Deploying Oracle Hyperion 11.1 with NetScaler

This deployment guide focuses on defining the deployment process for Oracle Hyperion 9.2 with Citrix NetScaler. It includes information on setting up basic deployment and optimization.
Table of Contents

Introduction 3
Configuration 3
NetScaler features to be enabled 4
Steps for load balancing configuration 4
Solution Description 5
Quick Configuration Table 5
Configuring Load Balancing 7
Verification 11
Authentication 11
Configuring Optimization on NetScaler 12
HTTP Compression 12
Integrated Caching 14
Front End Optimization 18
Conclusion 20
Citrix NetScaler is a world-class product with the proven ability to load balance, accelerate, optimize, and secure enterprise applications.

For several years, Citrix has completed certifications and provided deployment guides for key enterprise applications. NetScaler’s rich application delivery capabilities significantly enhance the performance of these applications. With a comprehensive feature set, it provides availability, scalability, optimization and security for Oracle Hyperion deployments.

**Introduction**

This guide defines the process for deploying Oracle Hyperion 12c with NetScaler. Citrix NetScaler is a world class application delivery controller, with the proven ability to load balance, accelerate, secure and optimize enterprise applications.

Oracle Hyperion Financial Management provides financial consolidation and reporting that enables rapid consolidation and reporting of financial results, meeting global regulatory requirements, reduces the cost of compliance and delivers confidence in the numbers.

**Configuration**

[Diagram showing the flow of Oracle products with NetScaler at the center]
### Recommended Product Versions

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Hyperion</td>
<td>11.1</td>
</tr>
<tr>
<td>NetScaler VPX</td>
<td>11.0 (Enterprise/Platinum License) – Load Balancing, Compression, Caching and FEO&lt;br&gt;11.0 (Standard License) – Only Load Balancing</td>
</tr>
</tbody>
</table>

### NetScaler features

The following NetScaler features are discussed in this deployment guide.
- Load balancing
- Front End Optimization, Compression, Caching

**Load balancing**

NetScaler load balancing evenly distributes requests to backend servers. Multiple algorithms (such as LEAST-CONNECTION, ROUNDROBIN etc.) are supported to provide efficient load balancing logic for every application server.

**HTTP Compression**

Compression of HTTP traffic using standard GZIP/DEFLATE compression methods.

**Front End Optimization (FEO)**

Advance optimization feature, FEO enables NetScaler to significantly accelerate web content with various acceleration methods such as image compression etc.

**Integrated Caching**

Content caching allows NetScaler to serve frequently used content without requiring round trips to the source webserver.

**Other considerations**
- Make sure you have installed, at a minimum, one license on the NetScaler appliance.
- Set the time zone and a NTP (Network Time Protocol) server, and check the date and time on the NetScaler virtual appliance, as server connections can be sensitive to time differences.
- Configure your DNS settings properly: Note that for the purposes of certificate-based authentication, all addressable hosts that are part of the network setup should have resolvable domain names, not just IP addresses.
## Quick Configuration Table

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Version</th>
</tr>
</thead>
</table>

### Virtual Servers: esslb (Suggested Name)

<table>
<thead>
<tr>
<th>esslb</th>
</tr>
</thead>
</table>

- **Protocol:** HTTP
- **Port:** 80 (or alternate as per your configuration)
- **Load Balancing Method:** Roundrobin/LeastConnection
- **Services Bound:** ess1, ess2
- **Compression Policy:** ess_Compression_Test
- **Cache Policy:** ess_Cache_Test
- **FEO Policy:** ess_Optimization_Test
- **Persistence:** Source IP

**CLI Commands:**

```
add lb vserver esslb HTTP <IP address for vserver> 80 -persistenceType SOURCEIP -lbMethod ROUNDROBIN -cltTimeout 180 -downStateFlush DISABLED
```

### Service Configuration

#### (System>Load Balancing>Services)

- **Note:** Both backend services are HTTP here

<table>
<thead>
<tr>
<th>ess1</th>
<th>ess2</th>
</tr>
</thead>
</table>

- **Protocol:** HTTP
- **Port:** 80 (or alternate as per your configuration)
- **IP:** IP address of 1st Hyperion server

**CLI Commands:**

```
add service ess1 <IP address for 1st front end server> HTTP 80 -gslb NONE -maxClient 0 -maxReq 0 -cip ENABLED X-Forwarded-for -usip NO -useproxyport NO -sp ON -cltTimeout 180 -svrTimeout 360 -CKA NO -TCPB NO -CMP YES
add service ess2 <IP address for 2nd front end server> HTTP 80 -gslb NONE -maxClient 0 -maxReq 0 -cip DISABLED -usip NO -useproxyport NO -sp ON -cltTimeout 180 -svrTimeout 360 -CKA NO -TCPB NO -CMP YES
```

### Compression Policy Definition

#### (Optimization>Integrated Caching>Policies)

<table>
<thead>
<tr>
<th>Policy Name: Hyperion_Compression_Test</th>
</tr>
</thead>
</table>

- **Response Action:** COMPRESS (GZIP/DEFLATE should work too)
- **Expression:** ns_true

**CLI Commands:**

```
add cmp policy Hyperion_Compression_Test -rule ns_true -resAction GZIP
bind lb vserver Hyperion_lb -policyName Hyperion_Compression_Test -priority 100
bind lb vserver Hyperion_lb_ssl -policyName Hyperion_Compression_Test -priority 100
```
**Configuration Item**

**Version**

Policy Name: Hyperion_Cache_Test  
Actions: CACHE  
Undefined-Result Action: -Global-undefined-result-action (or NOCACHE/RESET)  
Expression: ns_true  

Cache Content Group:  
Name: Test  
Type: HTTP  
Expiry Method: Heuristic (Recommended)/Custom (if specific settings are required)  
Default Expiry Times: As per requirement; set to 233 for test deployment.  
Parameterization: Leave values as is (unless Cache selectors are in use; not configured for our test setup)  
Memory: Define values as per your system limits  
Others: Use default settings. All settings have context-sensitive help available if modification is required.  

CLI Commands:  
add cache policy Hyperion_Caching_Test -rule "SYS.EVAL_CLASSIC_EXPR("ns_true")" -action CACHE -storeInGroup Hyperion_Caching_Test  

---

**Cache Policy**  
(Optimization>Integrated Caching>Policies)

**Configuration Item**

**Version**

Policy Name: Hyperion_Optimization_Test  
Optimization Action: MODERATE (Preconfigured)  
Expression: HTTP.REQ.HEADER("Accept").CONTAINS("html")  

Alternate Configuration (Custom Policy):  
Optimization Policy Name: Hyperion_Optimization_TestCustom  
Optimization Action: samplefeo  
Expression: HTTP.REQ.HEADER("Accept").CONTAINS("html")  

Hyperion_Optimization_TestCustom Configuration:  
Enabled Settings: JavaScript/Make Inline, JavaScript/Move to End of Body Tag, JavaScript/Minify, Image/Optimize, Image/Lazy Load, Image/Shrink to Attributes, Image/Optimize, Image/Convert to JXR format, Image/Convert GIF to PNG, CSS/Make Inline, CSS/Move to Head Tag, CSS/Minify, CSS/Image Inline, CSS/Combine, CSS/Convert Imports to Links, HTML/Remove Comments from HTML  

CLI Commands:  
add feo policy Hyperion_Optimization_Test "HTTP.REQ.HEADER("Accept") CONTAINS("html")" MODERATE  
add feo policy Hyperion_Optimization_Testcustom "HTTP.REQ.HEADER("Accept") CONTAINS("html")" MS_SP_custom  
bind lb vserver Hyperion_lb -policyName Hyperion_Optimization_Testcustom -priority 100 -gotoPriorityExpression END -type REQUEST  
bind lb vserver Hyperion_lb_ssl -policyName Hyperion_Optimization_Test -priority 100 -gotoPriorityExpression END -type REQUEST  

---

**FEO (Front End Optimization) Policy**  
(Optimization>Front end Optimization>Policies)

**Configuration Item**

**Version**

Optimization Policy Name: Hyperion_Optimization_Test  
Optimization Action: MODERATE (Preconfigured)  
Expression: HTTP.REQ.HEADER("Accept").CONTAINS("html")  

Hyperion_Optimization_TestCustom Configuration:  
Enabled Settings: JavaScript/Make Inline, JavaScript/Move to End of Body Tag, JavaScript/Minify, Image/Optimize, Image/Lazy Load, Image/Shrink to Attributes, Image/Optimize, Image/Convert to JXR format, Image/Convert GIF to PNG, CSS/Make Inline, CSS/Move to Head Tag, CSS/Minify, CSS/Image Inline, CSS/Combine, CSS/Convert Imports to Links, HTML/Remove Comments from HTML  

CLI Commands:  
add feo policy Hyperion_Optimization_Test "HTTP.REQ.HEADER("Accept") CONTAINS("html")" MODERATE  
add feo policy Hyperion_Optimization_Testcustom "HTTP.REQ.HEADER("Accept") CONTAINS("html")" MS_SP_custom  
bind lb vserver Hyperion_lb -policyName Hyperion_Optimization_Testcustom -priority 100 -gotoPriorityExpression END -type REQUEST  
bind lb vserver Hyperion_lb_ssl -policyName Hyperion_Optimization_Test -priority 100 -gotoPriorityExpression END -type REQUEST
Solution Description

Configuring Load Balancing

The configuration of load balancing consists of the definition of and load balancing virtual servers, as well as services that are linked to the LB vservers and bound to the individual Oracle Hyperion servers.

Step 1 - Define the load balancing virtual servers (LB vservers)
Log into the NetScaler GUI. On the Configuration tab, navigate to Traffic Management>Load Balancing>Virtual Servers. For this deployment exercise, we are load balancing two Oracle Hyperion server instances. To demonstrate the same, we create one load balancing virtual server – esslb (HTTP Port 80).

When defining a new LB vserver, you will be presented with the settings screen. Here, set the protocol to HTTP for the LB vserver. Set the IP address to an appropriate value.

Load Balancing Virtual Server

Basic Settings

Create a virtual server by specifying a name, an IP address, a port, and a public IP address. If the application is accessible only from the local area address, you can configure multiple virtual servers to receive client requests.

Name

esslb

Protocol

HTTP

IP Address Type

IP Address

10.10.15.23

Port

80
After clicking OK, you will see the Basic Settings screen for the LB vserver. Here, you may change settings such as the session persistence method, authentication and load balancing methods. Set session persistence to SOURCEIP and the load balancing method to LEASTCONNECTION for both virtual servers.

For more information on these features, please refer to https://docs.citrix.com/en-us/netscaler/11.html

### Load Balancing Virtual Server

<table>
<thead>
<tr>
<th>Load Balancing Virtual Server</th>
<th>Export as a Template</th>
</tr>
</thead>
</table>

#### Basic Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>csslb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>HTTP</td>
</tr>
<tr>
<td>State</td>
<td>DOWN</td>
</tr>
<tr>
<td>IP Address</td>
<td>10.105.15.23</td>
</tr>
<tr>
<td>Port</td>
<td>80</td>
</tr>
<tr>
<td>Traffic Domain</td>
<td>0</td>
</tr>
</tbody>
</table>

| Listen Priority | - |
| Listen Policy Expression Range | NONE |
| Redirect Mode | IP |
| RHI State | PASSIVE |
| AppFlow Logging | ENABLED |

#### Services and Service Groups

- No Load Balancing Virtual Server Service Binding
- No Load Balancing Virtual Server Service Group Binding

Optionally, you can also set this vserver up for SSL. More information on this is available on the following page.
To enable an SSL-based LB vserver, you should add an SSL certificate and key pair. For this, you may use either a self-signed certificate generated on the NetScaler appliance or a CA (Certificate Authority) signed one. The steps for generating a self-signed certificate on the NetScaler are as follows –
1. Login to your NetScaler appliance via the Configuration Utility.
2. Select Traffic Management > SSL
4. Here, the wizard will lead you through the series of steps for generating the self signed certificate –
   - Generate the private key
   - Generate the CSR (Certificate Signing Request)
   - Generate the Certificate (using the ns-root.cer NetScaler root certificate)
   - Save the Certificate and Key pair

Alternatively, if a certificate and key pair is already available, the same can be added by navigating to SSL>Certificates and clicking on the Add button. For more details refer to http://support.citrix.com/article/CTX109260

To improve site security and achieve an A/A+ rating on the SSLLabs.com evaluation, refer to https://www.citrix.com/blogs/06/09/scoring-an-a-at-ssllabs-com-with-citrix-netscaler-update/
Step 2 – Define LBVS server service group binding

Now click on the Load Balancing Virtual Server Service Binding tab in the Service and Service Groups section, or alternatively, click on Services in the Traffic Management>Load Balancing subsection and then, click on the Add button.

Every LB service is linked to a server; this can either be a new server or an existing server already defined in the Servers subsection under Load Balancing. Service groups extend this by allowing the creation of a group of services. An LB vserver can use a set of services or a service group.

Here, define the names for the services for each Hyperion instance, the IP address (or choose from a list in the case of an existing server) for the new server and the protocol it operates on.

Recommended Best Practices:
- Name your server instances as per their role, not with the IP address (for example, the Oracle Hyperion servers can be named ess1 and ess2)
- As there will be multiple items linked to each application (LB vservers, services, policies among others), it is recommended that they be named appropriately for convenience. This will make using tools such as grep with the CLI a lot easier.
You should enable Health Monitoring if you would like to have NetScaler poll the server periodically to verify its health – it is recommended that this setting should not be disabled except for diagnostic purposes. This and additional settings can be accessed by clicking on the More dropdown (as shown above). If Health Monitoring is disabled, the appliance shows the server UP at all times. Bind these service groups to the appropriate LB vservers and confirm that they have been bound correctly by checking the same in the LB vserver Basic Settings screen. Add all the Hyperion servers to be load balanced and bind them to the load balancing virtual server.

### Load Balancing Virtual Server

<table>
<thead>
<tr>
<th>Basic Settings</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>8SS1D</td>
</tr>
<tr>
<td>Protocol</td>
<td>HTTP</td>
</tr>
<tr>
<td>State</td>
<td>UP</td>
</tr>
<tr>
<td>IP Address</td>
<td>10.105.15.23</td>
</tr>
<tr>
<td>Port</td>
<td>80</td>
</tr>
<tr>
<td>Traffic Domain</td>
<td>0</td>
</tr>
<tr>
<td>Listen Priority</td>
<td>NONE</td>
</tr>
<tr>
<td>Listen Policy Expression Range</td>
<td>1</td>
</tr>
<tr>
<td>Redirection Mode</td>
<td>IP</td>
</tr>
<tr>
<td>RHI State</td>
<td>PASSIVE</td>
</tr>
<tr>
<td>AppFlow Logging</td>
<td>ENABLED</td>
</tr>
</tbody>
</table>

#### Services and Service Groups

- 2 Load Balancing Virtual Server Service Bindings
- No Load Balancing Virtual Server Service Group Binding

### Persistence

<table>
<thead>
<tr>
<th>Persistence</th>
<th>SOURCEIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-out (msec)</td>
<td>2</td>
</tr>
<tr>
<td>IPv4 Netmask</td>
<td>255.255.255.255</td>
</tr>
<tr>
<td>IPv6 Mask Length</td>
<td>128</td>
</tr>
</tbody>
</table>

Finally, the LB vservers created will be displayed on the configuration screen to the right in the same screen that is obtained by accessing Traffic Management>Load Balancing>Virtual Servers.

This completes the load balancing configuration for Oracle Hyperion.
Verification

The functioning solution can be verified with a default Hyperion installation by navigating to http://esslb.ctxns.net/workspace

This will show the default welcome screen for Hyperion. Make sure popups are enabled as the login prompt is shown separately.
Configuring Optimization on NetScaler

NetScaler provides a flexible, comprehensive suite of optimization capabilities that can be categorized as follows:

- HTTP Compression
- Integrated Caching
- Front End Optimization (additional optimization capabilities)

To configure HTTP Compression, Integrated Caching and Front End Optimization, expand the Optimization tab in the NetScaler GUI’s left hand side navigation panel.

**HTTP Compression**

NetScaler’s optimization suite is, like other NetScaler features, driven by a policy-action architecture.

To enable HTTP Compression for a particular service, you should:

- Define the HTTP Compression Policy and Action
- Bind the same to the relevant virtual server

To define the Compression Policy and Action, click on the Policies option under HTTP Compression, shown above. This gives you the following screen –

![Optimization](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Expression</th>
<th>Response Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmp_javascript</td>
<td>`REQ.HTTP.URL != '/js'</td>
<td></td>
</tr>
</tbody>
</table>

To add a new compression policy, click on the Add button. This will give you the following screen –
Here, you can define a name for the policy, an Expression that defines when this policy is triggered (for example, when a particular URL is encountered. To make the policy apply to all content, use `ns_true` in the Expression window. For assistance here, click on the Frequently Used Expressions drop down) and the Response Action that should be taken. Here, the Actions available are COMPRESS (GZIP or DEFLATE compression, with GZIP given priority), GZIP (GZIP standard compression), DEFLATE (DEFLATE compression) and NOCOMPRESS.

Here, you have the option to either add a new Action or reconfigure the existing ones. You can Add using the + button, or edit/configure using the pencil-shaped button. Either option gives you a screen similar to the one shown below.

Vary Header Insertion is an option that is relevant for caching; the value of the Vary header allows for different cache results to be returned for similar requests. For now, we will not be changing the options presented here. You can add a new action that uses a compression type of your choice.
For the Hyperion deployment, the following settings have been used for HTTP compression –
Policy Name: Hyperion_Compression_Test
Response Action: GZIP (Compress/DEFLATE should work too)
Expression: ns_true

**Integrated Caching**

To configure caching, you can use the integrated wizard that makes configuration very straightforward. To initiate the wizard, navigate to Optimization>Integrated Caching as shown below:

Here, you can initiate the Caching Wizard under Getting Started.

The first step requires you to specify the content type. This can be either static (examples given) or dynamic. Helpful hints are provided as shown above to help determine which type of content is relevant for you.
The next step involves defining which content should be cached. The Frequently Used Expressions dropdown helps you define the correct expression; however, if you want the caching policy to run for all content, you can use `ns_true` as the expression as shown below:
The next step involves definition of the caching space to be used on the NetScaler and the minimum size of objects to be cached.
Finally, the cache policy should be bound to the relevant vserver.

Static Content Caching Wizard

<table>
<thead>
<tr>
<th>Cache Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
</tr>
<tr>
<td>Test</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content Expiration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expiry Type</td>
</tr>
<tr>
<td>Weak relative expiry for negative (error) responses eg: 4xx 5xx</td>
</tr>
<tr>
<td>Weak relative expiry for positive (non-error) responses eg: 2xx 3xx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimize Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Abort Size: Continue caching if more than 4194303</td>
</tr>
<tr>
<td>D: 81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cache Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Load Balancing Virtual Server Request Binding</td>
</tr>
<tr>
<td>No Content Switching Virtual Server Request Binding</td>
</tr>
</tbody>
</table>

These definitions can be made under Cache Policies as shown in the screenshot above.

For the Hyperion deployment, the following settings have been used for caching –
Policy Name: Hyperion_Cache_Test
Actions: CACHE
Cache Content Group: Test
Undefined-Result Action: -Global-undefined-result-action (or NOCACHE/RESET)
Expression: ns_true

Cache Content Group:
Name: Test
Type: HTTP
Expiry Method: Heuristic (Recommended)/Custom (if specific settings are required)
Default Expiry Times: As per requirement, set to 233 for test deployment.
Parameterization: Leave values as is (unless Cache selectors are in use, not configured for our test setup)
Memory: Define values as per your system limits
Others: Use default settings. All settings have context-sensitive help available if modification is required.
**Front End Optimization**

The front end optimization feature set makes NetScaler an extremely capable optimization device by implementing enhanced optimization routines for specific front end entities such as images, JavaScript etc. These features provide improved optimization performance than that achieved by compression and caching.

Front End Optimization capabilities can be activated by navigating to Optimization>Front End Optimization. As with all NetScaler features, these are implemented using a policy-action mechanism.

To add a new policy, navigate to Optimization>Front End Optimization and then, click on Policies. To add a new policy, click on Add in the section displayed to the right of the navigation menu.

This will give you the following screen for definition of a new FEO policy.
The Expression here works much on the same lines as expressed for the earlier features; the Frequently Used Expressions drop down can be used for assistance. There are certain predefined actions that can be assigned here, all of which have different configurations for the same settings; you can also either edit or create a custom action, which can be done using the plus or pencil buttons next to the Action name. The Plus icon enables the setup of a custom profile.

Upon clicking either of these buttons, the following screen (or a similar one) is observed:

```
Configure Front End Optimization Action

Name
MODERATE

JavaScript
✓ Make Inline
✓ Minify

Image
✓ Shrink to Attributes
✓ Make Inline
✓ Optimize
✓ Convert to JPEG
✓ Lazy Load
✓ Convert to WEBP

CSS
✓ Make Inline
✓ Move to End of Body Tag
✓ Combine
✓ Convert Imports to Links
✓ Minify

HTML
✓ Remove comments from HTML

Miscellaneous optimization
✓ Extend Page Cache
✓ Enable Client Side Measurements
```

This screen presents all the various front end optimization options available; NetScaler can help to optimize web traffic with JavaScript, Image, CSS (Cascading Style Sheets), HTML and Miscellaneous Optimization. This last section also allows for domain sharding, which splits resources across subdomains to improve optimization and page load times.

For this deployment, the recommended FEO policy setting is MODERATE; this default setting provides a good level of optimization while not affecting the performance of the Hyperion setup. The lab tests show an approximate 65 percent reduction in load times, 10-15 percent reduction in the amount of data transferred and 10-15 percent reduction in number of requests on our test setup for generic operations. Results may differ for your setup.
Optimization settings for the Oracle Hyperion deployment:

Optimization Policy Name: Hyperion_Optimization_Test
Optimization Action: MODERATE (Preconfigured)
Expression: HTTP.REQ.HEADER("Accept").CONTAINS("html")

Alternate Configuration (Custom Policy)
Optimization Policy Name: Hyperion_Optimization_TestCustom
Optimization Action: samplefeo
Expression: HTTP.REQ.HEADER("Accept").CONTAINS("html")

Hyperion_Optimization_TestCustom Configuration:
Enabled Settings: JavaScript/Make Inline, JavaScript/Move to End of Body Tag, JavaScript/Minify, Image/Optimize, Image/Lazy Load, Image/Shrink to Attributes, Image/Optimize, Image/Convert to JXR format, Image/Convert GIF to PNG, CSS/Make Inline, CSS/Move to Head Tag, CSS/Minify, CSS/Image Inline, CSS/Combine, CSS/Convert Imports to Links, HTML/Remove Comments from HTML

Conclusion
NetScaler enables highly available Oracle Hyperion deployments with its load balancing capabilities. With NetScaler, enterprises can enable a host of additional capabilities including but not limited to authentication offload, end point analysis checks, selective server access, URL rewrites, compression, caching, front end optimizations and much more.

With NetScaler, enterprises can not only enable high availability for their Oracle Hyperion environments, but also extend capabilities for security and optimized access. The policy engine used by NetScaler enables enterprises to deploy any specific use cases that they may require, making the NetScaler solution a flexible and robust one that can meet all enterprise requirements.

To learn more about how NetScaler can bring these benefits to Oracle Hyperion installations or address other application delivery requirements, please visit http://www.citrix.com.