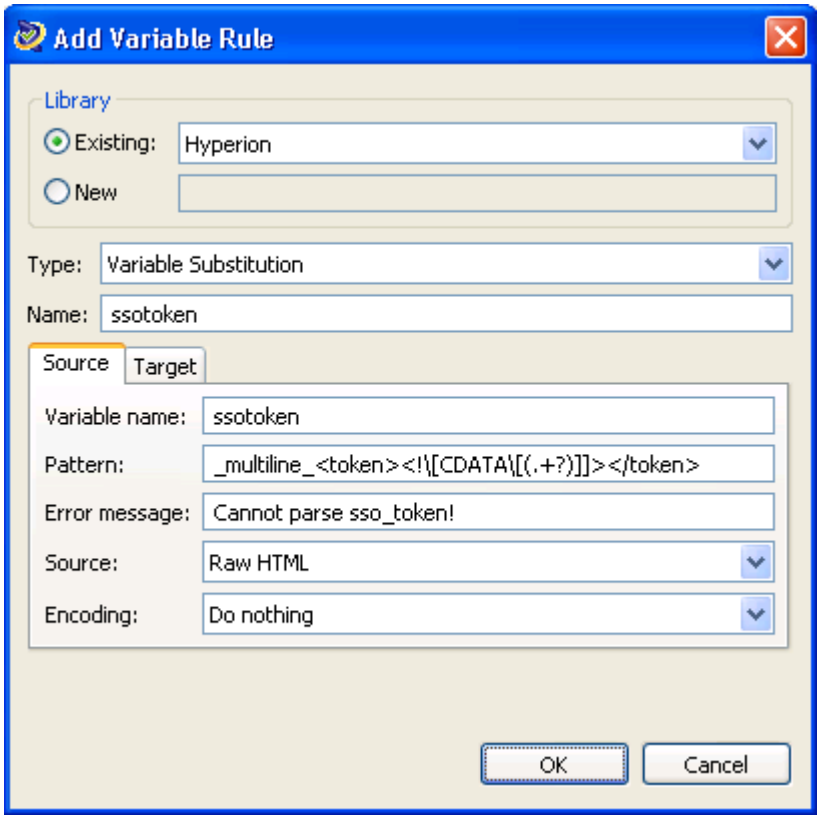

Figure 11: Create Custom Variable – sso_token – Screen 5

Back in the Create Variable Wizard, click "Finish." We now have a regular expression that will extract a new value for sso_token and set it to a variable.

Now, right-click the target sso_token name/value pair in the POST data and select "Substitute Variable…" to substitute this variable into your script.

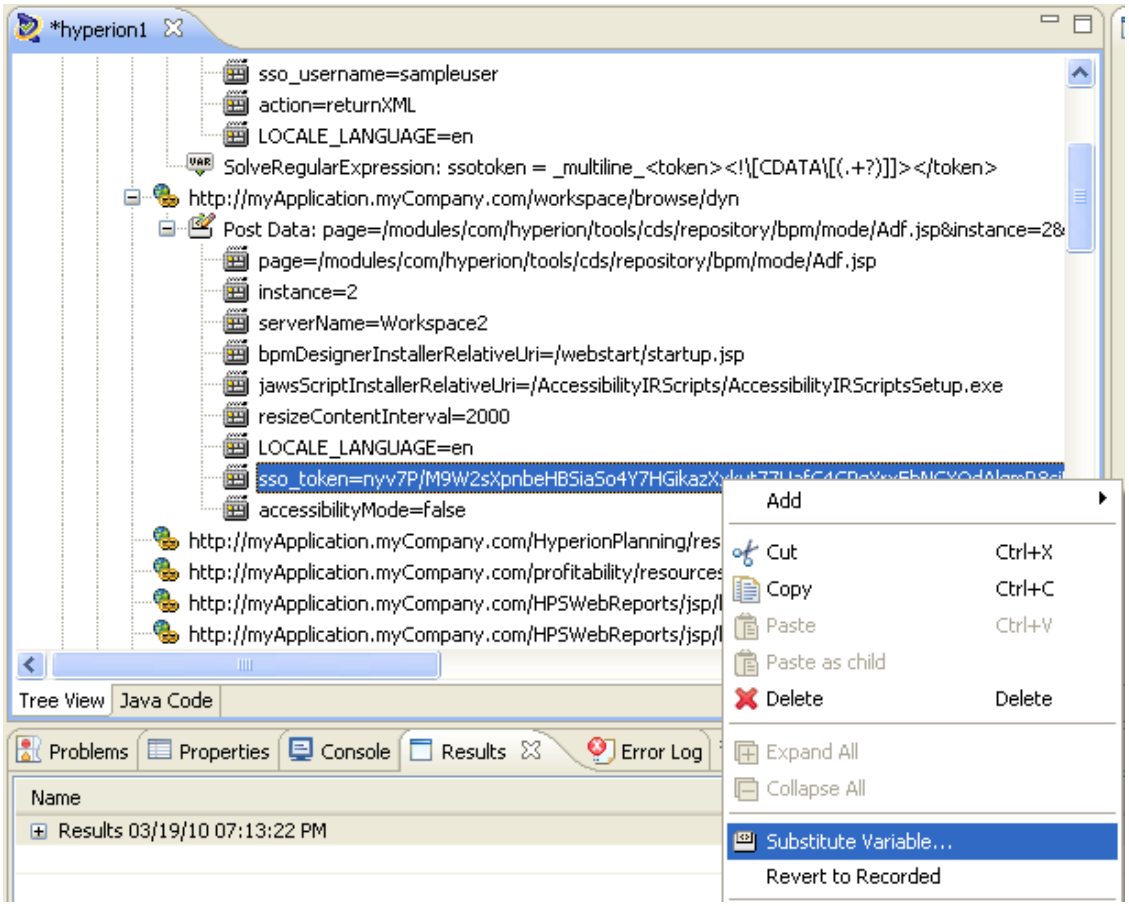

Figure 12: Substitute Variable

From the dialog, select the "ssotoken" variable created earlier.

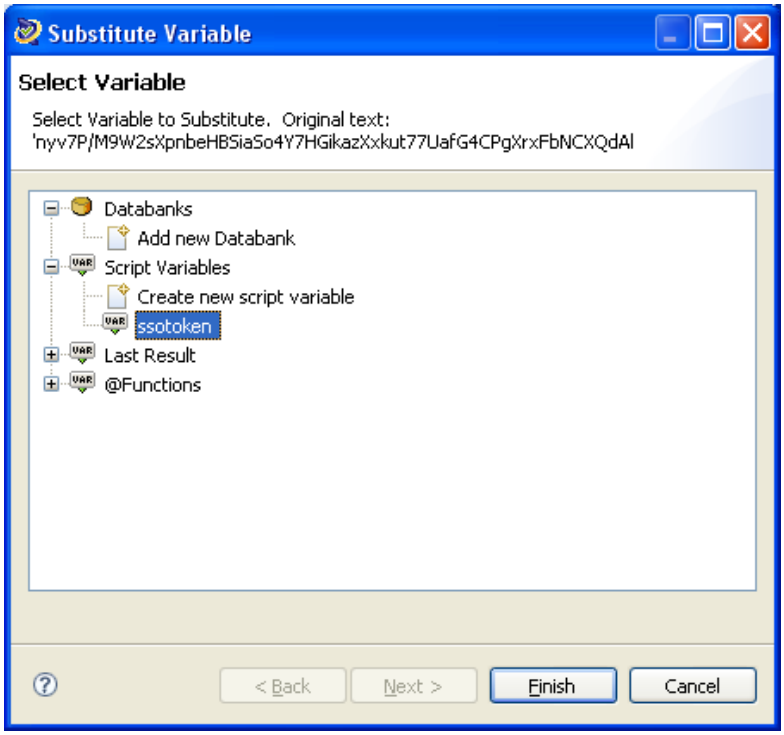

Figure 13: Select "ssotoken" variable.

Finally, verify that the value of sso_token has been substituted by a parameter.

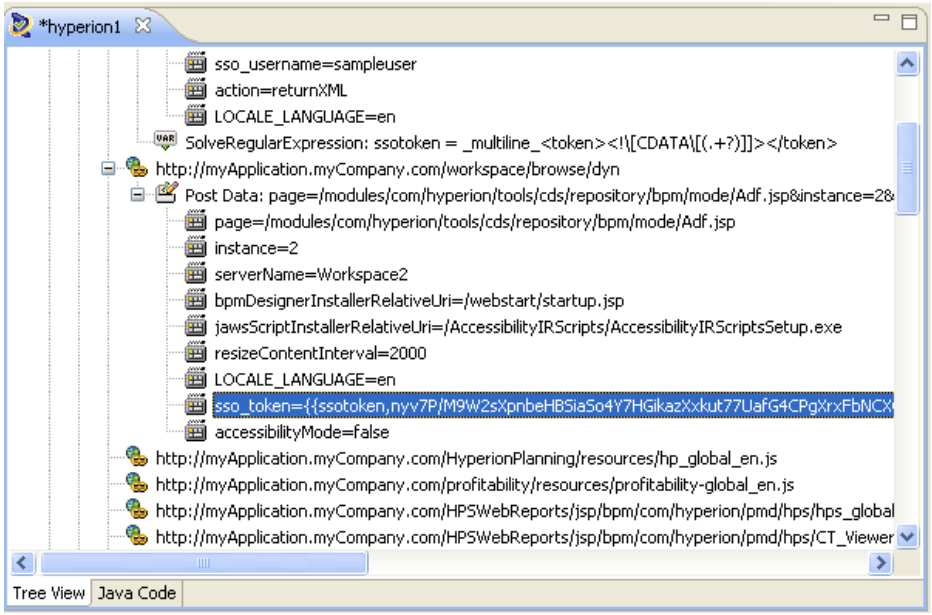

Figure 14: Ensure the sso_token is substituted

If scripts are not played back correctly and error messages indicate that there is a problem with sessions, check the requests and try to verify the sso token.

Apply Pattern Library

At this point you can apply the correlation library to the entire script.

You can correlate the entire script through the "Script – Correlate Script…" menu option.

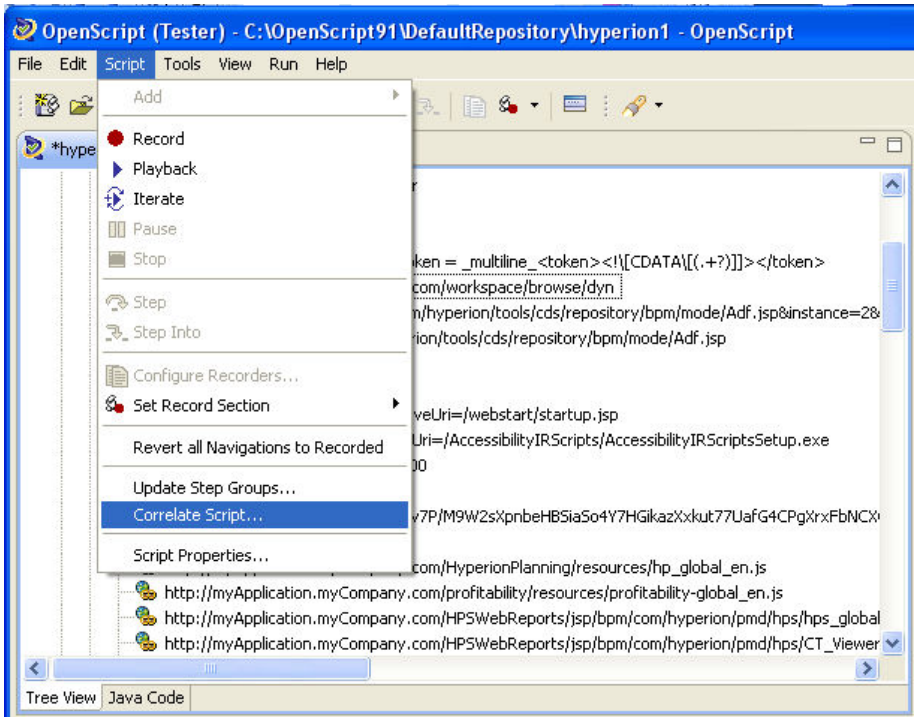

Figure 15: Correlate Script

After this operation the login script should contain variable substitutions for all dynamic sso_token parameters required to run.

REPOSITORY_TOKEN

The second most common parameter to parameterize from the start is repository_token. To correlate, locate repository_token in the script, right-click and select "Substitute Variable…"

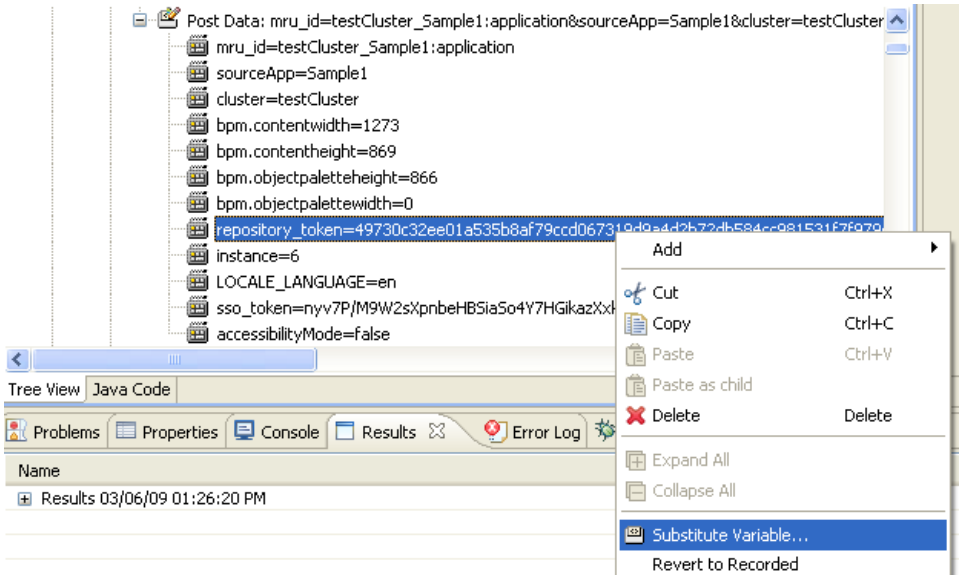

Figure 16: Correlate repository_token

In the Substitute Variable wizard, select "Create new script variable."

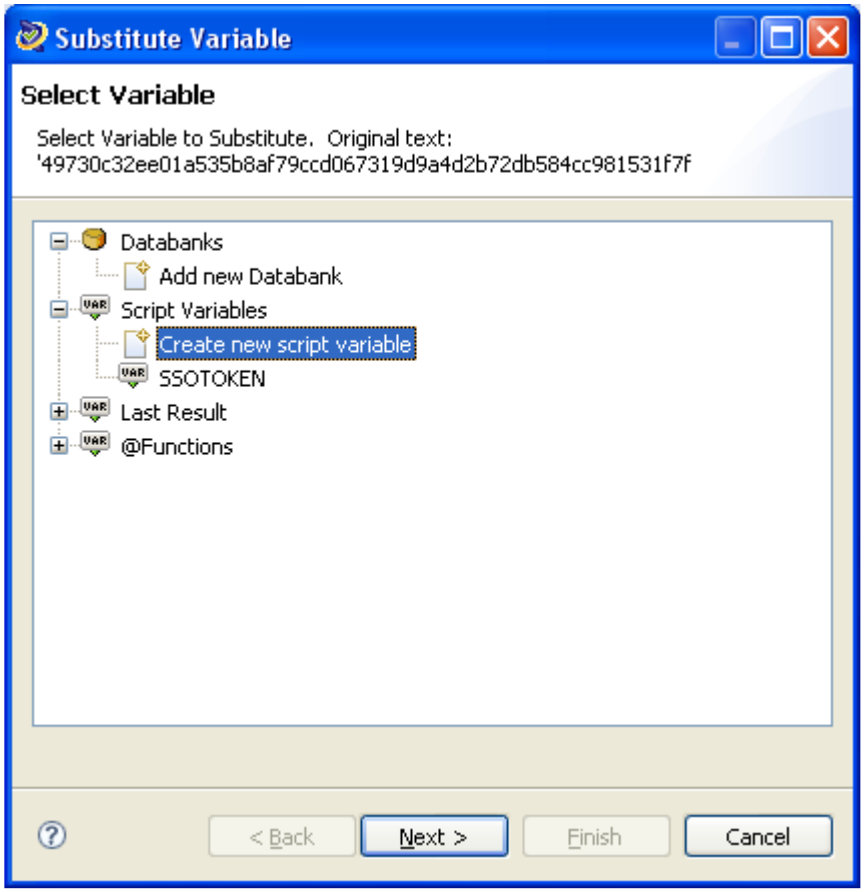

Figure 17: Create new script variable

Verify the contents found, and click "Next."

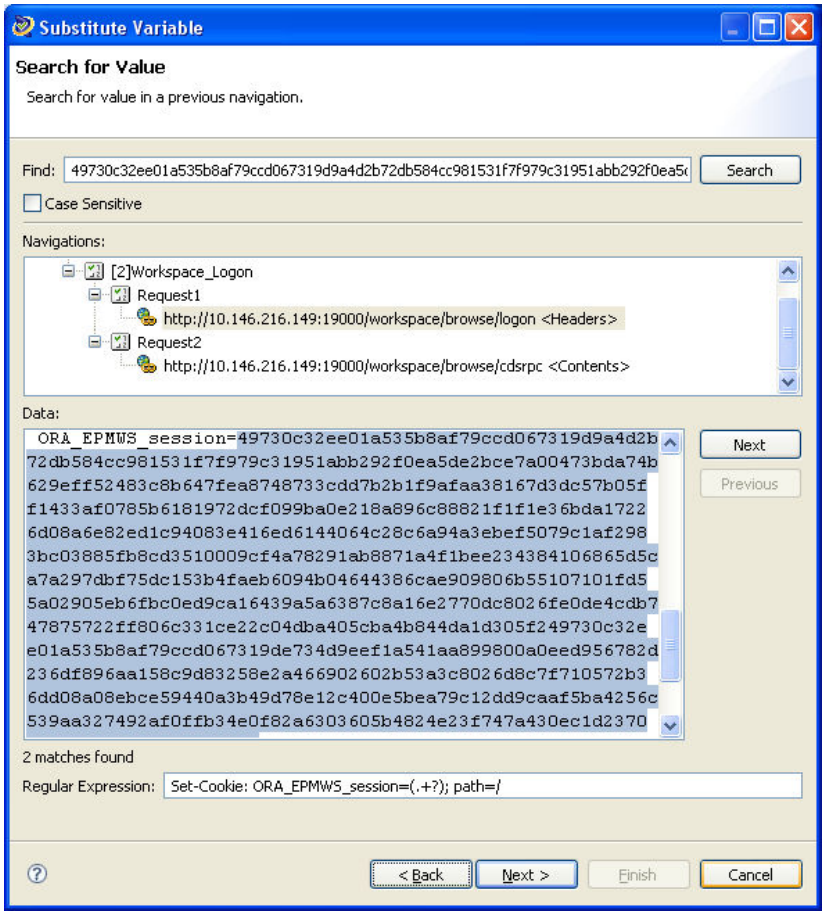

Figure 18: Verify variable source

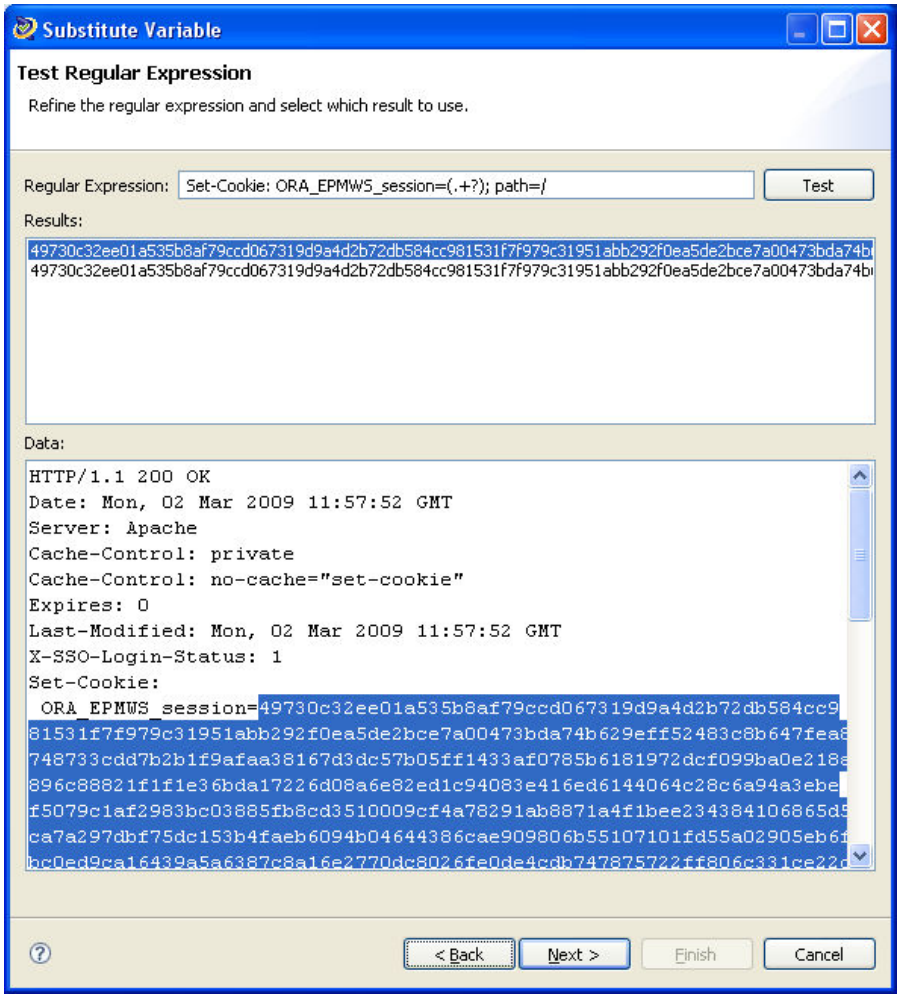

Figure 19: Verify regular expression

Then, as with sso_token, assign the variable a name and add to your correlation library before clicking "Finish."

Other scripts and other parameters

The most important parameters to parameterize from the start are sso_token and repository_token. However, a couple of other parameters may need to be managed as well. You would correlate these following similar steps as described above.

ssnkeystate

The **ssnkeystate** is a value that will change during the scripts pages. It is therefore important not to replace all ssnkeystates with ONE custom dynamic value, since this value will change a number of times in your script (e.g., a 27-page script which included entering data, saving data, and performing calculations resulted in a **ssnkeystate** which changed five times).

Subcubes, MODVAL, FormPOV

Subcubes, **MODVAL**, **FormPOV** are other examples of values that may have to be parameterized if the script is entering data into the system, performing calculations etc.

Additional Values For Other Hyperion Applications

The values that have been covered in this white paper are ones you may encounter using Hyperion Financial Management. However, when testing other Hyperion applications, such as Hyperon Planning or Financial Reporting, you may encounter additional values requiring correlation. A few of these are listed below, as examples of things you may want to look for.

- CI (Composite Grid Identity)
- GI_n (Grid Identity)
- InstanceID (Random value)
- Etc.

Load Testing

Oracle Load Testing lets you run realistic load tests for Web, SOA and Oracle packaged applications – helping you simulate thousands of concurrent users and analyze the impact of production load levels on application performance. Oracle Load Testing lets you simulate thousands of virtual users accessing the Hyperion application simultaneously to measure the effect of user load on application performance. During performance testing, Oracle Load Testing measures end-user response times as well as the performance of the underlying application infrastructure to help you analyze performance and identify bottlenecks.

Figure 20: Oracle Load Testing

Tips and Tricks

The following are some basic tips and tricks for configuring Oracle Load Testing for Hyperion testing.

Before Testing

Prior to beginning the load test, review the following tips and ensure that your settings are properly configured.

Zero length HTML

Deactivate the "Zero length HTML pages are fatal" setting in both the load testing tool and the Navigation Editor. Hyperion will return HTML pages without any payload and this is normal.

Page load times during recording

Always wait for pages to be fully loaded before performing the next action in the script.

(Sometimes page loads can take a long time during the recording). Make sure that a new page occurs in your "visual script page" after every action you perform in Hyperion.

Think times

If scripts are recorded with the external proxy recorder, remember to add think times to each step in the script or define fixed think times in Oracle Load Testing when performing tests.

Test data/users

Reports users can only log in once, one session at a time.
The same user ID cannot be logged in more than once at any one time

(To avoid this make sure that there are enough test data/users available)

Parameter substitution

Make sure the values for the parsed parameters look correct/reasonable

Choose battles wisely

When creating reports in HFM, reports may be stored in a tree hierarchy. For testing purposes copy the reports to the root folder. This will make scripting easier and have a minimal impact on the results. The reason for this is that when the tree is "navigated" the proxy recorder will not pick up the requests if the "normal" recording filters are applied. The navigations in the tree are XML based JSON requests. The alternative solution is to adapt the filters (or to turn the completely off) for the proxy recorder.

During Testing

During the load test, monitor the running test for the following:

Responses

Check the size of the response data after processing queries. If only a few hundred bytes are returned for a large result set, it is a sign that an error message was returned instead of data.

For each transaction, ensure there are no error messages in the response data.

Look for exceptions and stack traces in the data returned from the server.

Receiving lots of binary data from the server is a *good* thing. Error messages come back in clear text.

Session error messages normally indicate that the wrong pages (e.g. wrong sessions are being used), check if sso_tokens are encoded. If not encode and playback script again.

Application logs

Ensure there are no errors in the application logs and review any warnings that may appear.

Performance Diagnostics

When running a load test, Oracle Load Testing also allows you to monitor key performance diagnostics for Hyperion application infrastructure using ServerStats.

System Metrics:

- CPU utilization (% Processor Time_Total)
- Memory usage (available memory, process memory and swapping)
- Disk (Reads/writes per second, disk queue)
- Network (I/O Bytes, duplicate ACKs or errors)

Check both at the system level and the process level.

32-bit processes can use at most 2 GB of virtual memory on Windows (normal OS settings) and 4 GB on Unix. Check to be sure these limits are not being reached.

Process Level

Oracle Hyperion Shared Services:

- HyS9SharedServices.exe
- Process of the Open LDAP-SLAPD.exe

Oracle Hyperion Financial Management:

- IIS Process
- HsxServer
- HsvDataSource
    - For HsvDataSource, monitor Private and Virtual memory rising together.

Oracle Hyperion Financial Reporting:

- HyS9FRWeb
- HRS9FRRMI.exe
- HRCommSrv.exe
- HRPrintSrv.exe
- HRReportSrv.exe
- HrSchedSrv.exe

Oracle Hyperion Web Analysis:

- HyS9WebAnalysis.exe

Oracle Hyperion Workspace:

- HyS9Workspace.exe

Oracle Hyperion Essbase:

- Essbase, esssvr – monitor additional counters: Disk Read Bytes/Sec

Oracle Hyperion Data Relationship Management:

- InetInfo – MDM Web Tier – IIS Process
- W3WP – MDM Web Tier – IIS Pool Process
- Dllhost – MDM Web Tier – manages dll based applications.
- Mdm_ntier_engine
- Mdm_ntier_RIO
- Mdm_ntier_service
- GWservices.exe
- MDM_web_pub_engine
- MDM_web_publishing
- Oracle or SQL Server database process (note: further database monitoring is required in addition to process monitoring)

Oracle Enterprise Performance Management Architect:

- HyS9EPMAWebTier
- dimension_server
- HyS9EPMA_EngineManager
- HyS9EPMA_JobManager
- HyS9EPMA_EventManager
- HyS9EPMADataSynchronizer

Oracle Hyperion Financial Data Quality Management:

- upsAppSv
- upsLBMgr
- InetInfo
- w3wp
- dllhost

Oracle Business Intelligence EE (OBIEE):

- sawserver (BI Presentation Server)

- nqserver (BI Server)

Summary

Load Testing Hyperion can be a major challenge due to the complexity and dynamic nature of Hyperion applications. Oracle Load Testing provides a comprehensive solution for Hyperion load testing.

Oracle White Paper-Load Testing Hyperion System 9 HFM
March 2010
Author: Matthew Demeusy
Contributing Author: Joe Fernandes

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment