Deploying Oracle SOA Suite with NetScaler

This guide focuses on defining the process for deploying Oracle SOA Suite with Citrix NetScaler
# 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 SOA Suite deployments.

**Introduction**

This guide defines the process for deploying Oracle SOA Suite 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.

The rapid adoption of cloud-based applications by the enterprise, combined with organizations’ desire to integrate applications with mobile technologies, is dramatically increasing application integration complexity. Oracle SOA Suite 12c, the latest version of the industry’s most complete and unified application integration and SOA solution, meets this challenge. With simplified cloud, mobile, on-premises and Internet of Things (IoT) integration capabilities, all within a single platform, Oracle SOA Suite 12c delivers faster time to integration, increased productivity and lower TCO.

**Configuration**
Recommended Product Versions

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle SOA Suite Server</td>
<td>12c</td>
</tr>
<tr>
<td>NetScaler VPX</td>
<td>11.0 (Platinum License) – Load Balancing, Compression, Caching and FEO</td>
</tr>
<tr>
<td></td>
<td>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
- SSL offload
- Compression
- Caching
- FEO (Front End Optimization)

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 SOA Suite Server server connections can be very 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.

Steps for load balancing configuration

Broadly, the steps to configure a load balanced SOA Suite Server setup are as follows:

1. Complete initial setup for the SOA Suite Server;
2. Create a service for each SOA Suite Server and bind the server objects and appropriate monitors to it.
3. Now, create load balancing virtual servers (load balancing vservers) for the SOA Suite Server service and bind the appropriate services and certificate to them. For this deployment, we have used a self-signed certificate; however you may use any valid server certificate.
4. When defining the load balancing vservers, provide a valid, addressable IP address.
5. Set an appropriate load balancing method (such as LEASTCONNECTION) and a persistence method such as SOURCEIP. These will ensure effective load balancing.
# Solution Description

## Quick Configuration Table

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Virtual Servers:</strong> soa_lb_ssl, soa_lb (Suggested Names)</td>
<td></td>
</tr>
<tr>
<td>SOA_LB_SSL</td>
<td>SOA_LB</td>
</tr>
<tr>
<td><strong>Protocol:</strong> HTTPS</td>
<td>Protocol: HTTP</td>
</tr>
<tr>
<td><strong>Port:</strong> 443 (or alternate as per your configuration)</td>
<td>Port: 80 (or alternate as per your configuration)</td>
</tr>
<tr>
<td><strong>Load Balancing Method:</strong> Roundrobin / LeastConnection</td>
<td><strong>Load Balancing Method:</strong> Roundrobin / LeastConnection</td>
</tr>
<tr>
<td><strong>Services Bound:</strong> SOA1_svc, SOA2_svc</td>
<td><strong>Services Bound:</strong> SOA1_svc, SOA2_svc</td>
</tr>
<tr>
<td><strong>Compression Policy:</strong> SOA_Compression_Test</td>
<td><strong>Compression Policy:</strong> SOA_Compression_Test</td>
</tr>
<tr>
<td><strong>Cache Policy:</strong> SOA_Cache_Test</td>
<td><strong>Cache Policy:</strong> SOA_Cache_Test</td>
</tr>
<tr>
<td><strong>FEO Policy:</strong> SOA_Optimization_Test</td>
<td><strong>FEO Policy:</strong> SOA_Optimization_Test</td>
</tr>
<tr>
<td><strong>Certificate Binding:</strong> Standard</td>
<td><strong>Certificate Binding:</strong> Standard</td>
</tr>
<tr>
<td>Wildcard/SAN/SNI Server certificate support (Bind the appropriate server certificate as per your configuration)</td>
<td>Wildcard/SAN/SNI Server certificate support (Bind the appropriate server certificate as per your configuration)</td>
</tr>
<tr>
<td><strong>CLI Commands:</strong></td>
<td><strong>CLI Commands:</strong></td>
</tr>
<tr>
<td>add lb vserver SOA_lb_ssl SSL &lt;IP address for vserver&gt; 443 -persistenceType NONE -cltTimeout 180</td>
<td>add lb vserver SOA_LB HTTP &lt;IP address for vserver&gt; 80 -persistenceType NONE -lbMethod ROUNDROBIN -cltTimeout 180 -downStateFlush DISABLED</td>
</tr>
</tbody>
</table>

## Load Balancing

(Traffic Management>Load Balancing>Virtual Servers in the GUI)

## Service Configuration

(System>Load Balancing>Services)

Note: Both backend services are HTTP here

<table>
<thead>
<tr>
<th>SOA1_svc</th>
<th>SOA2_svc</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Protocol:</strong> HTTP</td>
<td><strong>Protocol:</strong> HTTP</td>
</tr>
<tr>
<td><strong>Port:</strong> 80 (or alternate as per your configuration)</td>
<td><strong>Port:</strong> 80 (or alternate as per your configuration)</td>
</tr>
<tr>
<td><strong>IP:</strong> IP address of 1st SOA Suite server</td>
<td><strong>IP:</strong> IP address of 2nd SOA Suite server</td>
</tr>
</tbody>
</table>

**CLI Commands:**

add service SOA1_svc <IP address for 1st CRM front end server> HTTP 80 -gslb NONE -maxClient 0 -maxReq 0 -cip ENABLED X-Forwarded-for -usip NO -useproxyport NO -sp ON -ctimeout 180 -svrTimeout 360 -CKA NO -TCPB NO -CMP YES

add service SOA2_svc <IP address for 2nd CRM front end server> HTTP 80 -gslb NONE -maxClient 0 -maxReq 0 -cip DISABLED -usip NO -useproxyport NO -sp ON -ctimeout 180 -svrTimeout 360 -CKA NO -TCPB NO -CMP YES

## Compression Policy Definition

(Optimizations>Integrated Caching>Policies)

**Policy Name:** SOA_Compression_Test

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

**Expression:** ns_true

**CLI Commands:**

add cmp policy SOA_Compression_Test -rule ns_true -resAction GZIP
bind lb vserver SOA_Lb -policyName SOA_Suite_Compression_Test -priority 100
bind lb vserver SOA_Lb_ssl -policyName SOA_Suite_Compression_Test -priority 100
### Configuration Item

Policy Name: SOA_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  
Expire 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:**

```bash
add cache policy SOA_Caching_Test -rule "\"ns_true\"" -action CACHE -storeInGroup SOA_Caching_Test
```

### FEO (Front End Optimization) Policy

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

**Alternate Configuration (Custom Policy):**

Optimization Policy Name: SOA_Optimization_TestCustom  
Optimization Action: samplefeo  
Expression: HTTP.REQ.HEADER("Accept").CONTAINS("html")

**SOA_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:**

```bash
add feo policy SOA_Optimization_Test "HTTP.REQ.HEADER("Accept")\"CONTAINS\"html\"\" MODERATE
add feo policy SOA_Optimization_Testcustom "HTTP.REQ.HEADER("Accept") CONTAINS("html")" MS_CRM_custom
bind lb vserver SOA_lb -policyName SOA_Suite_Optimization_Testcustom -priority 100 -gotoPriorityExpression END -type REQUEST
bind lb vserver SOA_lb_ssl -policyName SOA_Suite_Optimization_Test -priority 100 -gotoPriorityExpression END -type REQUEST
```
Configuring Load Balancing

A load balancing configuration consists of the definition of load balancing virtual servers (LB vServers), as well as services that are bound to the LB vservers. A service is simply a combination of a server and a protocol (e.g., HTTP, Port 80 or HTTPS, port 443).

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 SOA Suite Servers. Here, we will create two load balancing virtual servers – SOA Suite_lb (HTTP Port 80) and SOA Suite_lb_ssl (HTTPS/SSL Port 443). Note that either one of the two can also be set up and will suffice for successful load balancing.

When defining a new LB vserver, you will be presented with the settings screen. Here, set the protocol to HTTP for the first vserver and SSL for the second. Set the IP address to the appropriate value. (The steps shown here are for the SSL vserver. Follow the same steps to configure the HTTP vserver as well, only select port 80 as the port and HTTP as the protocol)

Load Balancing Virtual Server
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

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/2016/06/09/scoring-an-a-at-ssllabs-com-with-citrix-netscaler-2016-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 SOA Suite server 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. For this deployment, the IPs will correspond to 10.105.157.177 for the first server and 10.105.157.178 for the second one.

Recommended Best Practices:
- Name your server instances as per their role, not with the IP address (for example, the Oracle SOA Suite servers can be named SOA Suite1 and SOA Suite2)
- 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. For example, the servers above can be named SOA Suite1_svr, the services they bind to can be called SOA Suite1_svc etc. 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 SOA Suite Server servers to be load balanced as appropriate services, then bind them to the load balancing virtual servers.

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 essential load balancing configuration for SOA Suite Server.
**Verification**

The functioning solution can be verified with a default SOA suite installation by navigating to https://<FQDN of SOA LB vserver>/soa-infra. For details on deploying the Oracle SOA Suite, refer to Oracle’s documentation for setting up a high availability SOA suite deployment. This will show a default screen with information on the SOA server that the client is currently connected to.

---

**Welcome to the Oracle SOA Platform on WebLogic**

SOA Version: v12.2.1.0.0 - 12.2.1.0.0.151013.0700.0085
WebLogic Server 12.2.1.0.0 (12.2.1.0.0)
Running on: soa_server1

The following composites are currently deployed:

1. default/InsertEmp!1.0*soa_7f98c85c-b7e4-491f-a589-246306d0178b
   - Test insertemp_client_ep
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 –

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 SOA Suite deployment, the following settings have been used for HTTP compression –
- Policy Name: SOA_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>233</td>
</tr>
<tr>
<td>Weak relative expiry for positive (non-error) responses eg: 2xx 3xx</td>
</tr>
<tr>
<td>233</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimize Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Abort Size:</td>
</tr>
<tr>
<td>Continue caching if more than 4194303</td>
</tr>
<tr>
<td>Do not cache - if size is less than 0</td>
</tr>
<tr>
<td>Do not cache - if size is less than 81</td>
</tr>
</tbody>
</table>

**Cache Policies**

- **No** Load Balancing Virtual Server Request Binding
- **No** Content Switching Virtual Server Request Binding

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

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

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

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 Oracle SOA Suite setup. In our deployment lab test scenario, with the recommended optimization settings enabled, it is possible to have over 60% improvement in SOA Suite server response rates.
Optimization settings for the Oracle SOA Suite deployment:

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

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

**SOA Suite_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 SOA Suite Server 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 SOA Suite 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.