

Optimization

Oct 13, 2015

Optimization

The NetScaler optimization features reduce transaction times between the clients and the servers, and they reduce bandwidth consumption. They also enhance server performance by offloading some tasks and making others more efficient.

Client Keep-Alive	Handles multiple requests on a single client connection. The client does not have to negotiate a new connection for each request to the server.
HTTP Compression	Compresses HTTP responses sent from the servers to compression-aware browsers. The smaller responses reduce download time and save bandwidth.
Integrated Caching	Stores responses to client requests. Subsequent requests for the same content are served from the NetScaler cache instead of being forwarded to the origin server.
Front End Optimization	Reduces the load and render time of web pages by simplifying and optimizing the content served to the client browser. Note: Supported from NetScaler 10.5 onwards.
Content Accelerator	Stores server responses on a Citrix ByteMobile T2100 appliance. Note: Supported from NetScaler 10.1 onwards.
SPDY (Speedy)	Acts as a SPDY gateway between clients and your servers, providing SPDY support without the need to configure/upgrade SPDY on the servers. Note: Supported from NetScaler 10.1 onwards.

Client Keep-Alive

The client keep-alive feature enables multiple client requests to be sent on a single client connection. This feature helps in a transaction management environment where typically the server closes the client connection after serving the response. The client then opens a new connection for each request and spends more time on the transaction.

Client keep-alive resolves this issue by keeping the connection between the client and the appliance (client-side connection) open even after the server closes the connection with the appliance. This allows sending multiple client requests using a single connection and saves the round trips associated with opening and closing a connection. Client keep-alive is most beneficial in SSL sessions.

Client keep-alive is also useful under either of the following conditions:

- When the server does not support client keep-alive.
- When the server supports client keep-alive but an application on the server does not support client keep-alive.

Note: Client keep-alive is applicable for HTTP and SSL traffic.

Client-keep alive can be configured globally to be able to handle all traffic. It can also be configured to be active only on specific services.

In client keep-alive environment, the configured services intercept the client traffic and the client request is directed to the origin server. The server sends the response and closes the connection between the server and the appliance. If a "Connection: Close" header is present in the server response, the appliance corrupts this header in the client-side response, and the client-side connection is kept open. As a result, the client does not have to open a new connection for the next request; instead, the connection to the server is reopened.

Note: If a server sends back two "Connection: Close" headers, only one is edited. This results in significant delays on the client rendering of the object because a client does not assume that the object has been delivered completely until the connection is actually closed.

Configuring Client Keep-Alive

Updated: 2014-08-12

Client keep-alive, by default, is disabled on the NetScaler, both globally and at service level. Therefore, you must enable the feature at the required scope.

Note: If you enable client keep-alive globally, it is enabled for all services, regardless of whether you enable it at the service level.

Additionally, if required, you can configure some HTTP parameters to specify the maximum number of HTTP connections retained in the connection reuse pool, enable connection multiplexing, and enable persistence Etag.

Note: When Persistent ETag is enabled, the ETag header includes information about the server that served the content. This ensures that cache validation conditional requests or browser requests, for that content, always reaches the same server.

To configure client keep-alive by using the command line interface

At the command prompt, do the following:

1. Enable client keep-alive on the NetScaler.
 - At global level

```
enable ns mode cka
```

- At service level

```
set service <name> -CKA YES
```

Note: Client keep-alive can be enabled only for HTTP and SSL services.

2. Configure the required HTTP parameters on the HTTP profile that is bound to the service(s).

```
set ns httpProfile <name> -maxReusePool <value> -conMultiplex ENABLED -persistentETag ENABLED
```

Note: Configure these parameters on the nshttp_default_profile HTTP profile, to make them available globally.

To configure client keep-alive by using the configuration utility

1. Enable client keep-alive on the NetScaler.

- o At global level

Navigate to System > Settings, click Configure Modes and select Client side Keep Alive.

- o At service level

Navigate to Traffic Management > Load Balancing > Services, and select the required service. In the Settings grouping, enable Client Keep-Alive.

2. Configure the required HTTP parameters on the HTTP profile that is bound to the service(s).

Navigate to System > Profiles, and on HTTP Profiles tab, select the required profile and update the required HTTP parameters.

HTTP Compression

For websites with compressible content, the NetScaler HTTP compression feature implements lossless compression to alleviate latency, long download times, and other network-performance problems by compressing the HTTP responses sent from servers to compression-aware browsers. You can improve server performance by offloading the computationally intensive compression task from your servers to the NetScaler appliance.

The following table describes the capabilities of the HTTP compression feature:

Functionality	Description
Compression Ratio	Compression ratio depends on the types of files in the responses, but is always significant, noticeably reducing amount of data transmitted over the network.
Browser Awareness	NetScaler serves compressed data to compression aware browsers only, reducing the transaction time between the client and the server. Most modern web browsers support HTTP compression.
Compression blocking	You can define content filters to selectively block compression by applying built-in actions.
Compression Caching	With the integrated caching feature enabled, subsequent requests for the same content are served from the local cache, reducing the number of round trips to the server and improving transaction times.
HTTPS Support	Compression is particularly useful on SSL connections, because it reduces the amount of content that has to be encrypted, either on the server or by the NetScaler appliance, and decrypted by the client.
Intelligent Response Filtering	The NetScaler compression engine intelligently filters server responses on the basis of defined compression parameters. For example, the compression engine detects zero-content-length responses and compressed responses and does not compress them. The detection of compressed responses enables origin sites to use server-based compression in conjunction with the NetScaler compression feature.
Compression Switching	The NetScaler appliance transparently directs requests from compression aware clients to compression capable servers, so that responses to those clients are compressed, and responses to other clients are not delayed by compression processing.

How Compression Works

A NetScaler ADC can compress both static and dynamically generated data. It applies the GZIP or the DEFLATE compression algorithm to remove extraneous and repetitive information from the server responses and represent the original information in a more compact and efficient format. This compressed data is sent to the client's browser and uncompressed as determined by the browser's supported algorithm or algorithms (GZIP or DEFLATE).

NetScaler compression treats static and dynamic content differently.

- Static files are compressed only once, and a compressed copy is stored in local memory. Subsequent client requests for cached files are serviced from that memory.
- Dynamic pages are dynamically created each time a client requests them.

When a client sends a request to the server:

1. The client request arrives at the NetScaler ADC. The ADC examines the headers and stores information about what kind of compression, if any, the browser supports.
2. The ADC forwards the request to the server and receives the response.
3. The NetScaler compression engine examines the server response for compressibility by matching it against policies.

4. If the response matches a policy associated with a compression action, and the client browser supports a compression algorithm specified by the action, the NetScaler ADC applies the algorithm and sends the compressed response to the client browser.
5. The client applies the supported compression algorithm to decompress the response.

Configuring HTTP Compression

By default, compression is disabled on the NetScaler ADC. You must enable the feature before configuring it. If the feature is enabled, the ADC compresses server requests specified by compression policies.

To configure HTTP compression, do the following:

- [Configure compression actions](#)
- [Configure compression policies](#)
- [Bind the compression policies to global bind points or to virtual servers](#)
- [Optionally, configure global compression parameters](#)

Enabling HTTP Compression

Compression can be enabled for HTTP and SSL services only. You can enable it globally, so that it applies to all HTTP and SSL services, or you can enable it just for specific services.

To enable compression by using the command line interface

At the command prompt, enter one of the following commands to enable compression globally or for a specific service:

- `enable ns feature cmp`
OR
- `set service <name> -CMP YES`

To configure compression by using the configuration utility

Do one of the following:

- To enable compression globally, navigate to **System > Settings**, click **Configure Basic Features**, and select **HTTP Compression**.
- To enable compression for a specific service, navigate to **Traffic Management > Load Balancing > Services**, select the service, and click **Edit**. In the **Settings** group, click the pencil icon and enable **Compression**.

Configuring a Compression Action

A compression action specifies the action to take when a request or response matches the rule (expression) in the policy with which the action is associated. For example, you can configure a compression policy that identifies requests that will be sent to a particular server, and associate the policy with an action that compresses the server's response.

There are four built-in compression actions:

- **COMPRESS**: Uses the GZIP algorithm to compress data from browsers that support either GZIP or both GZIP and DEFLATE. Uses the DEFLATE algorithm to compress data from browsers that support only the DEFLATE algorithm. If the browser does not support either algorithm, the browser's response is not compressed.
- **NOCOMPRESS**: Does not compress data.
- **GZIP**: Uses the GZIP algorithm to compress data for browsers that support GZIP compression. If the browser does not support the GZIP algorithm, the browser's response is not compressed.
- **DEFLATE**: Uses the DEFLATE algorithm to compress data for browsers that support the DEFLATE algorithm. If the browser does not support the DEFLATE algorithm, the browser's response is not compressed. After creating an action, you associate the action with one or more compression policies.

To create a compression action by using the command line interface

At the command prompt, enter the following command to create a compression action:

```
add cmp action <name> <cmpType> [-addVaryHeader <addVaryHeader> -varyHeaderValue <string>]
```

To create a compression action by using the configuration utility

Navigate to **Optimization > HTTP Compression > Actions**, click **Add**, and create a compression action to specify the type of compression to be performed on the HTTP response.

Configuring a Compression Policy

A compression policy contains a rule, which is a logical expression that enables the NetScaler appliance to identify the traffic that should be compressed.

When the NetScaler ADC receives an HTTP response from a server, it evaluates the built-in compression policies and any custom compression policies to determine whether to compress the response and, if so, the type of compression to apply. Priorities assigned to the policies determine the order in which the policies are matched against the requests.

The following table lists the built-in HTTP compression policies. These policies are activated globally when you enable compression.

Built-in Classic or Default Syntax Policy	Description
ns_nocmp_mozilla_47 ns_adv_nocmp_mozilla_47	Prevents compression of CSS files when a request is sent from a Mozilla 4.7 browser.
ns_cmp_mscss ns_adv_cmp_mscss	Compresses CSS files when the request is sent from a Microsoft Internet Explorer browser.
ns_cmp_msapp ns_adv_cmp_msapp	Compresses files that are generated by the following applications:s <ul style="list-style-type: none">Microsoft Office WordMicrosoft Office ExcelMicrosoft Office PowerPoint
ns_cmp_content_type ns_adv_cmp_content_type	Compresses data when the response containsContent-Type header and contains text.
ns_nocmp_xml_ie ns_adv_nocmp_xml_ie	Prevents compression when a request is sent, from a Microsoft Internet Explorer browser and the response contains a Content-Type header and contains text or xml.

To create a compression policy by using the command line interface

At the command prompt, enter the following command to create a compression policy:

```
add cmp policy <name> -rule <expression> -resAction <string>
```

To create a compression policy by using the configuration utility

Navigate to **Optimization > HTTP Compression > Policies** , click **Add** , and create a compression policy by specifying the condition and the corresponding action to be executed.

Binding a Compression Policy

To put a compression policy into effect, you must bind it either globally, so that it applies to all traffic that flows through the NetScaler ADC, or to a specific virtual server, so that the policy applies only to requests whose destination is the VIP address of that virtual server.

When you bind a policy, you assign it a priority. The priority determines the order in which the policies you define are evaluated. You can set the priority to any positive integer.

To bind a compression policy by using the command line interface

At the command prompt, enter one of the following commands to bind a compression policy globally or to a specific virtual server:

- bind cmp global <policyName> [-priority <positive_integer>] [-state (ENABLED|DISABLED)]...

- o bind lb vserver <vserverName> -policyName <policyName> -priority <positive_integer>. Repeat this command for each virtual server to which you want to bind the compression policy.

To bind a compression policy by using the configuration utility

Do one of the following:

- o At global level Navigate to **Optimization > HTTP Compression > Policies**, click **Policy Manager** and bind the required policies by specifying the relevant **Bind Point** and **Connection Type** (Request/Response).
- o At virtual server level
 - For load balancing virtual server, Navigate to **Traffic Management > Load Balancing > Virtual Servers**, select the required virtual server, click **Policies**, and bind the relevant policy.
 - For content switching **virtual server**, Navigate to **Traffic Management > Content Switching > Virtual Servers**, select the required virtual server, click **Policies**, and bind the relevant policy.

Setting the Global Compression Parameters for Optimal Performance

Many users accept the default values for the global compression parameters, but you might be able provide more effective compression by customizing these settings.

The following table describes the compression parameters that you can set on the NetScaler ADC.

Compression Parameters	Description
Quantum size	Size, in KB, of the buffer maintained for accumulating server responses. The responses are compressed when the buffer size exceeds this value. For example, if you set the quantum size to 50 KB, the NetScaler ADC compresses the buffer's contents when its size becomes larger than 50 KB. Minimum value: 1. Maximum value: 63488. Default: 57344.
Compression level	Level of compression to apply to server responses. Possible values: Best Speed, Best Compression, optimal.
Minimum HTTP response size	Minimum size, in bytes, of an HTTP response that is compressed. Responses smaller than the value specified by this parameter are sent without being compressed.
Bypass compression on CPU usage	NetScaler CPU usage, as a percentage, at or above which no compression is done. Default: 100.
Policy Type*	Type of policies used for compression. Possible values: Classic, Default Syntax. Default: Classic.
Allow Server-side compression	Allow servers to send compressed data to the NetScaler ADC.
Compress push packet	Upon receipt of a packet with a TCP PUSH flag, compress the accumulated packets immediately, without waiting for the quantum buffer to be filled.
External Cache	Issue a private response directive indicating that the response message is intended for a single user and must not be cached by a shared or proxy cache.

To configure global compression parameters by using the command line interface

At the command prompt, enter the following command to configure compression parameters that apply globally:

```
set cmp parameter -cmpLevel <cmpLevel> -quantumSize <integer> [-addVaryHeader ( ENABLED | DISABLED ) [-varyHeaderValue <string>]]...
```

Note: Vary header parameters are available from NetScaler 10.5 onwards.

To configure global compression parameters by using the configuration utility

Navigate to **Optimization > HTTP Compression** , click **Change Compression Settings** , and set the relevant parameters.

Evaluating Your Compression Configuration

You can view the compression statistics in the dashboard utility or in an SNMP monitor. The dashboard utility displays summary and detailed statistics in a tabular and graphic format.

Optionally, you can also view statistics for a compression policy, including the number of hits that the policy counter increments during the policy based compression.

Note:

- For more information about the statistics and charts, see the Dashboard help on the Citrix NetScaler appliance.
- For more information about SNMP, see [SNMP](#).

To View Compression Statistics by Using the Command Line Interface

At the command prompt, enter the following commands to display the compression statistics:

1. To display compression statistics summary
stat cmp
Note: The stat cmp policy command displays statistics for default syntax compression policies only.
2. To display compression policy hits and details

show cmp policy <name>
3. To display detailed compression statistics

stat cmp -detail

To View Compression Statistics by Using the Dashboard

In the Dashboard utility, you can display the following types of compression statistics:

- Select Compression to display a summary of the compression statistics.
- To display detailed compression statistics by protocol type, click the Details
- To display the rate of requests processed by the compression feature, click the Graphical View tab.

To View Compression Statistics by Using SNMP

You can view the following compression statistics by using the SNMP network management application.

- Number of compression requests (OID: 1.3.6.1.4.1.5951.4.1.1.50.1)
- Number of compressed bytes transmitted (OID: 1.3.6.1.4.1.5951.4.1.1.50.2)
- Number of compressible bytes received (OID: 1.3.6.1.4.1.5951.4.1.1.50.3)
- Number of compressible packets transmitted (OID: 1.3.6.1.4.1.5951.4.1.1.50.4)
- Number of compressible packets received (OID: 1.3.6.1.4.1.5951.4.1.1.50.5)
- Ratio of compressible data received and compressed data transmitted (OID: 1.3.6.1.4.1.5951.4.1.1.50.6)
- Ratio of total data received to total data transmitted (OID: 1.3.6.1.4.1.5951.4.1.1.50.7)

To View Additional Compression Statistics by Using the Configuration Utility

1. To display HTTP compression statistics:

Navigate to Optimization > HTTP Compression and click Statistics.
2. To display statistics of a compression policy

Navigate to Optimization > HTTP Compression > Policies> select the policy, and click Statistics.
3. To display statistics of a compression policy label

Navigate to Optimization > HTTP Compression > Policies> select a policy label, and click Statistics.

Offloading HTTP Compression to the NetScaler ADC

Performing compression on a server can affect the server's performance. A NetScaler ADC placed in front of your web servers and configured for HTTP compression offloads compression of both static and dynamic content, saving server CPU cycles and resources.

You can offload compression from the Web servers in either of two ways:

- Disable compression on the web servers, enable the NetScaler Compression feature at a global level, and configure services for compression.
- Leave the compression feature enabled on the web servers and configure the NetScaler appliance to remove the "Accept-Encoding" header from all HTTP client requests. The servers then send uncompressed responses. The NetScaler ADC compresses the server responses before sending them to the clients.

Note: The second option does not work if the servers automatically compress all responses. The NetScaler ADC does not attempt to compress a response that is already compressed.

To offload HTTP compression by using the command line interface

Enable compression on each service whose responses you want to compress, and then set the servercmp parameter to OFF. At the command prompt, enter the following commands:

```
set service <service name> -CMP YES
```

Repeat this command for each service for which you want to enable compression.

```
show service <service name>
```

Repeat this command for each service, to verify that compression is enabled.

Save config

```
set cmp parameter "serverCmp OFF"
```

Integrated Caching

The integrated cache provides in-memory storage on the Citrix NetScaler appliance and serves Web content to users without requiring a round trip to an origin server. For static content, the integrated cache requires little initial setup. After you enable the integrated cache feature and perform basic setup (for example, determining the amount of NetScaler appliance memory the cache is permitted to use), the integrated cache uses built-in policies to store and serve specific types of static content, including simple Web pages and image files. You can also configure the integrated cache to store and serve dynamic content that is usually marked as non-cacheable by Web and application servers (for example, database records and stock quotes).

When a request or response matches the rule (logical expression) specified in a built-in policy or a policy that you have created, the NetScaler appliance performs the action associated with the policy. By default, all policies store cached objects in and retrieve them from the Default content group, but you can create your own content groups for different types of content.

To enable the NetScaler appliance to find cached objects in a content group, you can configure selectors, which match cached objects against expressions, or you can specify parameters for finding objects in the content group. If you use selectors (which Citrix recommends), configure them first, so that you can specify selectors when you configure content groups. Next, set up any content groups that you want to add, so that they are available when you configure the policies. To complete the initial configuration, create policy banks by binding each policy to a global bind point or a virtual server, or to a label that can be called from other policy banks.

You can tune the performance of the integrated cache, using methods such as pre-loading cached objects before they are scheduled to expire. To manage the handling of cached data once it leaves the NetScaler appliance, you can configure caching-related headers that are inserted into responses. The integrated cache can also act as a forward proxy for other cache servers.

Note: Integrated caching requires some familiarity with HTTP requests and responses. For information about the structure of HTTP data, see *Live HTTP Headers* at "<http://livehttpheaders.mozdev.org/>."

How the Integrated Cache Works

The integrated cache monitors HTTP and SQL requests that flow through the Citrix NetScaler appliance and compares the requests with stored policies. Depending on the outcome, the integrated cache feature either searches the cache for the response or forwards the request to the origin server. For HTTP requests, the integrated cache feature can also serve partial content from the cache in response to single byte-range and multi-part byte-range requests.

Cached data can be compressed if the client accepts compressed content. You can configure expiration times for a content group, and you can selectively expire entries in a content group.

Data that is served from the integrated cache is a cache hit, and data served from the origin is a cache miss, as described in the following table.

Table 1. Cache Hits and Misses

Transaction Type	Specifies
Cache Hit	<p>Responses that the NetScaler appliance serves from the cache, including:</p> <ul style="list-style-type: none">Static objects, for example, image files and static Web pages200 OK pages203 Non-Authoritative Response pages300 Multiple Choices pages301 Moved Permanently pages302 Found pages304 Not Modified pages <p>These responses are known as positive responses.</p> <p>The NetScaler appliance also caches the following negative responses:</p> <ul style="list-style-type: none">307 Temporary Redirect pages403 Forbidden pages404 Not Found pages410 Gone pages <p>To further improve performance, you can configure the NetScaler appliance to cache additional types of content.</p>
Storable Cache Miss	<p>For a storable cache miss, the NetScaler appliance fetches the response from the origin server, and stores the response in the cache before serving it to the client.</p>
Non-Storable Cache Miss	<p>A non-storable cache miss is inappropriate for caching. By default, any response that contains the following status codes is a non-storable cache miss:</p> <ul style="list-style-type: none">201, 202, 204, 205, 206 status codesAll 4xx codes, except 403, 404 and 4105xx status codes

Note: To integrate dynamic caching with your application infrastructure, use the NITRO API to issue cache commands remotely. For example, you can configure triggers that expire cached responses when a database table is updated.

To ensure the synchronization of cached responses with the data on the origin server, you configure expiration methods. When the NetScaler appliance receives a request that matches an expired response, it refreshes the response from the origin server.

Note: Citrix recommends that you synchronize the times on the NetScaler appliance and the back-end server(s).

Example of Dynamic Caching

Dynamic caching evaluates HTTP requests and responses based on parameter-value pairs, strings, string patterns, or other data. For example, suppose that a user searches for Bug 31231 in a bug reporting application. The browser sends the following request on the user's behalf:

```
GET /mybugreportingsystem/mybugreport.dll?IssuePage&RecordId=31231&Template=view&TableId=100
Host: mycompany.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9) Gecko/2008052906 Firefox
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
. . .
```

In this example, GET requests for this bug reporting application always contain the following parameters:

- IssuePage
- RecordID
- Template
- TableId

GET requests do not update or alter the data, so you can configure these parameters in caching policies and selectors, as follows:

- You configure a caching policy that looks for the string mybugreportingsystem and the GET method in HTTP requests. This policy directs matching requests to a content group for bugs.
- In the content group for bugs, you configure a hit selector that matches various parameter-value pairs, including IssuePage, RecordID, and so on.

Note that a browser can send multiple GET requests based on one user action. The following is a series of three separate GET requests that a browser issues when a user searches for a bug based on a bug ID.

```
GET /mybugreportingsystem/mybugreport.dll?IssuePage&RecordId=31231&Template=view&TableId=100
GET /mybugreportingsystem/mybugreport.dll?IssuePage&Template=viewbtns&RecordId=31231&TableId
GET /mybugreportingsystem/mybugreport.dll?IssuePage&Template=viewbody&RecordId=31231&tableid
```

To fulfill these requests, multiple responses are sent to the user's browser, and the Web page that the user sees is an assembly of the responses.

If a user updates a bug report, the corresponding responses in the cache should be refreshed with data from the origin server. The bug reporting application issues HTTP POST requests when a user updates a bug report. In this example, you configure the following to ensure that POST requests trigger invalidation in the cache:

- A request-time invalidation policy that looks for the string mybugreportingsystem and the POST HTTP request method, and directs matching requests to the content group for bug reports.
- An invalidation selector for the content group for bug reports that expires cached content based on the RecordID parameter. This parameter appears in all of the responses, so the invalidation selector can expire all relevant items in the cache.

The following excerpt shows a POST request that updates the sample bug report.

```
POST /mybugreportingsystem/mybugreport.dll?TransitionForm HTTP/1.1\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0) Opera 7.23 [en]\r\n
Host: mybugreportingsystem\r\n
Cookie: ttSearch.134=%23options%3Afalse%23active%23owner%3Afalse%23unowned%3Afalse%23submitte
Cookie2: $Version=1\r\n
. . .
\r\n
ProjectId=2&RecordId=31231&TableId=1000&TransitionId=1&Action=Update&CopyProjectId=0&ReloadF
```

When the Citrix NetScaler appliance receives this request, it does the following:

- Matches the request with an invalidation policy.
- Finds the content group that is named in the policy.
- Applies the invalidation selector for this content group and expires all responses that match RecordID=31231.

When a user issues a new request for this bug report, the NetScaler appliance goes to the origin server for updated copies of all the responses that are associated with the report instance, stores the responses in the content group, and serves them to the user's browser, which reassembles the report and displays it.

Setting Up the Integrated Cache

To use the integrated cache, you must install the license and enable the feature. After you enable the integrated cache, the Citrix® NetScaler® appliance automatically caches static objects as specified by built-in policies and generates statistics on cache behavior. (Built-in policies have an underscore in the initial position of the policy name.)

Even if the built-in policies are adequate for your situation, you might want to modify the global attributes. For example, you might want to modify the amount of NetScaler appliance memory allocated to the integrated cache.

If you would like to observe cache operation before changing settings, see "[Displaying Cached Objects and Cache Statistics](#)."

Note: The NetScaler cache is an in-memory store that is purged when you restart the appliance. This section includes the following details:

- [Installing the Integrated Cache License](#)
- [Enabling Integrated Caching](#)
- [Configuring Global Attributes for Caching](#)
- [Built-in Content Group, Pattern Set, and Policies for the Integrated Cache](#)

Installing the Integrated Cache License

Updated: 2013-10-28

An integrated cache license is required. For information about licenses, see information about obtaining NetScaler licenses at "<http://support.citrix.com/article/ctx121062>."

To install the license for the Integrated Caching feature

1. Obtain a license code from Citrix, go to the command line interface, and log in.
2. At the command line interface, copy the license file to the /nsconfig/license folder.
3. Reboot the NetScaler appliance by using the following command:

```
reboot
```

Enabling Integrated Caching

Updated: 2015-05-20

When you enable integrated caching, the NetScaler appliance begins caching server responses. If you have not configured any policies or content groups, the built in policies store cached objects in the Default content group.

To enable integrated caching by using the command line interface

At the command prompt, type one of the following commands to enable or disable integrated caching:

```
enable ns feature IC
```

To enable integrated caching by using the configuration utility

Navigate to System > Settings, click Configure Basic Features, and select Integrated Caching.

Configuring Global Attributes for Caching

Updated: 2014-08-08

Global attributes apply to all cached data. You can specify the amount of NetScaler memory allocated to the integrated cache, Via header insertion, a criterion for verifying that a cached object should be served, the maximum length of a POST body permitted in the cache, whether to bypass policy evaluation for HTTP GET requests, and an action to take when a policy cannot be evaluated.

The cache memory capacity is limited only by the memory of the hardware appliance. Also, any packet engine (the central distribution hub of all incoming TCP requests) in the nCore NetScaler appliance is aware of objects cached by other packet engines in the nCore NetScaler appliance.

Note that the default global memory limit is 0. Therefore, even if Integrated Caching is enabled, the NetScaler appliance does not cache any objects. You must explicitly set the global memory limit when integrated caching is enabled.

You can modify the global memory limit configured for caching objects. However, when you update the global memory limit to a value lower than the existing value (for example, from 10 GB to 4 GB), if a higher amount of memory (greater than 4 GB) is already being used to cache objects, the NetScaler continues using that amount of memory.

This means that even though the integrated caching limit is configured to some value, the actual limit used can be higher. This excessive memory is however released when the objects are removed from cache.

The output of the show cache parameter command indicates the configured value (Memory Usage limit) and the actual value being used (Memory usage limit (active value)).

To configure global settings for caching by using the command line interface

At the command prompt, type:

```
set cache parameter [-memLimit <MBytes>] [-via <string>] [-verifyUsing <criterion>] [-maxPostLen <positiveInteger>] [-prefetchMaxPending <positiveInteger>] [-enableBypass (YES|NO)] [-undefAction (NOCACHE|RESET)]
```

To configure global settings for caching by using the configuration utility

Navigate to Optimization > Integrated Caching, click Change Cache Settings, and configure the global settings for caching.

Built-in Content Group, Pattern Set, and Policies for the Integrated Cache

Updated: 2013-08-23

The Citrix NetScaler appliance includes a built-in integrated caching configuration that you can use for caching content. The configuration consists of a content group called `ctx_cg_poc`, a pattern set called `ctx_file_extensions`, and a set of integrated cache policies. In the content group `ctx_cg_poc`, only objects that are 500 KB or smaller are cached. The content is cached for 86000 seconds, and the memory limit for the content group is 512 MB. The pattern set is an indexed array of common file extensions for file-type matching.

The following table lists the built-in integrated caching policies. By default, the policies are not bound to any bind point. You must bind them to a bind point if you want the NetScaler appliance to evaluate traffic against the policies. The policies cache objects in the `ctx_cg_poc` content group.

Table 1. Built-in Integrated Caching Policies

Integrated caching policy name	Policy rule
<code>ctx_images</code>	<code>HTTP.REQ.URL.SET_TEXT_MODE(IGNORECASE).CONTAINS_INDEX(\"ctx_file. (101,150)</code>
<code>ctx_web_css</code>	<code>HTTP.REQ.URL.ENDSWITH(\".css\")</code>
<code>ctx_doc_pdf</code>	<code>HTTP.REQ.URL.ENDSWITH(\".pdf\")</code>
<code>ctx_web_JavaScript</code>	<code>HTTP.REQ.URL.ENDSWITH(\".js\")</code>
<code>ctx_web_JavaScript-Res</code>	<code>HTTP.RES.HEADER(\"Content-Type\").CONTAINS(\"application/x-javas</code>
<code>ctx_NOCACHE_Cleanup</code>	<code>TRUE</code>

--	--	--

Configuring Selectors and Basic Content Groups

You can configure selectors and apply them to content groups. When you add a selector to one or more content groups, you specify whether the selector is to be used for identifying cache hits or identifying cached objects to be invalidated (expired). Selectors are optional. Alternatively, you can configure content groups to use hit parameters and invalidation parameters. However, Citrix recommends that you configure selectors.

After configuring selectors, or deciding to use parameters instead, you are ready to set up a basic content group. After creating the basic content group, you need to decide how objects should be expired from the cache, and configure cache expiration. You can further modify the cache as described in ["Improving Cache Performance"](#) and ["Configuring Cookies, Headers, and Polling"](#), but you might first want to configure caching policies.

Note: Content group parameters and selectors are used only at request time, and you typically associate them with policies that use MAY_CACHE or MAY_NO_CACHE actions.

Advantages of Selectors

A selector is a filter that locates particular objects in a content group. If you do not configure a selector, the Citrix® NetScaler® appliance looks for an exact match in the content group. This can lead to multiple copies of the same object residing in a content group. For example, a content group that does not have a selector may need to store URLs for host1.domain.com\mypage.htm, host2.domain.com\mypage.htm, and host3.domain.com\mypage.htm. In contrast, a selector can match just the URL (mypage.html, using the expression `http.req.url`) and the domain (.com, using the expression `http.req.hostname.domain`), allowing the requests to be satisfied by the same URL.

Selector expressions can perform simple matching of parameters (for example, to find objects that match a few query string parameters and their values). A selector expression can use Boolean logic, arithmetic operations, and combinations of attributes to identify objects (for example, segments of a URL stem, a query string, a string in a POST request body, a string in an HTTP header, a cookie). Selectors can also perform programmatic functions to analyze information in a request. For example, a selector can extract text in a POST body, convert the text into a list, and extract a specific item from the list.

For more information about expressions and what you can specify in an expression, see ["Policies and Expressions."](#)

Using Parameters Instead of Selectors

Although Citrix recommends the use of selectors with a content group, you can instead configure hit parameters and invalidation parameters. For example, suppose that you configure three hit parameters in a content group for bug reports: BugID, Issuer, and Assignee. If a request contains BugID=456, with Issuer=RohitV and Assignee=Robert, the NetScaler appliance can serve responses that match these parameter-value pairs.

Invalidation parameters in a content group expire cached entries. For example, suppose that BugID is an invalidation parameter and a user issues a POST request to update a bug report. An invalidation policy directs the request to this content group, and the invalidation parameter for the content group expires all cached responses that match the BugID value. (The next time a user issues a GET request for this report, a caching policy can enable the NetScaler appliance to refresh the cached entry for the report from the origin server.)

Note that the same parameter can be used as a hit parameter or an invalidation parameter.

Content groups extract request parameters in the following order:

- URL query
- POST body
- Cookie header

After the first occurrence of a parameter, regardless of where it occurred in the request, all its subsequent occurrences are ignored. For example, if a parameter exists both in the URL query and in the POST body, only the one in the URL query is considered.

If you decide to use hit and invalidation parameters for a content group, configure the parameters when you configure the content group.

Note: Citrix recommends that you use selectors rather than parameterized content groups, because selectors are more flexible and can be adapted to more types of data.

Configuring a Selector

A content group can use a hit selector to retrieve cache hits or use an invalidation selector to expired cached objects and fetch new ones from the origin server.

A selector contains a name and a logical expression, called an *advanced expression*.

For more information about advanced expressions, see ["Policies and Expressions."](#)

To configure a selector, you assign it a name and enter one or more expressions. As a best practice, a selector expression should include the URL stem and host, unless there is a strong reason to omit them.

To configure a selector by using the command line interface

At the command prompt, type:

```
add cache selector <selectorName> ( <rule> ... )
```

For information about configuring the expression or expressions, see ["To configure a selector expression by using the command line interface."](#)

Examples

```
>add cache selector product_selector "http.req.url.query.value(\"ProductId\")" "http.req.url.  
> add cache selector batch_selector "http.req.url.query.value(\"ProductId\")" "http.req.url.  
> add cache selector product_id_selector "http.req.url.query.value(\"ProductId\")"  
> add cache selector batchnum_selector "http.req.url.query.value(\"BatchNum\")" "http.req.ur  
> add cache selector batchid_selector "http.req.url.query.value(\"depotLocation\")" "http.re
```

To configure a selector by using the configuration utility

Navigate to Optimization > Integrated Caching > Cache Selectors, and add the cache selector.

About Content Groups

A content group is a container for cached objects that can be served in a response. When you first enable the integrated cache, cacheable objects are stored in a content group named Default. You can create new content groups that have unique properties. For example, you can define separate content groups for image data, bug reports, and stock quotes, and you can configure the stock quote content group to be refreshed more often than the other groups.

You can configure expiration of an entire content group or selected entries in a content group.

The data in a content group can be static or dynamic, as follows:

- o **Static content groups.** Finds an exact match between the URL stem and host name on the request and the URL stem and host name of the response.
- o **Dynamic content groups.** Looks for objects that contains particular parameter-value pairs, arbitrary strings or string patterns. Dynamic content groups are useful when caching data that is updated frequently (for example, a bug report or a stock quote).

Process overview: Serving a hit from a content group

1. A user enters search criteria for an item, such as a bug report, and clicks the Find button in an HTML form.
2. The browser issues one or more HTTP GET requests. These requests contain parameters (for example, the bug owner, bug ID, and so on).
3. When the NetScaler appliance receives the requests, it searches for a matching policy, and if it finds a caching policy that matches these requests, it directs the requests to a content group.
4. The content group looks for appropriate objects in the content group, usually based on criteria that you configure in a selector.

For example, the content group can retrieve responses that match NameField=username and BugID=ID.

5. If it finds matching objects, the NetScaler appliance can serve them to the user's browser, where they are assembled into a complete response (for example, a bug report).

Example: Invalidating an object in a content group

1. A user modifies data (for example, the user modifies the bug report and clicks the Submit button).
2. The browser sends this data in the form of one or more HTTP requests. For example, it can send a bug report in the form of several HTTP POST requests that contain information about the bug owner and bug ID.
3. The NetScaler appliance matches the requests against invalidation policies. Typically, these policies are configured to detect the HTTP POST method.
4. If the request matches an invalidation policy, the NetScaler appliance searches the content group that is associated with this policy, and expires responses that match the configured criteria for invalidation.

For example, an invalidation selector can find responses that match NameField=username and BugID=ID.

5. The next time the NetScaler appliance receives a GET request for these responses, it fetches refreshed versions from the origin server, caches the refreshed responses, and serves these responses to the user's browser, where they are assembled into a complete bug report.

Setting Up a Basic Content Group

By default, all cached data is stored in the default content group. You can configure additional content groups and specify these content groups in one or more policies.

You can configure content groups for static content, and you must configure content groups for dynamic content. You can modify the configuration of any content group, including the default group.

To set up a basic content group by using the command line interface

At the command prompt, type:

```
add cache contentgroup <name> (-hitSelector <hitSelectorName> -invalSelector <invalidationSelectorName> | -hitParams <hitParamName> -invalParams <invalidationParamName>) -type <type> [-relExpiry <sec> | -relExpiryMilliSec <msec>] [-heurExpiryParam <positiveInteger>]
```

Examples

```
> add cache contentgroup Products_Details "hitSelector product_selector "invalSelector i
> add cache contentgroup bugrep -hitParams IssuePage RecordID Template TableId -invalParams
```

To set up a basic content group by using the configuration utility

Navigate to Optimization > Integrated Caching > Content Groups, and create the content group.

Expiring or Flushing Cached Objects

If a response does not have an Expires header or a Cache-Control header with an expiration time (Max-Age or Smax-Age), you must expire objects in a content group by using one of the following methods:

- Configure content group expiration settings to determine whether and how long to keep the object.
- Configure an invalidation policy and action for the content group. For more information, see "[Configuring Policies for Caching and Invalidation](#)."
- Expire the content group or objects within it manually.

After a cached response expires, the NetScaler appliance refreshes it the next time the client issues a request for the response. By default, when the cache is full, the NetScaler appliance replaces the least recently used response first.

The following list describes methods for expiring cached responses using settings for a content group. Typically, these methods are specified as a percent or in seconds:

- **Manual.** Manually invalidate all responses in a content group or all responses in the cache.
- **Response-based.** Specific expiration intervals for positive and negative responses. Response-based expiry is considered only if the Last-Modified header is missing in the response.
- **Heuristic expiry.** For responses that have a Last-Modified header, heuristic expiry is a percentage of the time since the response was modified (calculated as current time minus the Last-Modified time, multiplied by the heuristic expiry value). For example, if a Last-Modified header indicates that a response was updated 2 hours ago, and the heuristic expiry setting is 10%, cached objects expire after 0.2 hours. This method assumes that frequently updated responses need to be expired more often.
- **Absolute or relative.** Specify an exact (absolute) time when the response expires every day, in HH:MM format, local time or GMT. Local time may not work in all time zones.

Relative expiration specifies a number of seconds or milliseconds from the time a cache miss causes a trip to the origin server to the expiration of the response. If you specify relative expiration in milliseconds, enter a multiple of 10. This form of expiration works for all positive responses. Last-Modified, Expires, and Cache-Control headers in the response are ignored.

Absolute and relative expiration override any expiration information in the response itself.

- **On download.** The option Expire After Complete Response Received expires a response as soon as it is downloaded. This is useful for frequently updated responses, for example, stock quotes. By default, this option is disabled.

Enabling both Flash Cache and Expire After Complete Response Received accelerates the performance of dynamic applications. When you enable both options, the NetScaler appliance fetches only one response for a block of simultaneous requests.

For more information, see "[Queuing Requests to the Cache](#)."

- **Pinned.** By default, when the cache is full the NetScaler appliance replaces the least recently used response first. The NetScaler appliance does not apply this behavior to content groups that are marked as pinned.

If you do not configure expiration settings for a content group, the following are additional options for expiring objects in the group:

- Configure a policy with an INVALID action that applies to the content group.
- Enter the names of content groups when configuring a policy that uses an INVALID action.

How Expiration Methods Are Applied

Expiration works differently for positive and negative responses. Positive and negative responses are described in the table, *Expiration of Positive and Negative Responses* mentioned below.

The following are rules of thumb for understanding the expiration method that is applied to a content group:

- You can control whether the NetScaler appliance evaluates response headers when deciding whether to expire an object.

- o Absolute and relative expiration cause the NetScaler appliance to ignore the response headers (they override any expiration information in the response).
- o Heuristic expiration settings and "Weak Positive" and "Weak Negative" expiration (labeled as **Default** values in the configuration utility) cause the NetScaler appliance to examine the response headers. These settings work together as follows:
 - The value in an Expires or Cache-Control header overrides these content group settings.
 - For positive responses that lack an Expires or Cache-Control header but have a Last-Modified header, the NetScaler appliance compares heuristic expiration settings with the header value.
 - For positive responses that lack an Expires, Cache-Control, or Last-Modified header, NetScaler appliance uses the "weak positive" value.
 - For negative responses that lack an Expires or Cache-Control header, NetScaler appliance uses the "weak negative" value.

For a list of expiration parameters see, "Configuring Periodic Expiration of a Content Group." The following table describes how these methods are applied.

Table 1. Expiration of Positive and Negative Responses

Response Type	Expiration Header Type	Content Group Setting	Period the Object Remains in the Cache
Positive	any header	Expire Content After (relExpiry) with no other settings	Use the value of the Expire Content After setting.
Positive	any header	Expire Content At (absExpiry) with no other settings	Subtract current date from the value of the Expire Content At setting.
Positive	any header	Expire Content After (relExpiry) and Expire content at (absExpiry)	Use the smaller of the two values for the content group settings. See the previous rows in this table.
Positive	Last-Modified (with any other headers)	Heuristic (heurExpiry Param) with any other setting	Subtract the Last-Modified date from the current date, multiply the result by the value of the heuristic expiry setting, and then divide by 100.
Positive	Last-Modified (with any other headers)	Default (positive) (weakPosRel Expiry) and no other setting	Use the value of the Default (positive) expiry setting.
Positive	Expires or Cache-Control: Max-Age header is present Last-Modified header is absent	Heuristic (heurExpiry Param), Default (positive) (weakPosRel Expiry), or both	Subtract the current date from the Expires or the Cache-Control:Max-Age date.
Positive	no caching headers	Default (positive) (weakPosRel Expiry) and any other expiration setting.	Use the value of the Default (positive) setting.
Positive	no caching headers	Heuristic (heurExpiry Param) is present	If the Last-Modified header is absent, the response is not cached or it is cached with an Already Expired status.

		Default (positive) (weakPosRel Expiry) setting is absent	If the Last-Modified header is present, use the heuristic expiry value.
Negative	Expires or Cache-Control:Max-Age	Expire Content After (relExpiry), Expire Content At (absExpiry), or both settings	Subtract the current date from the value of the Expires header, or use the value of the Cache-Control:Max-Age header.
Negative	Expires or Cache-Control headers are absent	Expire Content After (relExpiry), Expire Content At (absExpiry), or both settings	Response is not cached, or is cached with an Already Expired status.
Negative	Expires or Cache-Control:Max-Age	Any setting	Subtract the current date from the Expires or Cache-Control:Max-Age date.
Negative	Expires and Cache-Control:Max-Age headers are absent	Default (negative) (weakNegRel Expiry)	Use the value of the Default (negative) setting.
Negative	Expires and Cache-Control:Max-Age headers are absent	Any setting other than Default (negative) (weakNegRel Expiry)	Object is not cached or is cached with an Already Expired status.

Expiring a Content Group Manually

You can manually expire all of the entries in a content group.

To manually expire all responses in a content group by using the command line interface

At the command prompt, type:

```
expire cache contentGroup <name>
```

To manually expire all responses in a content group by using the configuration utility

Navigate to Optimization > Integrated Caching > Content Groups, select the content group, and click Invalidate to expire all the responses in a content group.

To manually expire all responses in the cache by using the configuration utility

Navigate to Optimization > Integrated Caching > Content Groups, and click Invalidate All to expire all the responses in cache.

Configuring Periodic Expiration of a Content Group

You can configure a content group so that it performs selective or full expiration of its entries. The expiration interval can be fixed or relative.

To configure content group expiration by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> (-relExpiry|-relExpiryMilliSec|-absExpiry|-absExpiryGMT| -heurExpiryParam|-weakPosRelExpiry|-weakNegRelExpiry| -expireAtLastBye) <expirationValue>
```

To configure content group expiration by using the configuration utility

Navigate to **Optimization > Integrated Caching > Content Groups**, select the content group, and specify expiry method.

Expiring Individual Responses

Expiring a response forces the NetScaler appliance to fetch a refreshed copy from the origin server. Responses that do not have validators, for example, ETag or Last-Modified headers, cannot be revalidated. As a result, flushing these responses has the same effect as expiring them.

To expire a cached response in a content group for static data, you can specify a URL that must match the stored URL. If the cached response is part of a parameterized content group, you must specify the group name as well as the exact URL stem. The host name and the port number must be the same as in the host HTTP request header of the cached response. If the port is not specified, port 80 is assumed.

To expire individual responses in a content group by using the command line interface

At the command prompt, type:

```
expire cache object -url <URL> -host <hostName> [-port <port>] [-groupName <contentGroupName>] [-httpMethod GET|POST]
```

To expire individual responses in a content group by using the command line interface (selector-based)

At the command prompt, type the following command:

```
expire cache object -locator <positiveInteger>
```

To expire a cached response by using the configuration utility

Navigate to **Optimization > Integrated Caching > Cached Objects**, select the cached response, and expire.

To expire a response by using the Lookup tool (selector-based)

Navigate to **Optimization > Integrated Caching > Cached Objects**, click **Search** and set the search criteria to find the required cached response and expire.

Flushing Responses in a Content Group

You can remove, or flush, all responses in a content group, some responses in a group, or all responses in the cache. Flushing a cached response frees up memory for new cached responses.

Note: To flush responses for more than one object at a time, use the configuration utility method. The command line interface does not offer this option.

To flush responses from a content group by using the command line interface

At the command prompt, type:

```
flush cache contentGroup <name> [-query <queryString> | [-selectorValue <selectorExpressionIDList> -host <hostName>]]
```

To flush responses from a content group by using the configuration utility

1. Navigate to **Optimization > Integrated Caching > Content Groups**.
2. In details pane, flush the responses as follows:
 - To flush all responses in all content groups, click **Invalidate All**, and flush all the responses.

- o To flush responses in a particular content group, select the content group, click Invalidate, and flush all the responses.

Note: If this content group uses a selector, you can selectively flush responses by entering a string in the Selector value text box, entering a host name in the Host text box. Then click Flush and OK. The Selector value can be a query string of up to 2319 characters that is used for parameterized invalidation.

If the content group uses an invalidation parameter, you can selectively flush responses by entering a string in the Query field.

If the content group uses an invalidation parameter and Invalidate objects belonging to target host is configured, enter strings in the Query and Host fields.

To flush a cached response by using the command line interface

At the command prompt, type:

```
flush cache object -locator <positiveInteger> | -url <URL> -host <hostName> [-port <port>] [-groupName <contentGroupName>] [-httpMethod GET|POST]
```

To flush a cached response by using the configuration utility

Navigate to Optimization > Integrated Caching > Cached Objects, select the cached object, and flush.

Deleting a Content Group

You can remove a content group if it is not used by any policy that stores responses in the cache. If the content group is bound to a policy, you must first remove the policy. Removing the content group removes all responses stored in that group.

You cannot remove the Default, BASEFILE, or DeltaJS group. The Default group stores cached responses that do not belong in any other content group.

To delete a content group by using the command line interface

At the command prompt, type:

```
rm cache contentgroup<name>
```

To delete a content group by using the configuration utility

Navigate to Optimization > Integrated Caching > Content Groups, select the content group, and delete.

Configuring Policies for Caching and Invalidation

Policies enable the integrated cache to determine whether to try to serve a response from the cache or the origin. The Citrix NetScaler appliance provides built-in policies for integrated caching, and you can configure additional policies. When you configure a policy, you associate it with an action. An action either caches the objects to which the policy applies or invalidates (expires) the objects. Typically, you based caching policies on information in GET and POST requests. You typically base invalidation policies on the presence of the POST method in requests, along with other information. You can use any information in a GET or POST request in a caching or an invalidation policy.

You can view some of the built-in policies in the integrated cache's Policies node in the configuration utility. The built-in policy names begin with an underscore (_).

Actions determine what the NetScaler appliance does when traffic matches a policy. The following actions are available:

- **Caching actions.** Policies that you associate with the CACHE action store responses in the cache and serve them from the cache.
- **Invalidation actions.** Policies that you associate with the INVALID action immediately expire cached responses and refresh them from the origin server. Note that for Web-based applications, invalidation policies often evaluate POST requests.
- **Do not cache actions** Policies that you associate with a NOCACHE action never store objects in the cache.
- **Provisionally cache actions** Policies that you associate with a MAYCACHE or MAYNOCACHE action depend on the outcome of additional policy evaluations.

Although the integrated cache does not store objects specified by the LOCK method, you can invalidate cached objects upon receipt of a LOCK request. For invalidation policies only, you can specify LOCK as a method by using the expression `http.req.method.eq("lock")`. Unlike policies for GET and POST requests, you must enclose the LOCK method in quotes because the NetScaler appliance recognizes this method name as a string only.

After you create a policy, you bind it to a particular point in the overall processing of requests and responses. Although you create a policy before binding it, you should understand how the bind points affect the order of processing before you create your policies.

The policies bound to a particular bind point constitute a policy bank. You can use goto expressions to modify the order of execution in a policy bank. You can also invoke policies in other policy banks. In addition, you can create labels and bind policies to them. Such a label is not associated with a processing point, but the policies bound to it can be invoked from other policy banks.

Actions to Associate with Integrated Caching Policies

The following table describes actions for integrated caching policies.

Table 1. Actions That You Can Associate with an Integrated Caching Policy

Action	Specifies
CACHE	<p>Serves a response from the cache if the response has not expired. If the response must be fetched from the origin server, the NetScaler appliance caches the response before serving it.</p> <p>Even data that is updated and accessed frequently can be cached. For example, stock quotes are updated frequently, but they can be cached so that they can be served quickly to multiple users. If necessary, cached data can be refreshed immediately after it is downloaded.</p> <p>A CACHE action can be overridden by built-in policies.</p>
NOCACHE	<p>Always fetches the response from the origin server and marks the response as non-storable.</p> <p>You typically configure NOCACHE policies for data that is sensitive or personalized.</p>
MAY_CACHE	<p>Used in a request-time policy, this setting provisionally enables a response to be stored in a content group, pending evaluation of response-time policies. The following are possible:</p> <ul style="list-style-type: none"> ○ If a matching response-time policy has a CACHE action but does not specify a content group, the response is stored in the Default group unless built-in policies override this policy. ○ If a matching response-time policy has a CACHE action and specifies the same content group as the one in the request-time policy, the response is stored in the named content group unless built-in policies override this policy. ○ If a matching response-time policy has a CACHE action but specifies a different content group from the one in the request-time policy, a NOCACHE action is applied. ○ If a matching response-time policy has a NOCACHE action, perform a NOCACHE action. ○ If there is no matching response-time policy, a CACHE action is applied, unless a built-in policy overrides this policy.
MAY_NOCACHE	<p>For a request-time policy, this setting provisionally prevents caching the response. At response time, one of following actions is taken:</p> <ul style="list-style-type: none"> ○ If no response-time policy matches the request, the final action is NOCACHE. ○ If a matching response-time policy contains a CACHE action, the final action is CACHE, unless built-in policies override this policy. ○ If a matching response-time policy contains a NOCACHE action, the final action is NOCACHE. ○ If a matching response-time policy has a CACHE action but does not specify a content group, the final action is to CACHE the response in the Default content group, unless built-in policies override this policy.
INVAL	<p>Expires cached responses. Depending on how the policy and the content group are configured, all responses in one or more content groups are expired, or selected objects in the content group are expired.</p> <p>Note: You can specify INVAL actions in request-time policies only.</p>

Bind Points for a Policy

You can bind the policy to one of the following bind points:

- **A global policy bank.** These are the request-time default, request-time override, response-time default, and response-time override policy banks, as described in ["Order of Policy Evaluation."](#)
- **A virtual server.** Policies that you bind to a virtual server are processed after the global override policies and before the global default policies, as described in ["Order of Policy Evaluation."](#) Note that when binding a policy to a virtual server, you bind it to either request-time or response-time processing.
- **An ad-hoc policy label.** A policy label is a name assigned to a policy bank. In addition to the global labels, the integrated cache has two built-in custom policy labels:
 - _reqBuiltinDefaults.** This policy label, by default, is invoked from the request-time default policy bank.
 - _resBuiltinDefaults.** This policy label, by default, is invoked from the response-time default policy bank.

You can also define new policy labels. Policies bound to a user-defined policy label must be invoked from within a policy bank for one of the built-in bind points. For more information about creating a policy label, see ["Configuring a Policy Label in the Integrated Cache."](#) For more information about policy label invocation, see ["Configuring a Policy Bank for Caching."](#)

Important: You should bind a policy with an INVAL action to a request-time override or a response-time override bind point. To delete a policy, you must first unbind it.

Order of Policy Evaluation

For an advanced policy to take effect, you must ensure that the policy is invoked at some point during the NetScaler appliance's processing of traffic. To specify the invocation time, you associate the policy with a bind point. The following are the bind points, listed in order of evaluation:

- **Request-time override.** If a request matches a request-time override policy, by default request-time policy evaluation ends and the NetScaler appliance stores the action that is associated with the matching policy.
- **Request-time load balancing virtual server.** If policy evaluation cannot be completed after all the request-time override policies are evaluated, the NetScaler appliance processes request-time policies that are bound to load balancing virtual servers. If the request matches one of these policies, evaluation ends and the NetScaler appliance stores the action that is associated with the matching policy.
- **Request-time content switching virtual server.** Policies that are bound to this bind point are evaluated after request-time policies that are bound to load balancing virtual servers.
- **Request-time default.** If policy evaluation cannot be completed after all request-time, virtual server-specific policies are evaluated, the NetScaler appliance processes request-time default policies. If the request matches a request-time default policy, by default request-time policy evaluation ends and the NetScaler appliance stores the action that is associated with the matching policy.
- **Response-time override.** Similar to request-time override policy evaluation.
- **Response-time load balancing virtual server.** Similar to request-time virtual server policy evaluation.
- **Response-time content switching virtual server.** Similar to request-time virtual server policy evaluation.
- **Response-time default.** Similar to request-time default policy evaluation.

You can associate multiple policies with each bind point. To control the order of evaluation of the policies associated with the bind point you configure a priority level. In the absence of any other flow control information, policies are evaluated according to priority level, starting with the lowest numeric priority value.

After all integrated caching policies have been evaluated, if there are conflicting actions specified in request-time and response-time policies, the NetScaler appliance determines the final action as specified in the table, ["Actions That You Can Associate with an Integrated Caching Policy."](#)

Note: Request-time policies for POST data or cookie headers must be invoked during request-time override evaluation, because the built-in request-time policies in the integrated cache return a NOCACHE action for POST requests and a MAY_NOCACHE action for requests with cookies. Note that you would associate MAY_CACHE or MAY_NOCACHE actions with a request-time policy that points to a parameterized content group. The response time policy determines whether the transaction is stored in the cache.

Configuring a Policy in the Integrated Cache

You configure new policies to handle data that the built-in policies cannot process. You configure separate policies for caching, preventing caching from occurring, and for invalidating cached data. Following are the main components of a policy for integrated caching:

- Rule: A logical expression that evaluates an HTTP request or response.
- Action: You associate a policy with an action to determine what to do with a request or response that matches the policy rule.
- Content groups: You associate the policy with one or more content groups to identify where the action is to be performed.

To configure a policy for caching by using the command line interface

At the command prompt, type:

```
add cache policy <policyName> -rule <expression> -action CACHE|MAY_CACHE|NOCACHE|MAY_NOCACHE [-storeInGroup <contentGroupName>] [-undefAction NOCACHE|RESET]
```

Examples

```
> add cache policy image_cache -rule "http.req.url.contains(\"jpg\") || http.req.url.contains(\".png\")" -action MAY_CACHE
> add cache policy bugReportPolicy -rule "http.req.url.query.contains(\"IssuePage\")" -action MAY_NOCACHE
> add cache policy my_form_policy -rule "http.req.header(\"Host\")contains(\"my.company.com\")" -action MAY_CACHE
> add cache policy viewproducts_policy -rule "http.req.url.contains(\"viewproducts.aspx\")" -action MAY_NOCACHE
```

To configure a policy for invalidation by using the command line interface

At the command prompt, type:

```
add cache policy <policyName> -rule <expression> -action INVALID [-invalObjects "<contentGroupName1>[,<selectorName1>"]. . ."] [-invalGroup <contentGroupName1>[,<contentGroupName2> . . .]] [-undefAction NOCACHE|RESET]
```

Examples

```
> add cache policy invalidation_events_policy -rule "http.req.header(\"Host\")contains(\"my.company.com\")" -action INVALID
> add cache policy inval_all -rule "http.req.method.eq(\"POST\") && http.req.url.contains(\"my.company.com\")" -action INVALID
> add cache policy bugReportInvalidationPolicy -rule "http.req.url.query.contains(\"TransitID\")" -action INVALID
> add cache policy editproducts_policy -rule "http.req.url.contains(\"editproducts.aspx\")" -action INVALID
```

To configure a policy for caching or invalidation by using the configuration utility

Navigate to Optimization > Integrated Caching > Policies, and create the new policy.

Globally Binding an Integrated Caching Policy

When you globally bind a policy, it is available to all virtual servers on the NetScaler appliance.

To bind an integrated caching policy globally by using the command line interface

At the command prompt, type:

```
bind cache global <policy> -priority <positiveInteger> [-type REQ_OVERRIDE|REQ_DEFAULT|RES_OVERRIDE|RES_DEFAULT] [-gotoPriorityExpression <expression>] [-invoke <labelType> <labelName>]
```

Example

```
> bind cache global myCachePolicy -priority 100 -type req_default
```

Note that the type argument is optional for globally bound policies, to maintain backward compatibility with policies that you defined using earlier versions of the NetScaler appliance. If you omit the type, the policy is bound to REQ_DEFAULT or RES_DEFAULT, depending on whether the policy rule is a response-time or a request-time expression. If the rule contains both request time and response time parameters, it is bound to RES_DEFAULT. Following is an example of a binding that omits the type.

```
> bind cache global myCache Policy 200
```

To bind an integrated caching policy globally by using the configuration utility

Navigate to Optimization > Integrated Caching, click Cache Policy Manager, and bind policies by specifying the relevant bind point and connection type (Request/Response).

Binding an Integrated Caching Policy to a Virtual Server

When you bind a policy to a virtual server, it is available only to requests and responses that match the policy and that flow through the relevant virtual server.

When using the configuration utility, you can bind the policy using the configuration dialog box for the virtual server. This enables you to view all of the policies from all NetScaler modules that are bound to this virtual server. You can also use the Policy Manager configuration dialog for the integrated cache. This enables you to view only the integrated caching policies that are bound to the virtual server.

To bind an integrated caching policy to a virtual server by using the command line interface

At the command prompt, type:

- `bind lb vserver <name>@ -policyName <policyName> -priority <positiveInteger> -type (REQUEST|RESPONSE)`
- `bind cs vserver <name>@ -policyName <policyName> -priority <positiveInteger> -type (REQUEST|RESPONSE)`

To bind an integrated caching policy to a virtual server by using the configuration utility (virtual server method)

- CS Virtual Server - Navigate to Traffic Management > Content Switching > Virtual Servers, select the virtual server, and bind relevant cache policies.
- LB Virtual Server - Navigate to Traffic Management > Load Balancing > Virtual Servers, select the virtual server, and bind relevant cache policies.

To bind an integrated caching policy to a virtual server by using the configuration utility (Policy Manager method)

Navigate to Optimization > Integrated Caching, click Cache Policy Manager, and bind cache policies by specifying the relevant bind point and connection type.

Note: You can bind cache policies to both LB virtual server and CS virtual server by selecting the appropriate bind point.

Example: Caching Compressed and Uncompressed Versions of a File

By default, a client that can handle compression can be served uncompressed responses or compressed responses in gzip, deflate, compress, and pack200-gzip format. If the client handles compression, an Accept-Encoding:compression format header is sent in the request. The compression type accepted by the client must match the compression type of the cached object. For example, a cached .gzip file cannot be served in response to a request with an Accept-Encoding: deflate header.

A client that cannot handle compression is served a cache miss if the cached response is compressed.

For dynamic caching, you need to configure two content groups, one for compressed data and one for uncompressed versions of the same data. The following is an example of configuring the selectors, content groups, and policies for serving uncompressed files from the cache to clients that cannot handle compression, and serving compressed versions of the same files to client that can handle compression.

```
add cache selector uncompressed_response_selector http.req.url "http.req.header(\"Host\")"
add cache contentGroup uncompressed_group -hitSelector uncompressed_responst_selector -inval
add cache policy cache_uncompressed -rule "HTTP.REQ.URL.CONTAINS(\"xyz\")" && !HTTP.REQ.HEADE
bind cache global cache_uncompressed -priority 100 -gotoPriorityExpression END -type REQ_OVE
add cache selector compressed_response_selector HTTP.REQ.URL "HTTP.REQ.HEADER(\"Host\")" "HT
add cache contentGroup compressed_group -hitSelector compressed_response_selector
add cache policy cache_compressed -rule "HTTP.REQ.URL.CONTAINS(\"xyz\")" && HTTP.REQ.HEADER(\
bind cache global cache_compressed -priority 200 -gotoPriorityExpression END -type REQ_OVERR
```

Configuring a Policy Bank for Caching

All of the policies that are associated with a particular bind point are collectively known as a policy bank. In addition to configuring priority levels for policies in a bank, you can modify the order of evaluation order in a bank by configuring Goto expressions. You can further modify the evaluation order by invoking an external policy bank from within the current policy bank. You can also configure new policy banks, to which you assign your own labels. Because such policy banks are not bound to any point in the processing cycle, they can be invoked only from within other policy banks. For convenience, policy banks whose labels do not correspond to a built-in bind point are called policy labels.

In addition to controlling order of policy evaluation by binding the policy and assigning a priority level, as described in "[Binding Policies That Use the Default Syntax](#)", you can establish the flow within a bank of policies by configuring a Goto expression. A Goto expression overrides the flow that is determined by the priority levels. You can also control the evaluation flow by invoking an external policy bank after evaluating an entry in the current bank. Evaluation always returns to the current bank after evaluation has completed for the external bank.

The following table summarizes the entries to control evaluation in a policy bank.

Table 1. Entries to Control Evaluation Flow in a Policy Bank

Attribute	Specifies
Name	<p>The name of a policy, or, to invoke another policy bank without evaluating the policy, the keyword NOPOLICY.</p> <p>You can specify NOPOLICY more than once in a policy bank, but you can specify a named policy only once.</p>
Priority	<p>An integer. The lower the integer, the higher the priority.</p>
Goto Expression	<p>Determines the next policy or policy bank to evaluate. You can provide one of the following values:</p> <ul style="list-style-type: none">◦ NEXT: Go to the policy with the next higher priority.◦ END: Stop evaluation.◦ USE_INVOCATION_RESULT: Applicable if this entry invokes another policy bank. If the final Goto in the invoked bank has a value of END, evaluation stops. If the final Goto is anything other than END, the current policy bank performs a NEXT.◦ Positive number: Priority number of the next policy to be evaluated.◦ Numeric expression: Expression that produces the priority number of the next policy to be evaluated. <p>The Goto can only proceed forward in a policy bank. Omitting the Goto expression is the same as specifying END.</p>
Invocation Type	<p>Designates a policy bank type. The value can be one of the following:</p> <ul style="list-style-type: none">◦ Request Vserver: Invokes request-time policies that are associated with a virtual server.◦ Response Vserver: Invokes response-time policies that are associated with a virtual server.◦ Policy label: Invokes another policy bank, as identified by the policy label for the bank.
Invocation Name	<p>Name of a virtual server or a policy label, depending on the value that you specified for the Invocation Type.</p>

The integrated cache has two built-in policy labels, and you can configure additional policy labels:

- **_reqBuiltInDefaults**: This policy label is invoked from the request-time default bind point.
- **_resBuiltInDefaults**: This policy label is invoked from the response-time default bind point.

Note: For information about creating policy labels, see "[Configuring a Policy Label in the Integrated Cache](#)."

To invoke a policy label in a caching policy bank by using the command line interface

At the command prompt, type:

```
bind cache policylabel <labelName> -policname<policyName> -priority<priority> [-gotoPriorityExpression <gotopriorityExpression>] [-invoke <labelType> <labelName>]
```

To invoke a policy label in a caching policy bank by using the configuration utility

1. Navigate to Optimization > Integrated Caching, click Cache policy manager, and specify the relevant bind point (Override Global or Default Global) and connection type to view the list of policies bound to this bind point.
2. If you want to invoke a policy label without evaluating a policy, click NOPOLICY.
Note: To invoke an external policy bank, click the field in the Invoke Type column, and select the type of policy bank that you want to invoke at this point in the policy bank. This can be a global label or a virtual server bank. In the Invoke Name field, enter the label or virtual server name. See ["Entries to Control Evaluation Flow in a Policy Bank"](#) for details

To invoke a caching policy label in a virtual server policy bank by using the command line interface

At the command prompt, type:

- o bind lb vserver <name>@ -policyName <policyName>|<NOPOLICY-CACHE> -priority <positiveInteger> -gotoPriorityExpression <expression> -type REQUEST|RESPONSE -invoke <labelType> <labelName>
- o bind cs vserver <name> -policyName <policyName>|<NOPOLICY-CACHE> -priority <positiveInteger> -gotoPriorityExpression <expression> -type REQUEST|RESPONSE -invoke <labelType> <labelName>

For more information, see ["Entries to Control Evaluation Flow in a Policy Bank."](#)

To invoke a caching policy label in a virtual server policy bank by using the configuration utility

1. Navigate to Traffic Management > Load Balancing/Content Switching > Virtual Servers, select the virtual server, and click Policies.
2. If you are configuring an existing entry in this bank, skip this step. If you are adding a new policy to this policy bank, or you want to use the "NOPOLICY" entry, click Add, and do one of the following:
 - o To configure a new policy, click Cache and configure the new policy as described in ["Configuring a Policy in the Integrated Cache."](#)
 - o To invoke a policy bank without processing a policy a rule, select the NOPOLICY-CACHE option.

Note: To invoke an external policy bank, click the field in the Invoke Type column, and select the type of policy bank that you want to invoke at this point in the policy bank. This can be a global label or a virtual server bank. In the Invoke Name field, enter the label or virtual server name. See ["Entries to Control Evaluation Flow in a Policy Bank"](#) for details

Configuring a Policy Label in the Integrated Cache

In addition to configuring policies in a policy bank for one of the built-in bind points or a virtual server, you can create caching policy labels and configure banks of policies for these new labels.

A policy label for the integrated cache can be invoked only from one of the bind points that you can view in the Policy Manager in the **Integrated Caching** details pane (request override, request default, response override, or response default) or the built-in policy labels `_reqBuiltinDefaults` and `_resBuiltinDefaults`. You can invoke a policy label any number of times unlike a policy, which can only be invoked once.

The configuration utility provides an option to rename a policy label. Renaming a policy label does not affect the process of evaluation of the policies bound to the label.

Note: You can use the NOPOLICY "dummy" policy to invoke any policy label from another policy bank. The NOPOLICY entry is a placeholder that does not process a rule.

To configure a policy label for caching by using the command line interface

At the command prompt, type the following command to create a policy label and verify the configuration:

- o add cache policylabel <labelName> -evaluates (REQ|RES)
- o show cache policylabel <labelName>

Invoke this policy label from a policy bank. For more information, see "[Configuring a Policy Bank for Caching](#)."

To configure a policy label for caching by using the configuration utility

Navigate to Optimization > Integrated Caching > Policy Labels, add a policy label, and bind the cached policies.

Note: To ensure that the NetScaler ADC processes the policy label at the right time, configure an invocation of this label in one of the policy banks that are associated with the built-in bind points

To rename a policy label by using the configuration utility

Navigate to Optimization > Integrated Caching > Policy Labels, select the policy label, and rename.

Unbinding and Deleting an Integrated Caching Policy and Policy Label

You can unbind a policy from a policy bank, and you can delete it. To delete the policy, you must first unbind it. You can also remove a policy label invocation and delete a policy label. To delete the policy label, you must first remove any invocations that you have configured for the label.

You cannot unbind or delete the labels for the built-in bind points (request default, request override, response default, and response override).

To unbind a global caching policy by using the command line interface

At the command prompt, type:

```
unbind cache global <policy>
```

To unbind a virtual server-specific caching policy by using the command line interface

At the command prompt, type:

```
(unbind lb vserver|unbind cs vserver) <vserverName> -policyName <policyName> -type (REQUEST|RESPONSE)
```

To delete a caching policy by using the command line interface

At the command prompt, type:

```
rm cache policy <policyName>
```

To unbind a caching policy by using the configuration utility

Navigate to Optimization > Integrated Caching, click Cache Policy Manager, and unbind policies by specifying the relevant bind point and connection type (Request/Response).

To delete a policy label invocation by using the configuration utility

1. Navigate to Optimization > Integrated Caching, click Cache policy manager, and specify the relevant bind point (LB virtual server or CS virtual server) and connection type to view the list of cache policies bound to this virtual server.
2. In the policy Invoke column, clear the entry.

Caching Support for Database Protocols

The integrated cache monitors database requests that flow through the Citrix® NetScaler® appliance and caches them as determined by the cache policies. Users have to configure the cache policies for MySQL and MSSQL protocols, because the NetScaler does not provide any default policies for these protocols. When configuring the protocols, remember that request based policies currently support CACHE and INVALID actions, while response based policies currently support only NOCACHE action. After configuring the policies, bind them to virtual servers. MySQL and MSSQL policies, both request and response, can be bound only to virtual servers

Before creating a cache policy, create a cache content group of type MySQL or MSSQL. When you create a MySQL or MSSQL cache content group, associate at least one hit selector with it. See ["Setting Up a Basic Content Group"](#) for setting up cache content groups.

The following example illustrates the procedure for configuring and verifying cache support for SQL protocols.

```
> enable feature IC
> set cache parameter -memlimit 100
> add cache selector sell mssql.req.query.text

> add cache contentgroup cg1 -type "MSSQL" -hitselector "sell" -invalselector "inval_sel" -
"100"
> add cache policy cp1 -rule "mssql.req.query.command.contains(\"select\")" -action "CACHE"
> add cache policy cp2 -invalObjects "cg1" -rule "mssql.req.query.text.contains(\"insert\")"
> add db user user1 -password "Pass1"
> add service svc_sql_1 10.102.147.70 mssql 64834 -healthMonitor "NO" -downstateflush "ENABL
> add lb vserver lb_mssql1 mssql 10.102.147.77 1433 -lbmethod "roundrobin"
> bind lb vserver lb_mssql1 svc_sql_1
> bind lb vserver lb_mssql1 -policyName cp1 -type "REQUEST" -priority "2"
> bind lb vserver lb_mssql1 -policyName cp2 -type "REQUEST" -priority "1"

> show cache selector sell
    Name:sell
                                Expressions:
                                1)mssql.req.query.text

> show cache policy cp1
                                Name:cp1
    Rule:mssql.req.query.command.contains("select")
                                CacheAction:CACHE
                                Stored in group: cg1
                                UndefAction:Use Global
                                Hits:2
                                Undef Hits:0
                                Policy is bound to following entities
                                1) Bound to:
                                                REQ VSERVER lb_mssql1
                                                Priority:2
                                                GotoPriorityExpression: END
```

Note: The methods for reducing flash crowds, as explained in ["Reducing Flash Crowds"](#), are not supported for MySQL and MSSQL protocols.

Configuring Expressions for Caching Policies and Selectors

A request-time expression examines data in request-time transaction, and a response-time expression examines data in a response-time transaction. In a policy for caching, if an expression matches data in a request or response, the Citrix NetScaler appliance takes the action associated with the policy. In a selector, request-time expressions are used to find matching responses that are stored in a content group.

Before configuring policies and selectors for the integrated cache, you need to know, at minimum, the host names, paths, and IP addresses that appear in HTTP request and response URLs. And you probably need to know the format of entire HTTP requests and responses. Programs such as Live HTTP Headers (<http://livehttpheaders.mozdev.org/>) or HTTPFox (<https://addons.mozilla.org/en-US/firefox/addon/6647>) can help you investigate the structure of the HTTP data that your organization works with.

Following is an example of an HTTP GET request for a stock quote program:

```
GET /quote.dll?page=dynamic&mode=data&mode=stock&symbol=CTXS&page=multi&selected=CTXS&random
Host: quotes.mystockquotes.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9) Gecko/2008052906 Firefox
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate,compress,pack200-gzip
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://quotes.mystockquotes.com/quote.dll?mode=stock&symbol=CTXS&page=multi&selecte
Cookie: __qca=1210021679-72161677-10297606
```

When configuring an expression, note the following limitations:

Table 1. Restrictions on Request-Time and Response-Time Expressions

Expression Type	Restrictions
Request	Do not configure request-time expressions in a policy with a CACHE or NOCACHE action. Use MAY_CACHE or MAY_NOCACHE instead.
Response	<p>Configure response-time expressions in caching policies only.</p> <ul style="list-style-type: none">Selectors can use only request-time expressions.Do not configure response-time expressions in a policy with an INVALID action. <p>Do not configure response-time expressions in a policy with a CACHE action and a parameterized content group. Use the MAY_CACHE action.</p>

Note: For a comprehensive discussion of advanced expressions, see "Policies and Expression."

Expression Syntax

Following are basic components of the syntax:

- Separate keywords with periods (.), as follows:

```
http.req.url
```

- Enclose string values in parentheses and quotes, as follows:

```
http.req.url.query.contains("this")
```

- When configuring an expression from the command line, you must escape internal quote marks (the quotes that delimit values in the expression, as opposed to the quotes that delimit the expression). One method is to use a slash, as followings:

```
\ "abc\"
```

Selector expressions are evaluated in order of appearance, and multiple expressions in a selector definition are joined by a logical AND. Unlike selector expressions, you can specify Boolean operators and modify the precedence in an advanced expression for a policy rule.

Configuring an Expression in a Caching Policy or a Selector

Updated: 2014-08-04

Note that on the command line, the syntax for a policy expression is somewhat different from a selector expression. For a comprehensive discussion of advanced expressions, see ["Policies and Expressions."](#)

To configure a policy expression by using the command line interface

1. Start the policy definition as described in ["Globally Binding an Integrated Caching Policy."](#)
2. To configure the policy rule, delimit the entire rule in quotes, and delimit string values within the rule in escaped quotes.

The following is an example:

```
"http.req.url.contains(\"jpg\")"
```

3. To add Boolean values, insert &&, ||, or ! operators.

The following are examples:

```
"http.req.url.contains(\"jpg\") || http.req.url.contains(\"jpeg\")"
```

```
"http.req.url.query.contains(\"IssuePage\")"
```

```
"http.req.header(\"Host\")contains(\"my.company.com\") && http.req.method.eq(\"GET\") &
```

4. To configure an order of evaluation for the constituent parts of a compound

```
"http.req.url.contains(\"jpg\") || (http.req.url.contains(\"jpeg\") && http.req.method.
```

To configure a selector expression by using the command line interface

1. Start the selector definition as described in ["About Content Groups."](#)
2. To configure the selector expression, delimit the entire rule in quotes, and delimit string values within the rule in escaped quotes.

The following is an example:

```
"http.req.url.contains(\"jpg\")"
```

3. You cannot add Boolean values, insert &&, ||, or ! operators. Enter each expression element delimited in quotes. Multiple expressions in the definition are treated as a compound expression joined by logical ANDs.

The following are examples:

```
"http.req.url.query.value(\"ProductId\")" "http.req.url.query.value(\"BatchNum\")" "htt
```

To configure a policy or selector expression by using the configuration utility

1. Start the policy or selector definition as described in ["To configure a policy for caching or invalidation by using the configuration utility"](#) or ["To configure a selector by using the configuration utility."](#)
2. In the Expression field, you can either manually type the default syntax by clicking Switch to Classic Syntax or create new expression using Expression Editor.
3. To insert an operator between two parts of a compound expression, click the Operators button and select the operator type. The following is an example of a configured expression with a Boolean OR (signaled by double vertical bars, ||):
4. Click Frequently Used Expressions drop-down to insert the commonly used expressions.
5. To test the expression, click the Evaluate. In the Expression Evaluator dialog box, select the Flow Type that matches the expression. In the data field, paste the HTTP request or response that you hope to parse using the expression, and click Evaluate.

Displaying Cached Objects and Cache Statistics

You can view particular cached objects, and you can view summary statistics on cache hits, misses, and memory usage. The statistics provide insight on the amount of data that is being served from the cache, what items are responsible for the largest performance benefit, and what you can tune to improve cache performance.

This section includes the following details:

- [Viewing Cached Objects](#)
- [Finding Particular Cached Responses](#)
- [Viewing Cache Statistics](#)

Viewing Cached Objects

Updated: 2014-08-04

After enabling caching, you can view details for cached objects. For example, you can view the following items:

- Response sizes and header sizes
- Status codes
- Content groups
- ETag, Last-Modified, and Cache-Control headers
- Request URLs
- Hit parameters
- Destination IP addresses
- Request and response times

To view a list of cached objects by using the command line interface

At the command prompt, type:

show cache object

Table 1. Properties of Cached Objects

Properties	Specifies
Response size (bytes)	The size of the response header and body.
Response header size (bytes)	The size of the header portion of the response.
Response status code	The status code sent with the response.
ETag	The ETag header inserted in the response. Typically, this header indicates whether the response has changed recently.
Last-Modified	The Last-Modified header inserted in the response. This header indicates the date that the response was last changed.
Cache-Control	The Cache-Control header inserted in the response.
Date	The Date header that indicates when the response was sent.
Contentgroup	The content group where the response is stored.

Complex match	If this object was cached on the basis of parameterized values, this field value is YES.
Host	The host specified in the URL that requested this response.
Host port	The listen port for the host specified in the URL that requested this response.
URL	The URL issued for the stored response.
Destination IP	The IP address of the server from which this response was fetched.
Destination port	The listen port for the destination server.
Hit parameters	If the content group that stores the response uses hit parameters, they are listed in this field.
Hit selector	If this content group uses a hit selector, it is listed in this field.
Invalid selector	If this content group uses an invalidation selector, it is listed in this field.
Selector Expressions	If this content group uses a selector, this field displays the expression that defines the selection rule.
Request time	The time in milliseconds since the request was issued.
Response time	The time in milliseconds since the cache started to receive the response.
Age	Amount of time the object has been in the cache.
Expiry	Amount of time after which the object is marked as expired.
Flushed	Whether the response has been flushed after expiry.
Prefetch	If Prefetch has been configured for this content group, the amount of time before expiry during which the object is fetched from the origin. Prefetch does not apply to negative objects (for example, 404 "object not found" responses).
Current readers	Approximately the current number of hits being served. When a response with a Content-Length header object is being downloaded, the current misses and the current readers values are each typically 1. When a chunked response object is being downloaded, the current misses value is typically 1, but the current readers value is typically 0, because the chunked response that is served to the client does not come from the integrated caching buffers.
Current misses	The current number of requests that resulted in a cache miss and fetching from the origin server. This value is typically 0 or 1. If Poll Every Time is enabled for a content group, the count can be greater than 1.
Hits	The number of cache hits for this object.
Misses	The number of cache misses for this object.

Compression format	The type of compression applied to this object. Compression formats include gzip, deflate, compress, and pack200-gzip.
HTTP version in response	The version of HTTP that was used to send the response.
Weak etag present in response	Strong etag headers change if the bits of an entity change. Strong headers are based on the octet values of an object. Weak etag headers change if the meaning of an entity changes. Weak etag values are based on semantic identity. Weak etags values start with a "W."
Negative marker cell	A marker object is cacheable, but it does not yet meet all the criteria for being cached. For example, the object may exceed the maximum response size for the content group. A marker cell is created for objects of this type. The next time a user sends a request for this object, a cache miss is served.
Reason marker created	The reason a marker cell was created (for example, "Waiting for minhit" or "Content-length response data is not in group size limit").
Auto poll every time	If the integrated cache receives an already expired 200 OK response with validators (either the Last-Modified or the ETag response headers) it stores the response and marks it as Auto-PET (automatically poll every time).
NetScaler Etag inserted in response	A variation of the ETag header generated by the NetScaler appliance. A value of YES appears if the NetScaler inserts an Etag in the response.
Full response present in cache	Indicates whether this is a complete response.
Destination IP verified by DNS	Indicates whether DNS resolution was performed when storing the object.
Object stored through a cache forward proxy	Indicates whether this response was stored due to a forward proxy that is configured in the integrated cache.
Object is a Delta basefile	A response that is delta-compressed.
Waiting for minhits	Indicates whether this content group requires a minimum number of origin server hits before caching a response.
Minhit count	If this content group requires a minimum number of origin server hits before caching an object, this field displays a count of the number of hits received so far.
HTTP Request Method	The method, GET or POST, used in the request that obtained this object.
Stored by policy	

	The name of the caching policy that caused this object to be stored. A value of NOT AVAILABLE indicates that the policy has been deactivated or deleted. A value of NONE indicates that the object did not match a visible policy, but was stored according to internal criteria for caching.
Application firewall metadata exists	This parameter is used when the application firewall and the integrated cache are both enabled. The application firewall analyzes the contents of a response page, stores its metadata (for example, URLs and forms contained in page), and exports the metadata with the response to the cache. The cache stores the page and the metadata, and when the cache serves the page, it sends the metadata back to the requestâ€™s session.
HTTP callout object, name, type, response	These cells indicate whether this data was stored as a result of an HTTP Callout expression, and provide information about various aspects of the callout and the corresponding response. For more information about HTTP callouts, see "HTTP Callouts".

To view cached objects by using the configuration utility

To view cached objects by using the configuration utility

Navigate to Optimization > Integrated Caching > Cache Objects. You can view all the cached objects and sort them accordingly as per your requirement.

Finding Particular Cached Responses

Updated: 2014-08-08

You can find individual items in the cache based on search criteria. There are different methods for finding cached items, depending on whether the content group that contains the data uses hit and invalidation selectors, as follows:

- If the content group uses selectors, you can only conduct the search using the Locator ID for the cached item.
- If the content group does not use selectors, you conduct the search using criteria such as URL, host, content group name, and so on.

When searching for a cached response, you can locate some items by URL and host. If the response is in a content group that uses a selector, you can find it only by using a Locator number (for example, 0x00000000ad7af00000050). To save a Locator number for later use, right-click the entry and select **Copy**. For more information about selectors, see "Configuring Selectors and Basic Content Groups."

To display cached responses in content groups that do not have a selector by using the command line interface

At the command prompt, type:

```
show cache object [-locator <positiveInteger>] | [(-url <URL> (-host <hostName> [-port <port>] [-groupName <contentGroupName>] [-httpMethod GET | POST ])) | [-httpStatus<positive integer>] | -group <contentGroupName> | -ignoreMarkerObjects (ON | OFF) | -includeNotReadyObjects (ON | OFF)]
```

To display cached responses in content groups that have a selector by using the command line interface

At the command prompt, type:

```
show cache object -locator <locatorString> MarkerObjects ( ON | OFF ) | -includeNotReadyObjects ( ON | OFF ) | [-httpStatus <positive integer>]
```

To display cached responses in content groups that do not have a selector by using the configuration utility

Navigate to Optimization > Integrated Caching > Cache Objects, click Search, and set the search criteria to view the required cached response.

If you have not yet configured any content groups, all of the objects are in the Default group.

To display cached responses in content groups that have a selector by using the configuration utility

Navigate to Optimization > Integrated Caching > Cache Objects, click Search, and set the selector search criteria to view the required cached response.

Viewing Cache Statistics

Updated: 2013-10-28

The following table summarizes the detailed cache statistics that you can view.

Table 2. Integrated Cache Statistics

Counter	Specifies
Hits	Responses that are found in and served from the integrated cache. Includes static objects such as image files, pages with status codes 200, 203, 300, 301, 302, 304, 307, 403, 404, 410, and responses that match a user-defined policy with a CACHE action..
Misses	Intercepted HTTP requests where the response was ultimately fetched from origin server.
Requests	Total cache hits plus total cache misses.
Non-304 hits	<p>If the user requests an item more than once, and the item in the cache is unchanged since the last time the NetScaler appliance served it, the NetScaler appliance serves a 304 response instead of the cached object.</p> <p>This statistic indicates how many items the NetScaler appliance served from the cache, excluding 304 responses.</p>
304 hits	Number of 304 (object not modified) responses the NetScaler appliance served from the cache.
304 hit ratio (%)	Percentage of 304 responses that the NetScaler appliance served, relative to other responses.
Hit ratio (%)	Percentage of responses that the NetScaler appliance served from the cache (cache hits) relative to responses that could not be served from the cache.
Origin bandwidth saved (%)	An estimate of the processing capacity that the NetScaler appliance saved on the origin server due to serving responses from the cache.
Bytes served by the NetScaler	Total number of bytes that the NetScaler appliance served from the origin server and the cache.
Bytes served by cache	Total number of bytes that the NetScaler appliance served from the cache.
Byte hit ratio (%)	Percentage of data that the NetScaler appliance served from the cache, relative to all of the data in all served responses.
Compressed bytes from cache	Amount of data, in bytes, that the NetScaler appliance served in compressed form.

Storable misses	If the NetScaler appliance does not find a requested object in the cache, it fetches the object from the origin server. This is known as a cache miss. A storable cache miss can be stored in the cache.
Non-storable misses	A non-storable cache miss cannot be stored in the cache.
Misses	All cache misses.
Revalidations	Max-Age setting in a Cache-Control header determines, in number of seconds, when an intervening cache must revalidate the content with the integrated cache before serving it to the user. For more information, see "Inserting a Cache-Control Header."
Successful revalidations	Number of re-validations that have been performed. For more information, see "Inserting a Cache-Control Header."
Conversions to conditional req	A user-agent request for a cached PET object is always converted to a conditional request and sent to the origin server. For more information, see "Polling the Origin Server Every Time a Request Is Received."
Storable miss ratio (%)	Storable cache misses as a percentage of non-storable cache misses.
Successful reval ratio (%)	Successful revalidations as a percentage of all revalidation attempts. For more information, see "Inserting a Cache-Control Header."
Expire at last byte	Number of times that the cache expired content immediately after receiving the last body byte. Only applicable to positive responses, as described in the table "Cache Hits and Misses." For more information, see "Example of Performance Optimization."
Flashcache misses	If you enable Flash Cache, the cache allows only one request to reach the server, eliminating flash crowds. This statistic indicates the number of Flash Cache requests that were cache misses. For more information, "Queuing Requests to the Cache."
Flashcache hits	Number of Flash Cache requests that were cache hits. For more information, see "Queuing Requests to the Cache."
Parameterized inval requests	Requests that match a policy with an invalidation (INVAL) action and a content group that uses an invalidation selector or parameters to selectively expire cached objects in the group.
Full inval requests	Requests that match an invalidation policy where the invalGroups parameter is configured and expires one or more content groups.
Inval requests	Requests that match an invalidation policy and result in expiration of specific cached responses or entire content groups.
Parameterized requests	Number of cache requests that were processed using a policy with a parameterized content group.

Parameterized non-304 hits	Number of cache requests that were processed using a policy with a parameterized content group, where full cached response was found, and the response was not a 304 (object not updated) response.
Parameterized 304 hits	Number of cache requests that were processed using a policy with a parameterized content group, where the cached object was found, and the object was a 304 (object not updated) response.
Total parameterized hits	Number of cache requests that were processed using a policy with a parameterized content group, where the cached object was found.
Parameterized 304 hit ratio (%)	Percentage of 304 (object not updated) responses that were found using a parameterized policy, relative to all cache hits.
Poll every time requests	<p>If Poll Every Time is enabled, the NetScaler appliance always consults the origin server before serving a stored object.</p> <p>For more information, see "Polling the Origin Server Every Time a Request Is Received."</p>
Poll every time hits	<p>Number of times a cache hit was found using the Poll Every Time method.</p> <p>For more information, see "Polling the Origin Server Every Time a Request Is Received."</p>
Poll every time hit ratio (%)	<p>Percentage of cache hits using the Poll Every Time method, relative to all searches for cached objects using Poll Every Time.</p> <p>For more information, see "Polling the Origin Server Every Time a Request Is Received."</p>
Maximum memory (KB)	Maximum amount of memory in the NetScaler appliance that is allocated to the cache. For more information, see "Configuring Global Attributes for Caching."
Maximum memory active value (KB)	Maximum amount of memory (active value) that will be set after the memory is actually allocated to the cache. For more information, see "How to Configure the Integrated Caching Feature of a NetScaler Appliance for various Scenarios."
Utilized memory (KB)	Amount of memory that is actually being used.
Memory allocation failures	Number of failed attempts to utilize memory for the purpose of storing a response in the cache.
Largest response so far	Largest response in bytes found in either the cache or the origin server and sent to the client.
Cached objects	Number of objects in the cache, including responses that have not yet been fully downloaded and responses that have been expired but not yet flushed.
Marker objects	Marker objects are created when a response exceeds the maximum or minimum response size for the content group, or has not yet received the minimum number of hits for the content group.

Hits being served	Number of hits that have been served from the cache.
Misses being handled	Responses that were fetched from the origin server, stored in the cache, and then served. Should approximate the number for storable misses. Does not include non-storable misses.

To view summary cache statistics by using the command line interface

At the command prompt, type:

```
stat cache
```

To view specific cache statistics by using the command line interface

At the command prompt, type:

```
stat cache -detail [-fullValues] [-ntimes <positiveInteger>] [-logFile <inputFilename>]
```

To view summary cache statistics by using the configuration utility

1. Click the Dashboard tab at the top of the page.
2. Scroll down to the Integrated Caching section of the window.
3. To see detailed statistics, click the More... link at the bottom of the table.

To view specific cache statistics by using the configuration utility

1. Click the Reporting tab at the top of the page.
2. Under Built-In Reports, expand Integrated Cache, and then click the report with the statistics you want to view.
3. To save the report as a template, click Save As and name the report. The saved report appears under Custom Report

Improving Cache Performance

You can improve the performance of integrated cache, including handling simultaneous requests for the same cached data, avoiding delays that are associated with refreshing cached responses from the origin server, and ensuring that a response is requested often enough to be worth caching.

This section includes the following details:

- [Reducing Flash Crowds](#)
- [Caching a Response after a Client Halts a Download](#)
- [Requiring a Minimum Number of Server Hits before Caching](#)
- [Example of Performance Optimization](#)

Reducing Flash Crowds

Updated: 2015-05-20

Flash crowds occur when many users simultaneously request the same data. All of the requests in a flash crowd can become cache misses if you configured the cache to serve hits only after the entire object is downloaded.

The following techniques can reduce or eliminate flash crowds:

- **PREFETCH:** Refreshes a positive response before it expires to ensure that it never becomes stale or inactive.

For more information, see ["Refreshing a Response Prior to Expiration."](#)

- **Cache buffering:** Starts serving a response to multiple clients as soon as it receives the response header from the origin server, rather than waiting for the entire response to be downloaded.

The only limit on the number of clients that can download a response simultaneously is the available system resources.

The Citrix NetScaler appliance downloads and serves responses even if the client that initiated the download halts before the download is complete. If the size of the response exceeds the cache size or if the response is chunked, the cache stops storing the response, but service to the clients is not disrupted.

- **Flash Cache:** Flash Cache queues requests to the cache, and allows only one request to reach the server at a time.

For more information, see ["Queuing Requests to the Cache."](#)

Refreshing a Response Before Expiration

To ensure that a cached response is fresh whenever it is needed, the PREFETCH option refreshes a response before its calculated expiration time. The prefetch interval is calculated after receiving the first client request. From that point onward, the NetScaler appliance refreshes the cached response at a time interval that you configure in the PREFETCH parameter.

This setting is useful for data that is updated frequently between requests. It does not apply to negative responses (for example, 404 messages).

To configure prefetch for a content group by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -prefetch YES [-prefetchPeriod <seconds> | -prefetchPeriodMilliSec <milliseconds>] [-prefetchMaxPending <positiveInteger>]
```

To configure prefetch for a content group by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Others tab, in the Flash Crowd and Prefetch group, select Prefetch option, and specify the values in Interval and Maximum number of pending prefetches text boxes.

Queuing Requests to the Cache

The Flash Cache option queues requests that arrive simultaneously (a flash crowd), retrieves the response, and distributes it to all the clients whose requests are in the queue. If, during this process, the response becomes non-cacheable, the NetScaler appliance stops serving the response from the cache and instead serves the origin server's response to the queued clients. If the response is not available, the clients receive an error message.

Flash Cache is disabled by default. You cannot enable Poll Every Time (PET) and Flash Cache on the same content group.

One disadvantage of Flash Cache is if the server replies with an error (for example, a 404 that is quickly remedied), the error is fanned out to the waiting clients.

Note: If Flash Cache is enabled, in some situations the NetScaler appliance is unable to correctly match the Accept-Encoding header in the client request with the Content-Encoding header in the response. The NetScaler appliance can assume that these headers match and mistakenly serve a hit. As a work-around, you can configure Integrated Caching policies to disallow serving hits to clients that do not have an appropriate Accept-Encoding header.

To enable Flash Cache by using the command line interface

At the command prompt, type:

```
set cache contentgroup <contentGroupName> -flashcache yes
```

To enable Flash Cache by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Others tab, in the Flash Crowd and Prefetch group, select Prefetch option.

Caching a Response after a Client Halts a Download

Updated: 2014-08-08

You can set the Quick Abort parameter to continue caching a response, even if the client halts a request before the response is in the cache.

If the downloaded response size is less than or equal to the Quick Abort size, the NetScaler appliance stops downloading the response. If you set the Quick Abort parameter to 0, all downloads are halted.

To configure quick abort size by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -quickAbortSize <integerInKBytes>
```

To configure quick abort size by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Memory tab, set the relevant value in Quick Abort: Continue caching if more than text box.

Requiring a Minimum Number of Server Hits before Caching

Updated: 2015-05-19

You can configure the minimum number of times that a response must be found on the origin server before it can be cached. You should consider increasing the minimum hits if the cache memory fills up quickly and has a lower-than-expected hit ratio.

The default value for the minimum number of hits is 0. This value caches the response after the first request.

To configure the minimum number of hits that are required before caching by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -minhits <positiveInteger>
```

To configure the minimum number of hits that are required before caching by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Memory tab, set the relevant value in Do not cache, if hits are less than text box.

Example of Performance Optimization

Updated: 2013-10-28

In this example, a client accesses a stock quote. Stock quotes are highly dynamic. You configure the integrated cache to serve the same stock quote to concurrent clients without sending multiple requests to the origin server. The stock quote expires after it is downloaded to all of the clients, and the next request for a quote for the same stock is fetched from the origin server. This ensures that the quote is always up to date.

The following task overview describes the steps to configure the cache for the stock quote application.

Task overview: Configuring caching for a stock quote application

1. Create a content group for stock quotes.

For more information, see ["About Content Groups."](#)

Configure the following for this content group:

- On the Expiry Method tab, select the Expire after complete response received check box.
- On the Others tab, select the Flash Cache check box, and click Create.

2. Add a cache policy to cache the stock quotes.

For more information, see ["Configuring a Policy in the Integrated Cache."](#)

Configure the following for the policy:

- In the Action and Store in Group lists, select CACHE and select the group that you defined in the previous step.
- Click Add, and in the Add Expression dialog box configure an expression that identifies stock quote requests, for example:

```
http.req.url.contains( "cgi-bin/stock-quote.pl" )
```

3. Activate the policy.

For more information, see ["Globally Binding an Integrated Caching Policy."](#) In this example, you bind this policy to request-time override processing and set the priority to a low value.

Configuring Cookies, Headers, and Polling

This section describes the procedures to configure how the cache manages cookies, HTTP headers, and origin server polling, including modifying default behavior that causes the cache to diverge from documented standards, overriding HTTP headers that might cause cacheable content to not be stored in the cache, and configuring the cache to always poll the origin for updated content under specialized circumstances.

For details, see the following sections:

- [Divergence of Cache Behavior from the Standards](#)
- [Removing Cookies from a Response](#)
- [Inserting HTTP Headers at Response Time](#)
- [Ignoring Cache-Control and Pragma Headers in Requests](#)
- [Polling the Origin Server Every Time a Request Is Received](#)
- [PET and Client-Specific Content](#)
- [PET and Authentication, Authorization, and Auditing](#)

Divergence of Cache Behavior from the Standards

Updated: 2013-10-28

By default, the integrated cache conforms to the following standards:

- RFC 2616, "Hypertext Transfer Protocol HTTP/1.1"
- The caching behaviors described in RFC 2617, "HTTP Authentication: Basic and Digest Access Authentication"
- The caching behavior described in RFC 2965, "HTTP State Management Mechanism"

The built-in policies and the Default content group attributes ensure conformance with most of these standards.

The default integrated cache behavior diverges from the specifications as follows:

- There is limited support for the Vary header.

By default, any response containing a Vary header is considered to be non-cacheable unless it is compressed. A compressed response contains `Content-Encoding: gzip`, `Content-Encoding: deflate`, or `Content-Encoding: pack200-gzip` and is cacheable even if it contains the `Vary: Accept-Encoding` header.

- The integrated cache ignores the values of the headers `Cache-Control: no-cache` and `Cache-Control: private`.

For example, a response that contains `Cache-Control: no-cache=Set-Cookie` is treated as if the response contained `Cache-Control: no-cache`. By default, the response is not cached.

- An image (`Content-Type = image/*`) is always considered cacheable even if an image response contains `Set-Cookie` or `Set-Cookie2` headers, or if an image request contains a `Cookie` header.

The integrated cache removes `Set-Cookie` and `Set-Cookie2` headers from a response before caching it. This diverges from RFC 2965. You can configure RFC-compliant behavior as follows:

```
add cache policy rfc_compliant_images_policy -rule "http.res.header.set-cookie2.  
bind cache global rfc_compliant_images_policy -priority 100 -type REQ_OVERRIDE
```

- The following `Cache-Control` headers in a request force an RFC-compliant cache to reload a cached response from the origin server:

```
Cache-control: max-age=0  
Cache-control: no-cache
```

To guard against Denial of Service attacks, this behavior is not the default. For more information, see "[Inserting a Cache-Control Header](#)."

- By default, the caching module considers a response to be cacheable unless a response header states otherwise.

To make this behavior RFC 2616 compliant, set `-weakPosRelExpiry` and `-weakNegResExpiry` to 0 for all content groups.

Removing Cookies from a Response

Updated: 2014-08-12

Cookies are often personalized for a user, and typically should not be cached. The Remove Response Cookies parameter removes Set-Cookie and Set-Cookie2 headers before caching a response. By default, the Remove Response Cookies option for a content group prevents caching of responses with Set-Cookie or Set-Cookie2 headers.

Note that when images are cached, the built-in behavior is to remove the Set-Cookie and Set-Cookie2 headers before caching, no matter how the content group is configured.

Note: Citrix recommends that you accept the default Remove Response Cookies for every content group that stores embedded responses, for example, images.

To configure Remove Response Cookies for a content group by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -removeCookies YES
```

To configure Remove Response Cookies for a content group by using the configuration utility

- Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
- On Others tab, in the Settings group, select Remove response cookies option.

Inserting HTTP Headers at Response Time

Updated: 2014-08-12

The integrated cache can insert HTTP headers in responses that result from cache hits. The Citrix® NetScaler® appliance does not alter headers in responses that result from cache misses.

The following table describes headers that you can insert in a response.

Table 1. Different HTTP Headers You Can Insert in a Response That Is Served from the Cache

Header	Specifies
Age	Provides the age of the response in seconds, calculated from the time the response was generated at the origin server. By default, the cache inserts an Age header for every response that is served from the cache.
Via	Lists protocols and recipients between the start and end points for a request or a response. The NetScaler appliance inserts a Via header in every response that it serves from the cache. The default value of the inserted header is <code>NS-CACHE-9.2:last octet of the NetScaler IP address</code> . For more information, see "Configuring Global Attributes for Caching."
ETag	The cache supports response validation using Last-Modified and ETag headers to determine if a response is stale. The cache inserts an ETag in a response only if it caches the response and the origin server has not inserted its own ETag header. The ETag value is an arbitrary unique number. The ETag value for a response changes if it is refreshed from the origin server, but it stays the same if the server sends a 304 (object not updated) response. Origin servers typically do not generate validators for dynamic content because dynamic content is considered non-cacheable. You can override this behavior. With ETag header

	insertion, the cache is permitted to not serve full responses. Instead, the user agent is required to cache the dynamic response sent by the integrated cache the first time. To force a user agent to cache a response, you configure the integrated cache to insert an ETag header and replace the origin-provided Cache-Control header.
Cache-Control	<p>The NetScaler appliance typically does not modify cacheability headers in responses that is serves from the origin server. If the origin server sends a response that is labeled as non-cacheable, the client treats the response as non-cacheable even if the NetScaler appliance caches the response.</p> <p>To cache dynamic responses in a user agent, you can replace Cache-Control headers from the origin server. This applies only to user agents and other intervening caches. They do not affect the integrated cache.</p> <p>For more information, see "Inserting a Cache-Control Header."</p>

Inserting an Age, Via, or ETag Header

The following procedures describe how to insert Age, Via, and ETag headers.

To insert an Age, Via, or Etag header by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -insertVia YES -insertAge YES -insertETag YES
```

To insert an Age, Via, or Etag header by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Others tab, in the HTTP Header Insertions group, select the Via, Age, or ETag options, as appropriate.

The values for the other header types are calculated automatically. Note that you configure the Via value in the main settings for the cache.

Inserting a Cache-Control Header

When the integrated cache replaces a Cache-Control header that the origin server inserted, it also replaces the Expires header. The new Expires header contains an expiration time in the past. This ensures that HTTP/1.0 clients and caches (that do not understand the Cache-Control header) do not cache the content.

To insert a Cache-Control header by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -cacheControl <value>
```

To insert a Cache-Control header by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and
 - a. Click Expiry Method tab, clear the heuristic and default expiry settings and set the relevant value in Expire content after text box.
 - b. Click Others tab and type the header you want to insert in the Cache-Control text box. Alternatively, click Configure to set the Cache-Control directives in cached responses.

Ignoring Cache-Control and Pragma Headers in Requests

Updated: 2014-08-12

By default, the caching module processes Cache-Control and Pragma headers. The following tokens in Cache-Control headers are processed as described in RFC 2616.

- o max-age
- o max-stale
- o only-if-cached
- o no-cache

A Pragma: no-cache header in a request is treated in the same way as a Cache-Control: no-cache header.

If you configure the caching module to ignore Cache-Control and Pragma headers, a request that contains a Cache-Control: No-Cache header causes the NetScaler appliance to retrieve the response from the origin server, but the cached response is not updated. If the caching module processes Cache-Control and Pragma headers, the cached response is refreshed.

The following table summarizes the implications of various settings for these headers and the Ignore Browser's Reload Request setting.

Table 2. Outcome of Settings for Ignoring Reload Requests, Cache-Control, and Pragma Headers

Setting for Ignore Cache-Control and Pragma Headers	Setting for Ignore Browser's Reload Request	Outcome
Yes	Yes or No	Ignore the Cache-Control and Pragma headers from the client, including the Cache-Control: no-cache directive.
No	Yes	The Cache-Control: no-cache header produces a cache miss, but a response that is already in the cache is not refreshed.
No	No	A request that contains a Cache-Control: no-cache header causes a cache miss and the stored response is refreshed.

To ignore Cache-Control and Pragma headers in a request by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -ignoreReqCachingHdrs YES
```

To ignore browser reload requests by using the command line interface

At the command prompt, type:

```
set cache contentgroup <name> -ignoreReloadReq NO
```

Note that by default, the -ignoreReloadReq parameter is set to YES.

To ignore Cache-Control and Pragma headers in a request by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Others tab, in the Settings group, select Ignore Cache-control and Pragma Headers in Requests option.

Example of a Policy to Ignore Cache-Control Headers

In the following example, you configure a request-time override policy to cache responses that contain Content-type: image/* regardless of the Cache-Control header in the response.

To configure a request-time override policy to cache all responses with image/*

1. Flush the cache using the Invalidate All option.

For more information, see ["Flushing Responses in a Content Group."](#)

2. Configure a new cache policy, and direct the policy to a particular content group. For more information, see ["Configuring a Policy in the Integrated Cache."](#)
3. Ensure the content group that the policy uses is configured to ignore Cache-Control headers, as described in ["Ignoring Cache-Control and Pragma Headers in Requests."](#)
4. Bind the policy to the request-time override policy bank.

For more information, see ["Globally Binding an Integrated Caching Policy."](#)

Polling the Origin Server Every Time a Request Is Received

Updated: 2014-08-12

You can configure the NetScaler appliance to always consult the origin server before serving a stored response. This is known as Poll Every Time (PET). When the NetScaler appliance consults the origin server and the PET response has not expired, a full response from the origin server does not overwrite cached content. This property is useful when serving client-specific content.

After a PET response expires, the NetScaler appliance refreshes it when the first full response arrives from the origin server.

The Poll Every Time (PET) function works as follows:

- For a cached response that has validators in the form of an ETag or a Last-Modified header, if the response expires it is automatically marked PET and cached.
- You can configure PET for a content group.

If you configure a content group as PET, every response in the content group is marked PET. The PET content group can store responses that do not have validators. Responses that are automatically marked PET are always expired. Responses that belong to a PET content group can expire after a delay, based on how you configure the content group.

Two types of requests are affected by polling:

- Conditional Requests:** A client issues a conditional request to ensure that the response that it has is the most recent copy.

A user-agent request for a cached PET response is always converted to a conditional request and sent to the origin server. A conditional request has validators in If-Modified-Since or If-None-Match headers. The If-Modified-Since header contains the time from the Last-Modified header. An If-None-Match header contains the response's ETag header value.

If the client's copy of the response is fresh, the origin server replies with 304 Not Modified. If the copy is stale, a conditional response generates a 200 OK that contains the entire response.

- Non-Conditional Requests:** A non-conditional request can only generate a 200 OK that contains the entire response.

The following table summarizes response types based on the origin server's response

Table 3. How Responses Are Affected by Poll Every Time

Origin Server Response	Action
Send the full response	The origin server sends the response as-is to the client. If the cached response has expired, it is refreshed.
304 Not Modified	The following header values in the 304 response are merged with the cached response and the cached response is served to the client: <ul style="list-style-type: none"> Date Expires Age Cache-Control header Max-Age and S-Maxage tokens
401 Unauthorized 400 Bad Request 405 Method Not Allowed 406 Not Acceptable 407 Proxy Authentication Required	The origin's response is served as-is to the client. The cached response is not changed.

Any other error response, for example, 404 Not Found

The origin's response is served as-is to the client. The cached response is removed.

Note: The Poll Every Time parameter treats the affected responses as non-storable.

To configure poll every time by using the command line interface

At the command prompt, type:

```
add cache contentgroup <contentGroupName> -pollEveryTime YES
```

To configure poll every time by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Content Groups, and select the content group.
2. On Others tab, in the Settings group, select Poll every time (validate cached content with origin for every request) option.

PET and Client-Specific Content

The PET function can ensure that content is customized for a client. For example, a Web site that serves content in multiple languages examines the Accept-Language request header to select the language for the content that it is serving. For a multi-language Web site where English is the predominant language, all English language content can be cached in a PET content group. This ensures that every request goes to the origin server to determine the language for the response. If the response is English, and the content has not changed, the origin server can serve a 304 Not Modified to the cache.

The following example shows commands to cache English responses in a PET content group, configure a named expression that identifies English responses in the cache, and configure a policy that uses this content group and named expression. Bold is used for emphasis:

```
add cache contentgroup EnglishLanguageGroup -pollEveryTime YES
add expression containsENExpression "rule "http.res.header(\"Content-Language\").contains(
add cache policy englishPolicy -rule containsENExpression -action CACHE -storeInGroup english
bind cache policy englishPolicy -priority 100 -precedeDefRules NO
```

PET and Authentication, Authorization, and Auditing

Outlook Web Access (OWA) is a good example of dynamically generated content that benefits from PET. All mail responses (*.EML objects) have an ETag validator that enables them to be stored as PET responses.

Every request for a mail response travels to the origin server, even if the response is cached. The origin server determines whether the requestor is authenticated and authorized. It also verifies that the response exists in the origin server. If all results are positive, the origin server sends a 304 Not Modified response.

Configuring the Integrated Cache as a Forward Proxy

The integrated cache can service as a forward proxy device that passes requests to other NetScaler appliances or to other types of cache servers. You configure the integrated cache as a forward proxy by identifying the IP addresses of the cache server or servers. After configuring the forward proxy, the NetScaler appliance sends requests that contain the configured IP address on to the cache server instead of involving the integrated cache.

To configure the NetScaler as a forward cache proxy by using the command line interface

At the command prompt, type:

```
add cache forwardProxy <IPAddress> <port>
```

To configure the NetScaler as a forward cache proxy by using the configuration utility

1. Navigate to Optimization > Integrated Caching > Forward Proxy, and add a forward proxy by specifying the IP address and port number.

Default Settings for the Integrated Cache

The Citrix NetScaler integrated cache feature provides built-in policies with default settings as well as initial settings for the Default content group. The information in this section defines the parameters for the built-in policies and Default content group.

Default Caching Policies

Updated: 2013-08-26

The integrated cache has built-in policies. The NetScaler appliance evaluates the policies in a particular order, as discussed in the following sections.

You can override these built-in policies with a user-defined policy that is bound to a request-time override or response-time override policy bank.

Note that if you configured policies prior to release 9.0 and specified the `-precedeDefRules` parameter when binding the policies, they are automatically assigned to override-time bind points during migration.

Viewing the Default Policies

The built-in policy names start with an underscore (`_`). You can view the built-in policies from the command line and the administrative console using the `show cache policy` command.

Default Request Policies

You can override the following built-in request time policies by configuring new policies and binding them to the request-time override processing point. In the following policies, note that the `MAY_NOCACHE` action stipulates that the transaction is cached only when there is a user-configured or built-in `CACHE` directive at response time.

The following policies are bound to the `_reqBuiltinDefaults` policy label. They are listed in priority order.

1. Do not cache a response for a request that uses any method other than GET.

The policy name is `_nonGetReq`. The following is the policy rule:

```
!HTTP.REQ.METHOD.eq(GET)
```

2. Set a `NOCACHE` action for a request with header value that contains `If-Match` or `If-Unmodified-Since`.

The policy name is `_advancedConditionalReq`. The following is the policy rule:

```
HTTP.REQ.HEADER("If-Match").EXISTS || HTTP.REQ.HEADER("If-Unmodified-Since").EXISTS
```

3. Set a `MAY_NOCACHE` action for a request with the following header values: `Cookie`, `Authorization`, `Proxy-authorization` or a request which contains the `NTLM` or `Negotiate` header.

The policy name is `_personalizedReq`. The following is the policy rule:

```
HTTP.REQ.HEADER("Cookie").EXISTS || HTTP.REQ.HEADER("Authorization").EXISTS ||  
HTTP.REQ.HEADER("Proxy-Authorization").EXISTS || HTTP.REQ.IS_NTLM_OR_NEGOTIATE
```

Default Response Policies

You can override the following default response-time policies by configuring new policies and binding them to the response-time override processing point.

The following policies are bound to the `_resBuiltinDefaults` policy label and are evaluated in the order in which they are listed:

1. Do not cache HTTP responses unless they are of type 200, 304, 307, 203 or if the types are between 400 and 499 or between 300 and 302.

The policy name is `_uncacheableStatusRes`. The following is the policy rule:

```
!( (HTTP.RES.STATUS.EQ(200)) || (HTTP.RES.STATUS.EQ(304)) || (HTTP.RES.STATUS.  
BETWEEN(400,499)) || (HTTP.RES.STATUS.BETWEEN(300, 302)) || (HTTP.RES.STATUS.EQ  
(307)) || (HTTP.RES.STATUS.EQ(203)))
```

2. Do not cache an HTTP response if it has a `Vary` header with a value of anything other than `Accept-Encoding`.

The compression module inserts the Vary: Accept-Encoding header. The name of this expression is **_uncacheableVaryRes**. The following is the policy rule:

```
((HTTP.RES.HEADER("Vary").EXISTS) && ((HTTP.RES.HEADER("Vary").INSTANCE(1).LENGTH > 0) || (!HTTP.RES.HEADER("Vary").STRIP_END_WS.SET_TEXT_MODE(IGNORECASE).eq("Accept-Encoding"))))
```

3. Do not cache a response if its Cache-Control header value is No-Cache, No-Store, or Private, or if the Cache-Control header is not valid.

The policy name is **_uncacheableCacheControlRes**. The following is the policy rule:

```
((HTTP.RES.CACHE_CONTROL.IS_PRIVATE) || (HTTP.RES.CACHE_CONTROL.IS_NO_CACHE) || (HTTP.RES.CACHE_CONTROL.IS_NO_STORE) || (HTTP.RES.CACHE_CONTROL.IS_INVALID))
```

4. Cache responses if the Cache-Control header has one of the following values: Public, Must-Revalidate, Proxy-Revalidate, Max-Age, S-Maxage.

The policy name is **_cacheableCacheControlRes**. The following is the policy rule:

```
((HTTP.RES.CACHE_CONTROL.IS_PUBLIC) || (HTTP.RES.CACHE_CONTROL.IS_MAX_AGE) || (HTTP.RES.CACHE_CONTROL.IS_MUST_REVALIDATE) || (HTTP.RES.CACHE_CONTROL.IS_PROXY_REVALIDATE) || (HTTP.RES.CACHE_CONTROL.IS_S_MAXAGE))
```

5. Do not cache responses that contain a Pragma header.

The name of the policy is **_uncacheablePragmaRes**. The following is the policy rule:

```
HTTP.RES.HEADER("Pragma").EXISTS
```

6. Cache responses that contain an Expires header.

The name of the policy is **_cacheableExpiryRes**. The following is the policy rule:

```
HTTP.RES.HEADER("Expires").EXISTS
```

7. If the response contains a Content-Type header with a value of Image, remove any cookies in the header and cache it

The name of the policy is **_imageRes**. The following is the policy rule:

```
HTTP.RES.HEADER("Content-Type").SET_TEXT_MODE(IGNORECASE).STARTSWITH("image/")
```

You could configure the following content group to work with this policy:

```
add cache contentgroup nocookie_group -removeCookies YES
```

8. Do not cache a response that contains a Set-Cookie header.

The name of the policy is **_personalizedRes**. The following is the policy rule:

```
HTTP.RES.HEADER("Set-Cookie").EXISTS || HTTP.RES.HEADER("Set-Cookie2").EXISTS
```

Restrictions on Default Policies

You cannot override the following built-in request time policies with user-defined policies.

These policies are listed in priority order.

1. Do not cache any responses if the corresponding HTTP request lacks a GET or POST method.
2. Do not cache any responses for a request if the HTTP request URL length plus host name exceeds 1744 bytes.
3. Do not cache a response for a request that contains an If-Match header.
4. Do not cache a request that contains an If-Unmodified-Since header.

Note that this is different from the If-Modified-Since header.

5. Do not cache a response if the server does not set an expiry header.

You cannot override the following built-in response time policies. These policies are evaluated in the order in which they are listed:

1. Do not cache responses that have an HTTP response status code of 201, 202, 204, 205, or 206.
2. Do not cache responses that have an HTTP response status code of 4xx, with the exceptions of status codes 403, 404, and 410.
3. Do not cache responses if the response type is FIN terminated, or the response does not have one of the following attributes: Content-Length, or Transfer-Encoding: Chunked.

4. Do not cache the response if the caching module cannot parse its Cache-Control header.

Initial Settings for the Default Content Group

When you first enable integrated caching, the NetScaler appliance provides one predefined content group named the Default content group. The following table shows the settings for this group.

Table 1. Predefined Settings for the Default Content Group

Parameter	Description	Default Value
Hit parameters	<p>The hit parameters contain the parameter names that are significant for generating a response.</p> <p>In parameterized hit selection, NetScaler appliance matches the URL stem byte-for-byte, matches normalized values of the hit parameters, and matches the target service information.</p>	none
Invalidation Parameters	<p>These parameters mark a cached object as obsolete during parameterized selection. Specific objects, or all objects in a content group, are selected if the values of the invalidation parameters in the object and in the request are same after normalization. The invalidation parameters are a subset of the hit parameters.</p>	none
Poll Every Time	<p>Poll every time for the objects in this content group.</p>	NO
Ignore reload request	<p>Specifies whether a request can force the system to reload a cached object from the origin. To guard against Denial of Service attacks, you must set this flag to YES. To get RFC-compliant behavior you should set it to NO.</p>	YES
Remove Response Cookies	<p>If this option is disabled for a content group, and if the response contains cookies, the cookies are stored and served with every cache hit. By default, the remove cookies option is enabled for a content group, to prevent the integrated cache from storing any responses with Set-Cookie or Set-Cookie2 headers unless the response is an image.</p>	YES
Prefetch	<p>The Prefetch option refreshes an object when it is about to expire. This ensures that the object remains stale or inactive (and therefore it cannot be served) for a shorter duration of time.</p>	YES
Prefetch period	<p>This duration in seconds during which prefetch should be attempted, immediately before the object's calculated expiry time.</p>	heuristic
Maximum outstanding prefetches	<p>The number of items that can be subjected to a prefetch at a time.</p>	4294967295
Flashcache	<p>Determines whether to enable queuing of client requests and simultaneous distribution of responses to all clients in the queue.</p>	NO
Expire at last byte	<p>Determines whether to expire a cached response immediately after serving it.</p>	NO
Insert Via header	<p>Defines a string to be inserted in a Via header. By default, a Via header is inserted in all responses served from a content group. The Via header is not inserted for responses that are served by the origin server.</p>	YES

Insert Age header	The Age header contains information about the age of the object in seconds as calculated by the integrated cache.	YES
Insert ETag header	With ETag header insertion, the integrated cache does not serve full responses on repeat requests. This is done by forcing the user agent to cache the dynamic response sent by the cache the first time.	YES
Cache-control header	You can enable caching of dynamic objects in the user agent by replacing the Cache-Control headers that are inserted by the origin server. You must configure the new Cache-Control header to be inserted in the content group.	NONE
Quick abort size	If the size of an object that is being downloaded is less than or equal to the quick abort value, and a client aborts during the download, the cache stops downloading the response. If the object is larger than the quick abort size, the cache continues to download the response.	4194303 KBytes (maximum)
Minimum Response Size	You can control memory use by setting a minimum response size. Cached objects must be larger than the minimum response size.	0 KBytes
Maximum Response Size	You can control memory use by setting a maximum response size. Cached objects must be smaller than the maximum response size.	80 KBytes
Memory usage limit	Sets the maximum amount of memory that the cache can use. The effective limit is based on the available memory of the NetScaler appliance. The minimum value is 0 and the maximum value is unlimited.	UNLIMITED
Ignore caching headers in request	Disregards Cache-Control and Pragma headers in HTTP requests.	YES
MinHits configured	Number of hits that are required to qualify a response for storage in this content group.	0
Always evaluate policies	Â	NO
Pinned	By default, when the cache is full the NetScaler appliance replaces the least recently used response first. The NetScaler appliance does not apply this behavior to content groups that are marked as pinned.	NO
Lazy DNS resolution	If set to YES, DNS resolution is performed for responses only if the destination IP address in the request does not match the destination IP address of the cached response.	YES

Troubleshooting

If the integrated cache feature does not work as expected after you have configured it, you can use some common tools to access NetScaler resources and diagnose the problem.

Resources for Troubleshooting

Updated: 2013-07-22

For best results, use the following resources to troubleshoot an integrated cache issue on a NetScaler appliance:

- The relevant trace files
- The ns.conf file
- The RFC 2616 document
- A copy of the object, if possible

In addition to the above resources, the following tools expedite troubleshooting:

- The iehttpheaders or a similar utility
- The Wireshark application customized for the NetScaler trace files

Troubleshooting Integrated Caching Issues

Updated: 2013-08-02

The following are effective steps to troubleshoot the objects that are not cached:

1. Verify that the Integrated Caching feature is enabled.

Run the following command to verify the feature is enabled:

```
show ns feature
```

Following is sample output of the above command:

```
show ns feature
Feature status:
      Web Logging: OFF
      Surge Protection: OFF
      Load Balancing: ON
      Content Switching: ON
      Cache Redirection: OFF
      Sure Connect: OFF
      Compression Control: OFF
      Priority Queuing: OFF
      SSL Offloading: OFF
Global Server Load Balancing: OFF
      Http DoS Protection: OFF
      Dynamic Routing: OFF
      Content Filtering: OFF
      Integrated Caching: ON
      SSL VPN: OFF
      OSPF Routing: OFF
      RIP Routing: OFF
      BGP Routing: OFF

Done
```

The entry highlighted in boldface (for reference) in the above output indicates that the integrated caching feature is enabled. If the feature is not enabled, run the following command to enable it:

```
enable ns feature IC
```

2. Make sure that sufficient memory is available on the NetScaler appliance.

Depending on the size of the object to be cached, memory available to store the cacheable object might be insufficient. You can set the memory limit for the integrated cache either globally or for individual content groups.

Run the following command to verify the memory allocated to integrated cache globally:

```
show cache parameter
```

Following is sample output of the above command:

```
show cache parameter
Integrated cache global configuration:
Memory usage limit: 256 MBytes
Via header: NS-CACHE-6.1: 101
Verify cached object using: HOSTNAME_AND_IP
Max POST body size to accumulate: 32768
Current outstanding prefetches: 0
Max outstanding prefetches: 4294967294
Treat NOCACHE policies as BYPASS policies: YES

Done
```

The entry highlighted in boldface (for reference) in the preceding output indicates the amount of memory allocated to the integrated cache globally.

Run the following command to verify the memory allocated to an individual content group:

```
show cache contentGroup <Content_Group_Name>
```

Following is sample output of the above command:

```
show cache contentgroup content1
Name: content1
Heuristic expiry time parameter: 10 percent
Weak relative expiry time - Positive responses: 3600 secs
Weak relative expiry time - Negative responses: 600 secs
Hit parameters: NONE
Invalidation Parameters: NONE
Invalidation restricted to host: NO
Ignore parameter value case: NO
Match Request Cookies: NO
Poll Every Time: NO
Ignore reload request: YES
Remove Response Cookies: YES
Prefetch: YES
Prefetch period: heuristic
Current outstanding prefetches: 0
Max outstanding prefetches: 4294967294
Flashcache: NO
Expire at last byte: NO
Insert Via header: YES
Insert Age header: YES
Insert ETag header: YES
Cache-control header: NONE
Quick abort size: 4194303 KBytes (MAXIMUM)
Minimum Response Size: 0 KBytes
Maximum Response Size: 80 KBytes
Memory usage: 0 Bytes
Memory usage limit: 64 MBytes
Ignore caching headers in request: YES
Non-304 hits: 0
304 hits: 0
Cached objects: 0
Number of times expired/flushed: 1
MinHits configured: 0
Always evaluate policies: NO
Pinned: NO

Done
```

The entry highlighted in boldface (for reference) in the above output indicates the amount of memory allocated to the content group.

3. Verify that cacheable object is small enough to be stored within the configured memory limits.

Complete the following procedure to make space for the object to be cached:

- o Run the following command to flush the cache for the content group to which the object belongs:

```
flush cache contentGroup <Content_Group_Name>
```

- o Verify that the object is cached. If the object is cached successfully, increase the memory allocated for the content group. Otherwise, run the following command to flush the cache globally:

```
flush cache contentGroup ALL
```

- o Verify that the object is cached. If the object is cached successfully, consider increasing the global memory limit. If the object is still not cached, something else is causing the failure to cache the object.

The memory allocated to the integrated cache depends on the NetScaler appliance model. You can allocate approximately half the available memory to the integrated cache. Similarly, the maximum amount of memory you can allocate for a content group cannot be more than the memory allocated for global cache.

To increase the global memory limit for the integrated cache, run the following command:

```
set cache parameter -memLimit <Integer>
```

To increase the memory limit for a content group, run the following command:

```
set cache contentgroup <contentgroup name> -memLimit <Integer>
```

4. Verify that the cache policy is bound to an appropriate bind point, an appropriate priority is set for the policy, and an appropriate `precedeDefRules` switch is configured.

You must activate a caching policy by binding it globally. To verify that the policy is active, run the following command:

```
show cache global
```

Following is sample output of the above command:

```
show cache global
1)      Name: red_pol    State: ACTIVE    Priority: 1
      Rule: URL CONTAINS red
      Action: NOCACHE
      Precede default HTTP rules: YES
      Hits : 10

Done
```

In the output, verify the following settings:

- o **The policy is bound:** The output should contain all the active cache policies. If the cache policy for the object to be cached is not listed in the output, the policy is not yet bound. Run the following command to bind the policy globally:

```
bind cache global <Policy_Name> -priority <Integer> [-precedeDefRules YES|NO]
```

- o **The policy is Active:** If the policy is bound, verify that the state of the policy is displayed as Active. The entry indicating that the policy in the preceding output is active is the first highlighted entry in the sample output of the `show cache global` command, above. The policy is active if it is bound globally and an appropriate priority is set. Otherwise, the status of the policy is showed as Passive.
- o **An appropriate priority is assigned to the policy:** The first highlighted entry in the sample output above displays the priority of the policy. If the priority is not set, you can use the `bind` command to set the priority of the policy. Note that the higher the priority number, the lower the priority. The priority assigned to the policy enables the NetScaler appliance to determine the order in which the policy should be evaluated.

If evaluation of a particular policy fails, increase the priority of the policy so that it is evaluated before other policies. Caching policies, due to their high granularity, can be very complicated to configure. Therefore, two policies might be contradictory. As a result, only the higher-priority policy takes effect.

- o **The `precedeDefRules` switch** setting is correct: The second highlighted entry in the sample output of the `show cache global` command, above, indicates the `precedeDefRules` switch setting. This setting enables the NetScaler appliance to determine whether the policy should be evaluated before the default built-in policies, which implement the standard HTTP caching behavior, such as basing caching decisions on HTTP header fields (for example, the `If-Modified-Since` and `no-cache` fields). You can set this switch when binding the policy.

For certain types of HTTP(S) transactions, you might have to make sure that the policy precedes default HTTP rules, to force objects to be cached. Especially if requests include header fields, such as `If-Modified-Since`, or responses contain the `No-Cache` header field, you might have to make sure that the cache policy overrides the default in order for objects from these transactions to be cached. Force the policy to override default HTTP rules by rebinding the cache policy with the `-precedeDefRules YES` switch.

5. Verify the size of the object to be cached.

You can configure a content group with minimum, which by default is 0 KB, and maximum, which by default is 80 KB, response sizes for the objects to be cached. The object does not get cached if its size is not within the configured range. Additionally, verify that the cache expiry times are set to an appropriate value. For example, check for a very small time limit, such as one second.

Run the following command from the command line interface of the appliance to display the size limits and expiry times for a specific content group:

```
show cache contentGroup <Content_Group_Name>
```

Following is an example of this command's output:

```
show cache contentGroup content1
  Name: content1
  Heuristic expiry time parameter: 10 percent
  Weak relative expiry time - Positive responses: 3600 secs
  Weak relative expiry time - Negative responses: 600 secs
  Hit parameters: NONE
  Invalidation Parameters: NONE
  Invalidation restricted to host: NO
  Ignore parameter value case: NO
  Match Request Cookies: NO
  Poll Every Time: NO
  Ignore reload request: YES
  Remove Response Cookies: YES
  Prefetch: YES
  Prefetch period: heuristic
  Current outstanding prefetches: 0
  Max outstanding prefetches: 4294967294
  Flashcache: NO
  Expire at last byte: NO
  Insert Via header: YES
  Insert Age header: YES
  Insert ETag header: YES
  Cache-control header: NONE
  Quick abort size: 4194303 KBytes (MAXIMUM)
  Minimum Response Size: 0 KBytes
  Maximum Response Size: 80 KBytes
  Memory usage: 0 Bytes
  Memory usage limit: UNLIMITED
  Ignore caching headers in request: YES
  Non-304 hits: 0
  304 hits: 0
  Cached objects: 0
  Number of times expired/flushed: 0
  MinHits configured: 0
  Always evaluate policies: NO
  Pinned: NO
```

Done

In addition to the above steps for troubleshooting integrated caching issues, you can consider using the following troubleshooting techniques:

- o Depending on the configuration of a policy, there are virtually an unlimited number of reasons for the policy not getting evaluated. If you have completed the preceding steps to troubleshoot the issue, consider completing the following procedure to troubleshoot the issue further:
 - a. Flush the cache.
 - b. Verify the value of the hit parameter for the policy by running the following command:

```
show cache global
1)      Name: home_pol_1          State: ACTIVE    Priority: 99
      Rule: URL CONTAINS home
      Action: NOCACHE
      Precede default HTTP rules: NO
      Hits : 29
```

- c. Send an HTTP request for the related object from a Web browser.
- d. Run the show cache global command again and verify that the value for the hit parameter has incremented.

Depending on the policy receiving hits or not, you can determine whether the issue is due to the policy have not been configured correctly or to a more global cache setting.

Front End Optimization

Note: Front end optimization is available if you have an Enterprise or Platinum NetScaler license and are running NetScaler release 10.5 or later.

The HTTP protocols that underlie web applications were originally developed to support transmission and rendering of simple web pages. New technologies such as JavaScript and cascading style sheets (CSS), and new media types such as Flash videos and graphics-rich images, place heavy demands on front-end performance, that is, on performance at the browser level.

The NetScaler front end optimization (FEO) feature addresses such issues and reduces the load time and render time of web pages by:

- Reducing the number of requests required for rendering each page.
- Reducing the number of bytes in page responses.
- Simplifying and optimizing the content served to the client browser.

You can customize your FEO configuration to provide the best results for your users. NetScaler ADCs support numerous web content optimizations for both desktop and mobile users. The following tables describe the front-end optimizations provided by the FEO feature, and the operations performed on different types of files.

Optimizations Performed by the FEO Feature

Web Optimization	Problem	What NetScaler FEO feature does	Benefits
Inlining	Client browsers often send multiple requests to servers for loading external CSS, images, and JavaScript associated with the web page.	CSS inline JavaScript inline CSS combine	Loading the external CSS, images, and JavaScript inline with the HTML files improves page-rendering time. This optimization is beneficial for content that will be viewed only once, and for mobile devices that have limited cache sizes.
Minification	Data fetched from servers includes inessential characters such as white spaces, comments, and newline characters. The time that browsers spend in processing such data creates website latency.	CSS minification JavaScript minification Removal of HTML comments	Minified files consume less bandwidth and avoid the latency caused by special processing.
Image optimization	Mobile browsers often have slow connection speeds and limited cache memory. Downloading images on mobile clients consumes more bandwidth, processing time, and cache space, resulting in web site latency.	JPEG optimization CSS image inlining Image shrink-to attributes GIF to PNG conversion HTML image inlining	Reduces the image to the size indicated in image tag by NetScaler, enabling client browsers to load images faster.
Repositioning			

	Inefficient processing of external CSS, images, and JavaScript increases page-load time.	Image lazy loading CSS move to Head JavaScript move to end	Repositions HTML elements, to reduce rendering time for web pages and enable client browsers to load the objects faster.
Connection Management	Many browsers set limits on the number of simultaneous connections that can be established to a single domain. This can cause browsers to download webpage resources one at a time, resulting in higher browsers time.	Domain sharding CSS import to link	Overcomes the connection limitation, which improves page-rendering time by enabling client browsers to download more resources in parallel.

Web Optimizations performed on different file types

The following table lists the web optimizations that the NetScaler ADC performs on CSS, Images, JavaScript, and HTML:

Object	Optimization
CSS	<ul style="list-style-type: none"> Minify* linked CSS files. Combine multiple linked CSS files that are present within the <head> tag into a single CSS file. Convert linked CSS files to inline CSS files. Convert CSS import rule to linked CSS. Note: This optimization works if you have not defined the scope and media attributes for the <import> tag and when the <import> tag immediately follows the <style> tag. Within a linked CSS file, convert linked images to inline images. Move a CSS present within the <body> tag of an HTML page to the <head> tag. Note: The <head> tag must already be present within the HTML script.
Images	<ul style="list-style-type: none"> Optimize JPEG images by removing extraneous bytes. Reduce image size by weakening the image quality to a value specified in FEO parameters. Image shrink to attributes. Convert linked images to inline images. Note: For animated images in GIF format, only this optimization is supported. Convert non-animated images in GIF format to PNG format. Reduce the image size to that specified on the web page, if the size specified on the web page is smaller Convert images in GIF, PNG, JPEG format to WebP format Convert images in JPEG format to JPER-XR format Lazy loading.
JavaScript	<ul style="list-style-type: none"> Minify* linked JavaScript Convert linked JavaScript to inline JavaScript. Move JavaScript present in the <body> tag to the end of the <body> tag. Note: The size of the <body> tag must be lesser than 64 Kbytes. Extend cache expiry period.**

Note: The front end optimization feature supports ASCII characters only. It does not support the unicode character set.

How Front End Optimization Works

After the NetScaler ADC receives the response from the server:

1. Parses the contents of the page, creates an entry in the cache (wherever applicable), and applies the FEO policy.

For example, a NetScaler ADC can apply the following optimization rules:

- Remove white spaces or comments present within a CSS or JavaScript.
 - Combine one or more CSS files to one file.
 - Convert GIF image format to PNG format.
2. Rewrites the embedded objects and saves the optimized content in the cache, with a different signature than the one used for the initial cache entry.
 3. For subsequent requests, fetches the optimized objects from the cache, not from the server, and forwards the responses to the client.

*

Remove extraneous information such as white spaces and comments.

*

Remove extraneous information such as white spaces and comments.

**

The period during which the browser can use the cached resource without checking to see if fresh content is available on the server.

Configuring Front End Optimization

Optionally, you can change the values of the front end optimization global settings. Otherwise, begin by creating actions that specify the optimization rules to be applied to the embedded objects.

After configuring actions, create policies, each with a rule specifying a type of request for which to optimize the response, and associate the actions with the policies.

Note: The NetScaler ADC evaluates front end optimization policies at request time only, not at response time.

To put the policies into effect, bind them to bind points. You can bind a policy globally, so that it applies to all traffic that flows through the NetScaler ADC, or you can bind the policy to a load balancing or content switching virtual server of type HTTP or SSL. When you bind a policy, assign it a priority. A lower priority number indicates a higher value. The NetScaler ADC applies the policies in the order of their priorities.

Prerequisites for Front End Optimization

Front end optimization requires the NetScaler integrated caching feature to be enabled. Additionally, you must perform the following integrated caching configurations:

- o Allocate cache memory.
- o Set the maximum response size and memory limit for a default cache content group.

For more information on configuring integrated caching, see [Integrated Caching](#).

To configure front end optimization by using the command line interface

At the command prompt, do the following:

1. Enable the front end optimization feature.

```
enable ns feature FEO
```

2. Create one or more front end optimization actions.

```
add feo action <name> [-imgShrinkToAttrib] [-imgGifToPng] ...
```

Example: To add a front end optimization action for converting images in GIF format to PNG format and to extend cache expiry period:

```
add feo action allact -imgGifToPng -pageExtendCache
```

3. [Optional] Specify non-default values for front end optimization global settings.

```
set feo parameter [-cacheMaxage <integer>] [-JpegQualityPercent <integer>] [-cssInlineThresSize <integer>] [-inlineJsThresSize <integer>] [-inlineImgThresSize <integer>]
```

Example: To specify the cache maximum expiry period:

```
set feo parameter -cacheMaxage 10
```

4. Create one or more front end optimization policy.

```
add feo policy <name> <rule> <action>
```

Example: To add a front end optimization policy and associate it with the above specified *allact* action:

```
>add feo policy poll TRUE allact
>add feo policy poll "(HTTP.REQ.URL.CONTAINS(\"testsite\"))" allact1
```

5. Bind the policy to a load balancing or content switching virtual server, or bind it globally.

```
bind lb vserver <name> -policyName <string> -priority <num>
```

```
bind cs vserver <name> -policyName <string> -priority <num>
```

```
bind feo global <policyName> <priority> -type <type> <gotoPriorityExpression>
```

Example: To apply the front end optimization policy to a virtual server named "abc":

```
> bind lb vserver abc -policyName poll -priority 1 -type NONE
```

Example: To apply the front end optimization policy for all the traffic reaching the ADC:

```
> bind feo global poll 100 -type REQ_DEFAULT
```

6. Save the configuration.

```
save ns config
```

Configuring front end optimization by using the configuration utility

1. Enable the front end optimization feature using the command line interface.
2. Create a front end optimization action.

Navigate to Optimization > Front End Optimization > Actions, click Add and create a front end optimization action by specifying the relevant details.

3. [Optional] Specify the front end optimization global settings.

Navigate to Optimization > Front End Optimization, and on the right-pane, under Settings, click Change Front End Optimization settings and specify the front end optimization global settings.

4. Create a front end optimization policy.

Navigate to Optimization > Front End Optimization > Policies, click Add and create a front end optimization policy by specifying the relevant details.

5. Bind the policy to a load balancing or content switching virtual server.
 - a. Navigate to Optimization > Front End Optimization > Policies.
 - b. Select a front end optimization policy and click Policy Manager.
 - c. Under Front End Optimization Policy Manager, bind the front end optimization policy to a load balancing or content switching virtual server.

Verifying Front End Optimization Configuration

Updated: 2015-01-12

The dashboard utility displays summary and detailed statistics in tabular and graphic formats. You can view the FEO statistics to evaluate your FEO configuration.

Optionally, you can also display statistics for an FEO policy, including the number of hits that the policy counter increments during policy based FEO.

Note: For more information about statistics and charts, see the Dashboard help on the Citrix NetScaler appliance.

To View FEO statistics by using the command line interface

At the command prompt, type the following commands to display a summary of FEO statistics, FEO policy hits and details, and detailed FEO statistics, respectively:

- `stat feo`
Note: The **stat feo policy** command displays statistics only for advanced FEO policies.
- `show feo policy <name>`
- `stat feo -detail`

To view FEO statistics on the Dashboard

In the Dashboard utility, you can:

- Select **Front End Optimization** to display a summary of FEO statistics.
- Click the **Graphical View** tab to display the rate of requests processed by the FEO feature.

Sample Optimization

The following table lists some examples of content optimization actions that are applied on HTML content and the embedded objects within the HTML content.

Optimization rule	Sample
Collapse white spaces within an HTML page	Before: <pre><title>Hello, world! </title></pre>
	After: <pre><title>Hello, world!</title></pre>
Combine CSS	Before: <pre><link rel="stylesheet" type="text/css" href="sheet/abc.css"> <link rel="stylesheet" type="text/css" href="sheet/xyz.css"></pre>
	After: <pre><link rel="stylesheet" type="text/css" href="sheet/abc.css+xyz.css"></pre>
Inline CSS	Before <pre><html> <head> <link rel="sheet" href="abc.css"/> </head> <body> <div class="abc xyz"/> Hi! </div> </body> </html></pre> <p>Note: abc.css contains</p> <pre>.Alice {location: Australia;} .Tom {location: Asia;}</pre>
	After <pre><html> <head> <style> .Alice {location: Australia;} .Tom {location: Asia;} </style> </head> <body> <div class="abc xyz"> Hi! </div> </body> </html></pre>
Move CSS to head	Before: <pre><html> <head> </head> <body> <script src="abc.js" type="text/javascript"></script> <div class="monday tuesday"> Hi! </div></pre>

	<pre> <style type="text/css"> .foo { day: wednesday; } </style> <link rel="stylesheet" type="text/css" href="styles/all_styles.css"> </body> </html> </pre> <hr/> <p>After:</p> <pre> <html> <head> <style type="text/css"> .foo { day: wednesday; } </style> </head> <body> <script src="abc.js" type="text/javascript"></script> <div class="monday tuesday"> Hi! </div> <link rel="stylesheet" type="text/css" href="styles/all_styles.css"> </body> </html> </pre>
Minify JavaScript	<p>Before:</p> <pre> /* Remove this comment */ document.write("abc " + state); state += 1; // Update this. </pre> <hr/> <p>After:</p> <pre> document.write("abc "+state);state+=1; </pre>
Convert linked JavaScript to inline JavaScript	<p>Before</p> <pre> <html> <head> <script type="text/javascript" src="abc.js"></script> </head> <body> <div> Hi! </div> </body> </html> </pre> <p>Note: abc.js contains</p> <pre> /* contents of abc JavaScript file */ </pre> <hr/> <p>After</p> <pre> <html> <head> <script type="text/javascript"> /* contents of abc JavaScript file */ </script> </head> <body> <div class="abc"> Hi! </div> </body> </html> </pre>

Content Accelerator

Note: Supported on NetScaler 10.1.e and from NetScaler 10.5 onwards. It is not supported on NetScaler 10.1.

Content accelerator is a NetScaler feature that you can use in a Citrix ByteMobile T1100 deployment, to store data on a Citrix ByteMobile T2100 appliance. For more information about Citrix ByteMobile, see [How ByteMobile Works](#).

Storing data on a T2100 appliance saves bandwidth and provides faster response times, because the NetScaler does not have to connect to the server for repeated requests of the same data.

Note: Content accelerator works with a Citrix ByteMobile platinum license. Contact customer support for more information and for obtaining the license.

The following video shows the packet flow and configuration steps (also explained in subsequent topics) for the content accelerator feature.

Note: The video might take some time to load. If it does not open, click [here](#).

This document includes the following details:

- [How Content Accelerator Works](#)
- [Configuring Content Accelerator](#)

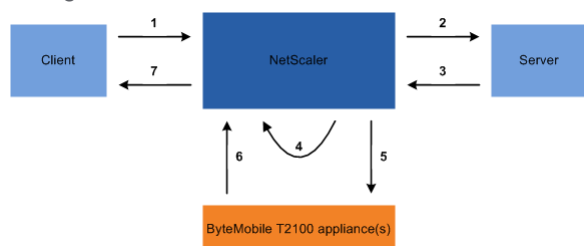
How Content Accelerator Works

Updated: 2015-05-20

When a load balancing or content switching virtual server receives a client request, the NetScaler appliance evaluates a content accelerator policy that you have bound to the virtual server. The policy filters the requests to identify the ones to which to apply the content accelerator feature.

Note: For HTTP requests, the content accelerator feature can serve partial content in response to single byte-range requests.

The following figure illustrates the operations that the appliance performs when a client request arrives at a virtual server configured to use the content accelerator feature:



The process flow is as follows:

1. Client sends request.
 2. NetScaler forwards the request to the server.
 3. Server responds with the predefined size of the response (specified by the accumResSize parameter of the add ca action command).
 4. NetScaler computes a hash of the response sent by the server.
 5. NetScaler looks up the hash on the T2100 appliance.
 6. A successful lookup indicates that the data is available and the T2100 appliance sends the data to the NetScaler.
- Note:
- When a lookup does not succeed, the NetScaler fetches all of the requested data from the server, and simultaneously serves the data to the client and updates the data on the T2100 appliance.
 - The T2100 appliance can be configured to specify the number of requests after which to cache the data.
7. NetScaler sends the response to the client.

Configuring Content Accelerator

Updated: 2013-12-16

Before configuring the content accelerator feature, you must enable it on the NetScaler appliance.

You can configure the content accelerator feature to use one or multiple T2100 appliances. You must add each T2100 appliance as a service and bind these services to a load balancing virtual server that is dedicated to distributing the load between the configured T2100 appliances.

You must also configure a content accelerator action to lookup the data on the T2100 appliance. The action must also specify the T2100 load balancing virtual server and the size of data (in KB) to be fetched from the server to calculate the hash.

The action must be bound to a content accelerator policy that defines the traffic on which to perform content acceleration. The content accelerator policy must be bound to a content switching or load balancing virtual server that receives client traffic. Alternatively, you can bind the policy globally to be applicable to all virtual servers.

Configuring content accelerator by using the command line interface

At the command prompt, do the following:

1. Enable the content accelerator feature.

```
enable ns feature ca
```

2. Identify the T2100 appliances and add each as a service on the NetScaler appliance.

```
add service <name> <IPAddress> <serviceType> <port>
```

Example:

```
> add service T2100-A 10.102.29.61 HTTP 30
> add service T2100-B 10.102.29.62 HTTP 40
> add service T2100-C 10.102.29.63 HTTP 50
```

Note: The services must be of type HTTP only.

3. Create a load balancing virtual server for the T2100 appliances. Specify the token load balancing method and the rule shown in the following syntax.

```
add lb vserver <name> <serviceType> <IPAddress> <port> -lbMethod TOKEN -rule "http.req.url.
after_str(\"/lookup/\") alt http.req.url.path.SKIP(1).PREFIX(64) "
```

Example:

```
> add lb vserver T2100-lbvserver HTTP 10.102.29.64 99 -lbMethod TOKEN -rule "http.req.v
http.req.url.path.SKIP(1).PREFIX(64) "
```

4. Bind the T2100 services to the load balancing virtual server that you created for them.

```
bind lb vserver <name> <serviceName>
```

Example:

```
> bind lb vserver T2100-lbvserver T2100-A
> bind lb vserver T2100-lbvserver T2100-B
> bind lb vserver T2100-lbvserver T2100-C
```

5. Define a content accelerator action.

```
add ca action <name> -accumResSize <KBytes> -lbvserver <string> -type lookup
```

Example:

```
> add ca action ca_action1 -type lookup -lbvserver T2100-lbvserver -accumResSize 60
```

6. Define a content accelerator policy.

```
add ca policy <name> -rule <expression> -action <name>
```

Example: To create a content accelerator policy that caches all video formats.

```
> add ca policy ca_mp4_pol -rule ns_video -action ca_action1
```

where ns_video is a built-in expression.

7. Bind the content accelerator policy to either a virtual server that receives traffic or globally to the NetScaler system.

```
bind lb vserver <name> -policyName <string>
```

```
bind cs vserver <name> -policyName <string>
```

```
bind ca global -policyName <string> -priority <num> -type <type>
```

Example: To apply the content accelerator policy to a virtual server named "traf_rec"

```
> bind lb vserver traf_rec -policyName ca_mp4_pol
```

Example: To apply the content accelerator policy for all traffic reaching the NetScaler.

```
> bind ca global -policyName ca_mp4_pol -priority 100 -type RES_DEFAULT
```

8. Save the configuration.

```
save ns config
```

Configuring content accelerator by using the configuration utility

1. Navigate to System > Settings > Configure Advanced Features and select Content Accelerator.
2. Create a service for each of the T2100 appliances.
 - a. Navigate to Traffic Management > Load Balancing > Services.
 - b. Click Add and specify the relevant details. In the Server field, make sure you specify the IP address of the T2100 appliance. In the Protocol field select HTTP.
3. Create a virtual server and bind the T2100 services to it.
 - a. Navigate to Traffic Management > Load Balancing > Virtual Servers.
 - b. Click Add and specify the relevant details.
 - c. In the Method and Persistence tab, specify the Method as Token.
 - d. In the Policies tab, specify the rule as `http.req.url.after_str("/lookup/") alt http.req.url path.SKIP(1).PREFIX(64)`.
 - e. In the Services tab, select the T2100 services that you want to bind to the virtual server.
4. Create a content accelerator action.
 - a. Navigate to Optimization > Content Accelerator > Actions.
 - b. Specify the relevant details.
5. Create a content accelerator policy.
 - a. Navigate to Optimization > Content Accelerator > Policies.
 - b. Click Add, specify the policy rule, and associate the content accelerator action.
6. Bind the content accelerator policy globally or to a virtual server.
 - a. Navigate to Optimization > Content Accelerator.
 - b. Under the Content Accelerator Policy Manager [REQUEST] or Content Accelerator Policy Manager [RESPONSE] sections, bind the content accelerator policy globally or to a virtual server.

SPDY (Speedy)

Note: Supported from NetScaler 10.1 onwards.

SPDY is an open networking experimental protocol developed by Google to reduce the time taken by a client to load a web page in a browser. An application layer protocol, SPDY changes the way in which HTTP requests and responses are handled. SPDY offers the following advantages compared to a regular HTTP transaction:

- Multiplexed requests and responsesâ€”In a single SPDY session, multiple requests from the client can be sent over a single TCP connection to the server. This reduces the number of TCP connections and also optimizes usage of each TCP connection.
- Request prioritizationâ€”When requesting services from the server, a client can assign a priority to each request.
- Header Compressionâ€”SPDY compresses the HTTP request and response headers, saving bandwidth and reducing latency.
- Server pushâ€”The server can send data to the client before the client requests it.
- Securityâ€”SPDY is secure by design, because SSL is required for SPDY connections.

NetScaler supports the SPDY/2 and SPDY/3 (from NetScaler 10.5 onwards) versions.

Note: SPDY support depends on the browser version being used.

If you use a NetScaler appliance as a SPDY gateway for your servers, the servers do not have to support SPDY. The NetScaler appliance accepts the incoming SPDY requests, converts them, and sends them to the servers as HTTP requests. It also converts the HTTP responses and sends them to the clients as SPDY responses. While the key value of SPDY is reduced bandwidth consumption and faster communication with clients, an additional benefit of the NetScaler solution is that you avoid the time consuming task of upgrading your web servers and applications to support SPDY.

To use a NetScaler appliance as a SPDY gateway, you must enable SPDY on the appliance.

This document includes the following details:

- [SPDY Requirements](#)
- [How SPDY Works over SSL](#)
- [Configuring SPDY on the NetScaler Appliance](#)
- [Troubleshooting for SPDY](#)

SPDY Requirements

Both ends of a SPDY connection must support the same version of SPDY. In addition, the clients must meet the following requirements:

- Support ZLIB compression and accept compressed data.
- Support the Next Protocol Negotiation (NPN) TLS extension, because NPN is used in the TLS handshake.

How SPDY Works over SSL

Updated: 2014-03-13

If SPDY is enabled, when the NetScaler appliance sees TLS ALPN extension with list of supported protocols in the Client Hello message, it responds with either SPDY/3 or SPDY/2 in the ALPN extension in its Server Hello.

NetScaler can also negotiate SPDY over NPN. When NetScaler sees an empty NPN extension in the Client Hello message, it responds with a list of the protocols that it supports. If SPDY is enabled on the NetScaler appliance, the appliance advertises HTTP/1.1 and SPDY/2 protocols. The client selects one protocol from this list and negotiates the protocol with the server. Because sending the negotiated protocol in plain text would raise security issues, the client sends the Change Cipher Spec notification which defines the details of the encryption for the session, followed by the Next Protocol message, which contains the encrypted protocol that the client has chosen. The client then sends the Finished message. The NetScaler appliance decrypts the Next Protocol message, and then sends a Finished message.

A session is then established, and application data can be exchanged.

Note: The NPN extension is not supported on a NetScaler FIPS appliance, and with TLS protocol versions 1.1 and 1.2.

Configuring SPDY on the NetScaler Appliance

Updated: 2014-09-15

By default, SPDY is disabled on the NetScaler appliance. After you enable SPDY, the appliance advertises SPDY/2 and/or SPDY/3 along with HTTP/1.1 during an SSL handshake. To enable SPDY on the NetScaler appliance, you must enable SPDY in the HTTP profile bound to the SSL virtual server.

To configure SPDY by using the command line interface

At the command prompt, do the following:

1. Enable SPDY on a HTTP profile.

```
set ns httpProfile <profileName> -SPDY <options>
```

Example

```
> set ns httpProfile profile1 -SPDY ENABLED
```

2. Bind the HTTP profile to a SSL virtual server.

```
set lb vserver <ssl-vserver-name> -httpProfileName <httpProfile-with-spdy>
```

Example

```
> set lb vserver SPDY_LB -httpProfileName profile1
```

Note: To apply SPDY globally, enable SPDY on the global HTTP profile (nshttp_default_profile).

You can view the statistics by using the following command:

```
stat protocol http -detail
```

To configure SPDY by using the configuration utility

1. Navigate to System > Profiles, and in the HTTP Profiles tab, update the profile on which you want to enable SPDY.
2. Navigate to Traffic Management > Load Balancing > Virtual Servers, and associate the HTTP profile to the appropriate SSL virtual server.

Troubleshooting for SPDY

If SPDY sessions are not enabled even after performing the required steps, check the following conditions.

- If the client is using a Chrome browser, SPDY might not work in some scenarios because Chrome sometimes does not initiate TLS handshake.
- If there is a forward-proxy between the client and the NetScaler appliance, and the forward-proxy doesn't support SPDY, SPDY sessions might not be enabled.
- NetScaler does not support NPN over TLS 1.1/1.2. To use SPDY, the client should disable TLS1.1/1.2 in the browser.
- Similarly, if the client wants to use SPDY, SSL2/3 must be disabled on the browser.

