Load Testing Hyperion System 9
HFM: Scripting Guide – Oracle
Load Testing for Web
Applications 8.30

An Oracle White Paper
July 2008
INTRODUCTION

This document is intended as a guide for creating load/performance test scripts against Hyperion Financial Applications (HFM) using Oracle Load Testing for Web Applications (version 8.30). This guide will assist during the script creation process and enable the scripter to create scripts faster and more reliably. It assumes that the person using this document has experience working with Oracle Load Testing for Web Applications. This document does not necessarily cover all parameters that have to be parameterized, nor does it guarantee that the parameters mentioned in this document will perfectly match your particular environment.

![Sample Hyperion Login Page](image)

Figure 1: Sample Hyperion Login Page

DYNAMIC PARAMETERS

The key to successful scripting is finding out the parameters that need to be dynamic in order to create functional scripts for transactions with different records. 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.

This describes some of the key parameters as well as how these may be found and identified.

Key parameters:

- sso_user
- sso_password
- sso_token
- repository_token
- ssnkeystate
- Entity
- CI (Composite Grid Identity)
- GI_n (Grid Identity)
- InstanceID (Random value)
- subcubes
- MODVAL
- FormPOV

**Script creation**

The preferred method for creating load scripts for Hyperion System 9 is to record proxy based scripts by using an external browser. (It is possible to create scripts by using the Oracle Functional Tester as “normal” as well as proxy recordings with the internal Oracle Functional tester browser, but easier to use the external browser recording)

**Recording Configuration**

If the target system (e.g. the Hyperion application) is only accessible by using a proxy server, it is important that this proxy server is added to the Oracle Functional Testing for Web Application configuration accordingly.
Figure 2: Proxy settings for successful Hyperion scripting (without external proxy server added)

Start the proxy recorder and then open MS Internet explorer to record your script.

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

Script example: Login to Hyperion System 9

One of the first scripts recorded will likely be a login script, where the user selects an 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.
Figure 3: Example of Recorded script that does a login, application selection and finally a logout.

Once finished, save the script and start the Navigation editor (in Oracle Functional Tester Press Ctrl + e)

Parameterizing scripts

The following describes the parameters and how to define these.

**SSO_USERNAME / SSO_PASSWORD**

Two parameters that are likely to be parameterized are the user credentials sso_username and sso_password. These parameters are likely to be based on Databanking (e.g. Data for these comes from a CSV/Text based file).
SSO_TOKEN

The navigation after the login POST contains another very important value: the sso_token.

Mark the sso_token and create a custom dynamic value based on this (after the value has been marked, right click on it, select Custom Dynamic Value – Custom Dynamic Value Wizard...)
Figure 6: Dynamic Value Wizard – sso_token – Screen 1

Figure 7: Dynamic Value Wizard – sso_token – Screen 2
Be sure to check the option “Look across multiple lines” for your regexp, or you may experience problems playing back different users.
The SSO token should be added to a Library, since it will be used for all scripts.

**NOTE:** The sso token is used in scripts both with and without URL encoding. For all occurrences used as parameters in requests (e.g. POST requests), the sso_token must to be URL Encoded.

If scripts are not played back correctly and error messages indicate that there is a problem with sessions, check the requests and try to change the sso token.
The next important parameter to manage is the repository_token. The repository token can be found on the page after the login.

Mark the repository_token value and create a custom dynamic value.
As seen in the screenshot above, the custom dynamic value wizard is not able to find the **repository_token**. The Navigation Editor scans through the HTML, URL, Post Data or Redirected URL, the value for the repository token however can be found in the Request header on the navigation before the page where the token appears for the first time.
Select previous navigation (keep the same page) and instead of looking in the HTML, select the Request Header and click Next>.

The navigation editor will now find the **repository_token** and suggest a regular expression to extract this value.

![Figure 16: Dynamic Value Wizard – repository_token – Screen 4](image)

The suggested regular expression is not ideal, since the time zone, usersvr and LanguageID may change, mark them in the Regular expression field and delete them. The expression that should be left should look like the following:
Click next and save the **repository_token** pattern in your library:

![Dynamic Value Wizard – repository_token – Screen 5](image1)

![Saving repository_token into pattern library](image2)

**Apply pattern library**
At this point you can apply the pattern library to the script. This will replace all static **sso_tokens** and the **repository_token** in your script with custom dynamic values (e.g. variables).

In the Navigation Editor: Select **RUN – Rebuild navigations** and apply the patterns.

**Figure 19: Rebuilding navigations**

After this operation the login script should contain all dynamic parameters required to run. (Please note that the sso_token may have to be ENCODED)

**Other scripts / 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.

**JSClientConstants.asp**

When working with Data Entry Forms a commonly found request is the JSClientConstants.asp page. The JSClientConstants.asp always contains a dynamic value coming from the previous page. Make sure that all “ClientConstants” are taken care of in order to enter data correctly into the system.

**Figure 20: JSClientConstants.asp navigation**
Create a custom dynamic value by first marking the value (in this case 38356%E32). Then, click through the Wizard and assign it a name (e.g., CCf). The value is always found on the page before and will never be used again.

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).

Figure 21: ssnkeystate

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.
TIPS AND TRICKS

Before testing

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.

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 for Web applications 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

**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

**MONITORING**

**Key areas to monitor**

- CPU utilization
- 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**

Collect these counters for all Apache, Java, BIService, das, lmgrd, HYSLD, slapd processes:

- \HOSTNAME\Process(ProcessName)\% Processor Time
- \HOSTNAME\Process(ProcessName)\Handle Count
- \HOSTNAME\Process(ProcessName)\ID Process
- \HOSTNAME\Process(ProcessName)\IO Data Bytes/sec
- \HOSTNAME\Process(ProcessName)\Page Faults/sec
- \HOSTNAME\Process(ProcessName)\Page File Bytes
- \HOSTNAME\Process(ProcessName)\Thread Count
- \HOSTNAME\Process(ProcessName)\Virtual Bytes
- \HOSTNAME\Process(ProcessName)\Working Set
### Machine Level

<table>
<thead>
<tr>
<th>HOSTNAME\Processor_Total% Processor Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOSTNAME\Memory\Available MBytes</td>
</tr>
<tr>
<td>HOSTNAME\Memory\Pages/sec</td>
</tr>
<tr>
<td>Interface(xxxx Gigabit Server Adapter _2)\Bytes Received/Sec</td>
</tr>
<tr>
<td>Interface(xxxx Gigabit Server Adapter _2)\Bytes Sent/sec</td>
</tr>
<tr>
<td>Interface(xxxx Gigabit Server Adapter _2)\Bytes Total/sec</td>
</tr>
<tr>
<td>Interface(xxxx Gigabit Server Adapter _2)\Current Bandwidth</td>
</tr>
<tr>
<td>HOSTNAME\PhysicalDisk(0 C:)% Disk Time</td>
</tr>
<tr>
<td>HOSTNAME\PhysicalDisk(0 C:)\Avg. Disk Queue Length</td>
</tr>
<tr>
<td>HOSTNAME\PhysicalDisk(0 C:)\Disk Bytes/sec</td>
</tr>
</tbody>
</table>