

# AppExpert

Oct 13, 2015

# AppExpert

The following topics provide a conceptual reference and configuration instructions for the AppExpert and other features of the NetScaler appliance.

Action Analytics	Collects run-time statistics on the basis of pre-defined criteria. When used with policies, the feature also provides you with the infrastructure for automatic, real-time traffic optimization.
AppExpert Applications and Templates	Simplify configuration steps for the Citrix® NetScaler® appliance by using applications, application templates, NetScaler Gateway applications, and entity templates.
Entity Templates	Describes how to use entity templates to set up and configure individual NetScaler entities, such as a policy or virtual server. An entity template provides a specification and a set of defaults for the object.
AppQoE	Application level Quality of Experience (AppQoE) integrates several existing policy-based security features of the NetScaler appliance into a single integrated feature that takes advantage of a new queuing mechanism, fair queuing.
HTTP Callouts	An HTTP request that the NetScaler appliance generates and sends to an external application when certain criteria are met during policy evaluation.
Pattern Sets	Allow string matching during the evaluation of a default syntax policy.
Policies and Expressions	Rules that determine the operations that the NetScaler appliance must perform.
Rate Limiting	Defines the maximum load for a given network entity or virtual entity on the NetScaler appliance.
Responder	Bases responses on who sends the request, where it is sent from, and other criteria with security and system management implications.
Rewrite	Rewrites information in the requests or responses handled by the NetScaler appliance.
String Maps	Perform pattern matching in all NetScaler features that use the default policy syntax.

## Action Analytics

The performance of your website or application depends on how well you optimize the delivery of the most frequently requested content. Techniques such as caching and compression help accelerate the delivery of services to clients, but you need to be able to identify the resources that are requested most frequently, and then cache or compress those resources. You can identify the most frequently used resources by aggregating real-time statistics about website or application traffic. Statistics such as how frequently a resource is accessed relative to other resources and how much bandwidth is consumed by those resources help you determine whether those resources need to be cached or compressed to improve server performance and network utilization. Statistics such as response times and the number of concurrent connections to the application help you determine whether you must enhance server-side resources.

If the website or application does not change frequently, you can use products that collect statistical data, and then manually analyze the statistics and optimize the delivery of content. However, if you do not want to perform manual optimizations, or if your website or application is dynamic in nature, you need infrastructure that can not only collect statistical data but can also automatically optimize the delivery of resources on the basis of the statistics. On the NetScaler appliance, this functionality is provided by the action analytics feature. The feature operates on a single NetScaler appliance and collects run-time statistics on the basis of criteria that you define. When used with NetScaler policies, the feature also provides you with the infrastructure that you need for automatic, real-time traffic optimization.

When configuring the action analytics feature, you specify the request attributes for which you want to collect statistical data (for example, URLs and HTTP methods) by configuring default syntax expressions in an entity called a selector. Then, you configure an identifier to configure settings such as the sampling interval and sample count. You also configure a policy that enables the appliance to evaluate traffic as specified by the selector-identifier pair. Finally, you bind the policy to a bind point to begin collecting statistics.

The appliance also provides you with a set of built-in selectors, identifiers, and responder policies that you can use to get started with the feature.

The appliance aggregates the following statistics:

- The number of requests.
- The bandwidth consumed by the requests.
- The response time.
- The number of concurrent connections.

You can configure the feature to perform run-time sorting of the records on an attribute of your choice. You can view the statistical data by using either the command-line interface or the Stream Sessions tool in the configuration utility.

## Configuring a Selector

A selector is a filter for identifying requests. It consists of up to five individual default syntax expressions that identify request attributes such as the client IP address and the URL in the request. Each expression is a non-compound default syntax expression and is considered to be in an AND relationship with the other expressions. Following are some examples of selector expressions:

- `HTTP.REQ.URL`
- `CLIENT.IP.SRC`
- `HTTP.RES.BODY(1000).AFTER_STR(\"<string>\").BEFORE_STR(\"<string>\")`
- `CLIENT.IP.SRC.SUBNET(24)`

Selectors are used in rate limiting and action analytics configurations. A selector is optional in a rate limiting configuration, but is required in a action analytics configuration.

The order in which you specify parameters is significant. For example, if you configure an IP address and a domain (in that order) in one selector, and then specify the domain and the IP address (in the reverse order) in another selector, the NetScaler considers these values to be unique. This can lead to the same transaction being counted twice. Also, if multiple policies invoke the same selector, the NetScaler, again, can count the same transaction more than once.

If you modify an expression in a selector, you may get an error if any policy that invokes it is bound to a new policy label or bind point. For example, suppose that you create a selector named `myLimitSelector1`, invoke it from `myLimitID1`, and invoke the identifier from a DNS policy named `dnsRateLimit1`. If you change the expression in `myLimitSelector1`, you might receive an error when binding `dnsRateLimit1` to a new bind point. The workaround is to modify these expressions before creating the policies that invoke them.

The NetScaler appliance provides the following built-in selectors for some of the most common use cases:

Table 1. Built-in Selectors

Selector	Selector Expressions
Top_URL	<code>HTTP.REQ.URL</code>
Top_CLIENTS	<code>CLIENT.IP.SRC</code>
Top_URL_CLIENTS_LBVSERVER	<ol style="list-style-type: none"><li>1. <code>HTTP.REQ.URL</code></li><li>2. <code>CLIENT.IP.SRC</code></li><li>3. <code>HTTP.REQ.LB_VSERVER.NAME</code></li></ol>
Top_URL_CLIENTS_CSVSERVER	<ol style="list-style-type: none"><li>1. <code>HTTP.REQ.URL</code></li><li>2. <code>CLIENT.IP.SRC</code></li><li>3. <code>HTTP.REQ.CS_VSERVER.NAME</code></li></ol>
Top_MSSQL_QUERY_DB_LBVSERVER	<ol style="list-style-type: none"><li>1. <code>MSSQL.REQ.QUERY.TEXT</code></li><li>2. <code>MSSQL.REQ.LB_VSERVER.NAME</code></li></ol>
Top_MYSQL_QUERY_DB_LBVSERVER	<ol style="list-style-type: none"><li>1. <code>MYSQL.REQ.QUERY.TEXT</code></li><li>2. <code>MYSQL.REQ.LB_VSERVER.NAME</code></li></ol>

You can also configure a selector with expressions that identify the request attributes of your choice. For example, you might want to create a record for a request that arrives with a specific header. To evaluate the header, you can add `HTTP.REQ.HEADER(\"<header_name>\")` to the selector that you intend to use.

## To configure a selector by using the command line interface

At the command prompt, type the following commands to configure a selector and verify the configuration:

- add stream selector `<name> <rule> ...`
- show stream selector

### Example



```
> add stream selector myselector HTTP.REQ.URL CLIENT.IP.SRC
Done
> show stream selector myselector
    Name: myselector
    Expressions:
        1) HTTP.REQ.URL
        2) CLIENT.IP.SRC
Done
>
```

## To modify or remove a selector by using the command line interface

- To modify a selector, type the set stream selector command, the name of the selector, and the rule parameter with the expressions. Enter the existing expressions that you want to retain, along with the new expressions that you want to add.
- To remove a selector, type the rm stream selector command and the name of the selector.

## To configure a selector by using the configuration utility

1. Navigate to AppExpert > Action Analytics > Selectors.
2. In the details pane, do one of the following:
  - To create a selector, click Add.
  - To modify a selector, select the selector, and then click Open.
3. In the Create Limit Selector or Configure Limit Selector dialog box, set one or more of the following parameters:
  - Name
  - Expressions
 

To add the expression to the selector configuration, click Add. To remove an expression from the selector configuration, in the Expression box, select the expression, and then click Remove.

Note: In the Expressions box, enter a valid parameter. For example, enter `HTTP`. Then, enter a period after this parameter. A drop-down menu appears. The contents of this menu provide the keywords that can follow the initial keyword that you entered. To select the next keyword in this expression prefix, double-click the selection in the drop-down menu. The Expressions text box displays both the first and second keywords for the expression prefix, for example, `HTTP.REQ`. Continue adding expression components until the complete expression is formed.
4. Click Add.
5. Continue adding up to five non-compound expressions.
6. Click Create or OK.

## Configuring a Stream Identifier

You configure a stream identifier to specify parameters for collecting statistical data from requests identified by a given selector. An identifier specifies the selector to be used, the statistics collection interval, the sample count, and the field on which the records are to be sorted.

The NetScaler appliance includes the following built-in stream identifiers for common use cases. All the built-in identifiers specify a sample count of 1 and an interval of 1 minute. Additionally, they sort the data on the `REQUESTS` attribute. They differ only in being associated with different built-in selectors. Each built-in identifier is associated with a built-in selector of the same name (for example, the built-in identifier `Top_URL` is associated with the built-in selector `Top_URL`). Following are the built-in identifiers:

- `Top_URL`
- `Top_CLIENTS`
- `Top_URL_CLIENTS_LBVSERVER`
- `Top_URL_CLIENTS_CSVSERVER`
- `Top_MSSQL_QUERY_DB_LBVSERVER`
- `Top_MYSQL_QUERY_DB_LBVSERVER`

For more information about the built-in selectors, see ["Configuring a Selector."](#)

Note: The maximum length for storing string results of selectors (for example, `HTTP.REQ.URL`) is 60 characters. If the string (for example, `URL`) is 1000 characters long, of which 50 characters are enough to uniquely identify a string, use an expression to extract only the required 50 characters.

You cannot modify a built-in identifier's configuration. However, you can create an identifier with a configuration of your choice.

## To configure a stream identifier by using the command line interface

At the command prompt, type the following commands to configure a stream identifier and verify the configuration:

- `add stream identifier <name> <selectorName> [-interval <positive_integer>] [-SampleCount <positive_integer>] [-sort <sort>]`
- `show stream identifier <name>`

### Example

```
> add stream identifier myidentifier Top_URL -interval 10 -sampleCount 100
Done
```

## To configure a stream identifier by using the configuration utility

1. Navigate to AppExpert > Action Analytics > Stream Identifiers.
2. In the details pane, do one of the following:
  - To create a stream identifier, click Add.
  - To modify a stream identifier, select the identifier, and then click Open.
3. In the Configure Stream Identifier dialog box, set one or more of the following parameters:
  - Name
  - Selector
  - Interval
  - Sample Count
  - Sort
4. Click Create or OK, and then click Close.

## Viewing Statistics

You can view the collected statistics in tabular format in the command-line interface and in graphical format in the configuration utility.

The following table describes the collected statistics:

Table 1. Statistical Information Displayed for a Stream Identifier

Statistics	Column name in the output of the stat stream identifier <identifier name> command	Description
Number of requests	Req	The number of requests for which records were created in the last <interval> number of minutes.
Bandwidth consumed	BandW	<p>The total bandwidth consumed by the requests that were received in the last &lt;interval&gt; number of minutes. The total bandwidth of a request is the bandwidth consumed by the request and its response.</p> <p>The value is rounded off to the next higher or next lower integer value. Consequently, it might differ slightly from the expected value. For example, if a request's total bandwidth consumption is 2.2 KB, one instance of the request might be shown as having consumed 2 KB and two instances might be shown as having consumed 4 KB, but three instances might be shown as having consumed 7 KB.</p>
Response time	RspTime	The average response time for all the requests received in the last <interval> number of minutes.
Concurrent connections	Conn	The total number of concurrent connections that are currently open.

## To view the statistical data collected for a stream identifier by using the command line

At the command prompt, type:

```
stat stream identifier <name> [<pattern> ...] [-detail] [-fullValues] [-ntimes <positive_integer>] [-logFile <input_filename>] [-sortBy <sortBy> [<sortOrder>]]
```

### Examples

Example 1 sorts the output on the BandW column, in the descending order. Example 2 sorts the output in Example 1, on the Req column, and in the ascending order

#### Example 1

```
> stat stream identifier myidentifier -sortBy BandW Descending -fullValues
Stream Session statistics
      Req      BandW
User1      508    125924
User2     5020    12692
User3     2025     4316

      RspTime      Conn
User1      5694         0
User2       109         0
User3         3         0
Done
```

#### Example 2

```
> stat stream identifier myidentifier -sortBy Req Ascending -fullValues
Stream Session statistics
      Req      BandW
```

User1	508	125924
User3	2025	4316
User2	5020	12692
	RspTime	Conn
User1	5694	0
User3	3	0
User2	109	0
Done		

## To view the statistical data collected for a stream identifier by using the configuration utility

1. Navigate to AppExpert > Action Analytics > Stream Identifiers.
2. Select the stream identifier whose sessions you want to view, and then click Stream Sessions. For information about how you can group the output on the basis of the values collected for various selector expressions, see "[Grouping Records on Attribute Values.](#)"

## Grouping Records on Attribute Values

Statistical information such as the number of times a particular URL has been accessed overall and per client, and the total number of GET and POST requests per client can provide valuable insights into whether any of your resources need to be expanded to meet the demand or be optimized for delivery. To obtain such statistics, you must use an appropriate set of selector expressions, and then use the pattern parameter in the stat stream identifier command. The grouping is based on the pattern that is specified in the command. Grouping can be performed concurrently on the values of multiple expressions.

In the command-line interface, you can group the output by using patterns of your choice. In the configuration utility, the pattern depends on the choices you make when drilling down through the values of various selector expressions. For example, consider a selector that has the expressions `HTTP.REQ.URL`, `CLIENT.IP.SRC`, and `HTTP.REQ.LB_VSERVER.NAME`, in that order. The statistics home page displays icons for each of these expressions. If you click the icon for `CLIENT.IP.SRC`, the output is based on the patterns `* ? *`. The output displays statistics for each client IP address. If you click an IP address, the output is based on the patterns `* <IP address> ?` and `? <IP address> *` where `<IP address>` is the IP address you selected. In the resulting output, if you click a URL, the pattern used is `<URL> <IP address> ?`.

## To group the records on the values of selector expressions by using the command line interface

At the command prompt, enter the following command to group the records on the basis of a selector expression:

```
stat stream identifier <name> [<pattern> ...]
```

### Examples

Each example uses a different pattern to demonstrate the effect of the pattern on the output of the stat stream identifier command. The selector expressions are `HTTP.REQ.URL` and `HTTP.REQ.HEADER("UserHeader")`, in that order. The requests contain a custom header whose name is `UserHeader`. Note that in the examples, a given statistical value changes as determined by the grouping, but the sum total of the values for a given field remains the same.

#### Example 1

In the following command, the pattern used is `? ?`. The appliance groups the output on the values collected for both selector expressions. The row headers consist of the expression values separated by a question mark (?). The row with the header `/mysite/mypage1.html?Ed` displays statistics for requests made by user `Ed` for the URL `/mysite/mypage1.html`.

```
> stat stream identifier myidentifier ? ? -fullValues
Stream Session statistics
```

	Req	BandW
/mysite/mypage2.html?Grace	1	2553
/mysite/mypage1.html?Grace	2	4
/mysite/mypage1.html?Ed	8	16
/mysite/mypage2.html?Joe	1	2554
/mysite/mypage1.html?Joe	5	10
/mysite/?Joe	1	4

	RspTime	Conn
/mysite/mypage2.html?Grace	0	0
/mysite/mypage1.html?Grace	0	0
/mysite/mypage1.html?Ed	0	0
/mysite/mypage2.html?Joe	0	0
/mysite/mypage1.html?Joe	0	0
/mysite/?Joe	6	0

```
Done
```

#### Example 2

In the following command, the pattern used is `* ?`. The appliance groups the output on the values accumulated for the second expression `HTTP.REQ.HEADER("UserHeader")`. The rows display statistics for all requests made by users `Grace`, `Ed`, and `Joe`.

```
> stat stream identifier myidentifier * ?
Stream Session statistics
```

	Req	BandW	RspTime	Conn
Grace	3	2557	0	0
Ed	8	16	0	0

```
Joe          7      2568      6      0
Done
```

### Example 3

In the following command, the pattern used is ? \*, which is the default pattern. The output is grouped on the values collected for the first selector expression. Each row displays statistics for one URL.

```
> stat stream identifier myidentifier ? * -fullValues
Stream Session statistics

                               Req          BandW
/mysite/mypage2.html           2          5107
/mysite/mypage1.html          15           30
/mysite/                       1           4

                               RspTime         Conn
/mysite/mypage2.html           0           0
/mysite/mypage1.html           0           0
/mysite/                       6           0
Done
```

### Example 4

In the following command, the pattern used is \* \*. The appliance displays one set of collective statistics for all the requests received, with no row title.

```
> stat stream identifier myidentifier * *
Stream Session statistics

      Req    BandW  RspTime    Conn
      18     5141      6       0

Done
```

### Example 5

In the following command, the pattern is /mysite/mypage1.html \*. The appliance displays one set of collective statistics for all the requests received for the URL /mysite/mypage1.html, with no row title.

```
> stat stream identifier myidentifier /mysite/mypage1.html *
Stream Session statistics

      Req    BandW  RspTime    Conn
      15     30      0       0

Done
```

## To group the records on the values of selector expressions by using the configuration utility

1. Navigate to AppExpert > Action Analytics > Stream Identifiers.
2. In the details pane, click the stream identifier for which you want to view statistics, and then click Stream Sessions.
3. On the Home page, click the icon for the stream selector by which you want to group the output.
4. To return to the Home page from the statistics page for a selector expression, click Home.
5. To view statistics for the value of a given selector expression, click the value. You can repeat this step for a selector expression value in each subsequent output until you obtain the statistics you want.

## Clearing a Stream Session

You can flush all the records that have been accumulated for a stream identifier.

### To clear a stream session by using the command line interface

At the command prompt, enter the following commands to clear a stream session and verify the results:

- clear stream session <name>
- stat stream identifier <name>

#### Example

This example uses the stat stream identifier command first, so that a comparison can be made with the stat stream identifier command that is used for verifying the result of the clear stream session command.

```
>stat stream identifier myidentifier
Stream Session statistics
      Req      BandW  RspTime      Conn
/aed....html      2          0          0          0
/                636      303      12          0
  Done
>clear stream session myidentifier
  Done
>stat stream identifier myidentifier
  Done
```

### To clear a stream session by using the configuration utility

1. Navigate to AppExpert > Action Analytics > Stream Identifiers.
2. Select the stream identifier whose sessions you want to clear, and then click Clear Sessions.

## Configuring a Policy for Analyzing and Optimizing Traffic

To put the selector-identifier pair in your action analytics configuration into effect, you must associate the pair with the point in the traffic flow at which you want to collect statistics. You can do so by configuring a default syntax policy and referencing the stream identifier from the policy rule. You can use compression policies, caching policies, rewrite policies, application firewall policies, responder policies, and any other policies whose action is based on a Boolean expression.

The action analytics feature introduces a set of default syntax expressions and functions for collecting and evaluating data. The expression `ANALYTICS.STREAM(<identifier_name>)` is used for referencing the identifier that you want to use. The expression `COLLECT_STATS` is used to collect statistical data. Functions such as `IS_TOP(<uint>)` and `IS_TOP_FREQUENTS(<uint>)` are used for making automatic, real-time traffic optimization decisions.

- **IS\_TOP(<number>).** Finds if a given object is in the top <number> of elements. For example, is the element among the top 10 elements. When multiple elements have the count, they are considered to be similar in nature. The sort function must be turned on to avoid an undef condition.
- **IS\_TOP\_FREQUENTS(<frequency>).** Finds if a given object is in the top <frequency> of the elements that are in the top elements. For example, is the element among the top 50% of all the top elements maintained. Elements having the same values are considered similar in nature. The sort function must be turned on to avoid an undef condition.

It is your policy configuration that determines whether the NetScaler appliance must only collect data from traffic or also perform an action. If the appliance must only collect statistical data, you can configure a policy with the rule `ANALYTICS.STREAM(<identifier_name>).COLLECT_STATS` and the action `NOOP`. The `NOOP` policy must be the policy with the highest priority at the bind point. This policy is sufficient if you are only collecting statistics. Traffic optimization decisions, such as what to compress or cache, must be based on manual, periodic evaluation of the statistical data.

If, in addition to collecting statistics, the appliance must also perform an action on the traffic, you must configure the `gotoPriorityExpression` parameter of the `NOOP` policy such that another policy that has the desired rule and action is evaluated subsequently. This second policy must have a rule that begins with the `ANALYTICS.STREAM(<identifier_name>)` prefix and a function that evaluates the data.

Following is an example of two responder policies that are configured and bound globally. The policy `responder_stat_collection` enables the appliance to collect statistics based on the identifier, `myidentifier`. The policy `responder_notify` evaluates the data that is collected.

### Example

```
> add responder action send_notification respondwith '"You are in the Top 10 list for bandwi
Done
> add responder policy responder_stat_collection' ANALYTICS.STREAM("myidentifier").COLLECT_S
Done
> add responder policy responder_notify 'ANALYTICS.STREAM("myidentifier").BANDWIDTH.IS_TOP(1
Done
> bind responder global responder_stat_collection 10 NEXT
Done
> bind responder global responder_notify 20 END
Done
```



## Use Case: Limiting Bandwidth Consumption per User or Client Device

Your web site, application, or file hosting service has finite network and server resources available to it to serve all its users. One of the most important resources is bandwidth. Substantial bandwidth consumption by only a subset of the user base can result in network congestion and reduced resource availability to other users. To prevent network congestion, you might have to limit a client's bandwidth consumption by using temporary service denial techniques such as responding to a client request with an HTML page if it has exceeded a preconfigured bandwidth value over a fixed time period leading up to the request.

In general, you can regulate bandwidth consumption either per client device or per user. This use case demonstrates how you can limit bandwidth consumption per client to 100 MB over a time period of one hour. The use case also demonstrates how you can regulate bandwidth consumption per user to 100 MB over a time period of one hour, by using a custom header that provides the user name. In both cases, the tracking of bandwidth consumption over a moving time period of one hour is achieved by setting the interval parameter in the stream identifier to 60 minutes. The use cases also demonstrate how you can import an HTML page to send to a client that has exceeded the limit. Importing an HTML page not only simplifies the configuration of the responder action in these use cases, but also simplifies the configuration of all responder actions that need the same response.

### To limit bandwidth consumption per user or client device by using the command line interface

In the command-line interface, perform the following tasks to configure action analytics for limiting a client's or user's bandwidth consumption. Each step includes sample commands and their output.

1. **Set up your load balancing configuration.** Configure load balancing virtual server `mysitevip`, and then configure all the services that you need. Bind the services to the virtual server. The following example creates ten services and binds the services to `mysitevip`.

```
> add lb vserver mysitevip HTTP 192.0.2.17 80
Done
> add service service[1-10] 192.0.2.[240-249] HTTP 80
service "service1" added
service "service2" added
service "service3" added
.
.
.
service "service10" added
Done
> bind lb vserver vserver1 service[1-10]
service "service1" bound
service "service2" bound
service "service3" bound
.
.
.
service "service10" bound
Done
```

2. **Configure the stream selector.** Configure one of the following stream selectors:
  - o To limit bandwidth consumption per client, configure a stream selector that identifies the client IP address.

```
> add stream selector myselector CLIENT.IP.SRC
Done
```

- o To limit bandwidth consumption per user on the basis of the value of a request header that provides the user name, configure a stream selector that identifies the header. In the following example, the name of the header is `UserHeader`.

```
> add stream selector myselector HTTP.REQ.HEADER("UserHeader")
Done
```

3. **Configure a stream identifier.** Configure a stream identifier that uses the stream selector. Set the interval parameter to 60 minutes.

```
> add stream identifier myidentifier myselector -interval 60 -sampleCount 1 -sort BANDW
Done
```

4. **Configure the responder action.** Import the HTML page that you want to send to users or clients that have exceeded the bandwidth consumption limit, and then use the page in responder action `crossed_limits`.

```
> import responder htmlpage http://192.0.2.20:80/stdpages/wait.html crossed_limits.html
This operation may take some time, Please wait...
```

Done

```
> add responder action crossed_limits respondwithhtmlpage crossed_limits.html
Done
```

5. **Configure the responder policies.** Configure responder policy `myrespol1` with the rule `ANALYTICS.STREAM("myidentifier").COLLECT_STATS` and the action `NOOP`. Then, configure policy `myrespol2` for determining whether a client or user has crossed the 100 MB limit. The policy `myrespol2` is configured with the responder action `crossed_limits`.

```
> add responder policy myrespol1 'ANALYTICS.STREAM("myidentifier").COLLECT_STATS' NOOP
Done
> add responder policy myrespol2 'ANALYTICS.STREAM("myidentifier").BANDWIDTH.GT(1048576)'
Done
```

6. **Bind the responder policies to the load balancing virtual server.** The policy `myrespol1`, which only collects statistical data, must have the higher priority and a `GOTO` expression of `NEXT`.

```
> bind lb vserver mysitevip -policyName myrespol1 -priority 1 -gotoPriorityExpression N
Done
> bind lb vserver mysitevip -policyName myrespol2 -priority 2 -gotoPriorityExpression E
Done
```

7. **Test the configuration.** Test the configuration by sending test HTTP requests, from multiple clients or users, to the load balancing virtual server and using the `stat stream identifier` command to view the statistics that are collected for the specified identifier. The following output displays statistics for clients.

```
> stat stream identifier myidentifier -sortBy BandW â€"fullValues
Stream Session statistics
```

	Req	BandW
192.0.2.30	5000	3761
192.0.2.31	29	2602
192.0.2.32	25	51

	RspTime	Conn
192.0.2.30	2	0
192.0.2.31	0	0
192.0.2.32	0	0

Done

```
>
```

# AppExpert Applications and Templates

An AppExpert application is a collection of configuration information that you set up on the Citrix NetScaler appliance for securing and optimizing traffic for a Web application, such as Microsoft SharePoint. Managing AppExpert applications is simplified by a graphical user interface (GUI) that allows you to specify application traffic subsets and a distinct set of security and optimization policies for processing each traffic subset. Additionally, it consolidates all deployment tasks in one view, so you can quickly configure target IP addresses for clients and specify host servers.

Prebuilt application templates for widely used Web applications, such as Microsoft Outlook Web Access and Microsoft SharePoint, are available on the AppExpert Templates page of the Citrix Community website at "<http://community.citrix.com/display/ns/AppExpert+Templates>."

Each prebuilt template provides you with an initial configuration for managing the associated Web application. You can customize prebuilt application templates for your organization. If a prebuilt application template does not suit your requirements, you can create a custom application without using a template.

Regardless of whether you use a prebuilt application template or you create a custom application, you can export the configuration to a template file. You can then share the template with other administrators or import the template to other NetScaler appliances that require a similar AppExpert application configuration.

To get started with an AppExpert application, you must first obtain the appropriate application template and import the template to the NetScaler appliance. After the AppExpert application is set up, you must verify that the application is working correctly. If required, you can customize the configuration to suit your requirements.

Periodically, you can verify and monitor the configuration by viewing the hit counters for various application components, statistics, and the Application Visualizer. You can also configure authentication, authorization, and auditing (AAA) policies for the application.

## AppExpert Application Terminology

Updated: 2013-08-30

Following are the terms used in the AppExpert applications feature and the descriptions of the entities for which the terms are used:

**Public Endpoint.** The IP address and port combination at which the NetScaler appliance receives client requests for the associated web application. A public endpoint can be configured to receive either HTTP or secure HTTP (HTTPS) traffic. All client requests for the web application must be sent to a public endpoint. An AppExpert application can be assigned multiple endpoints. You configure public endpoints after you import a template.

**Application Unit.** An AppExpert application entity that processes a subset of web application traffic and load balances a set of services that host the associated content. The subset of traffic that an application unit must manage is defined by a rule. Each application unit also defines its own set of traffic optimization and security policies for the requests and responses that it manages. The NetScaler services associated with these policies are Compression, Caching, Rewrite, Responder, and application firewall.

By default, every AppExpert application with at least one application unit includes a default application unit, which cannot be deleted. The default application unit is not associated with a rule for identifying requests and is always placed last in the order of application units. It defines a set of policies for processing any request that does not match the rules that are configured for the other application units, thereby ensuring that all client requests are processed.

Application units and their associated rules, policies, and actions are included in AppExpert application templates.

**Service.** The combination of the IP address of the server that hosts the web application instance and the port to which the application is mapped on the server, in the format <IP address>:<Port>. A web application that serves a large number of requests is usually hosted on multiple servers. Each server is said to host an instance of the web application, and each such instance of the web application is represented by a service on the NetScaler appliance. Services are deployment-specific, and are therefore not included in templates. You must configure services after you import a template.

**Application Unit Rule.** Either a classic expression or a default syntax expression that defines the characteristics of a traffic subset for an application unit. The following example rule is a default syntax expression that identifies a traffic subset that consists of four image types:

```
HTTP.REQ.URL.SUFFIX.EQ("bmp") || HTTP.REQ.URL.SUFFIX.EQ("gif") || HTTP.REQ.URL.SUFFIX.EQ("png") || HTTP.REQ.URL.SUFFIX.EQ("jpg")
```

For more information about default syntax expressions and classic policy expressions, see "[Policies and Expressions](#)."

**Traffic Subset.** A set of client requests that require a common set of traffic optimization and security policies. A traffic subset is managed by an application unit and is defined by a rule.

## How an AppExpert Application Works

When the endpoint receives a client request, the NetScaler appliance evaluates the request against the rule that is configured for the topmost application unit. If the request satisfies this rule, the request is processed by the policies that are configured for the application unit, and then forwarded to a service. The choice of service depends on which services are configured for the application, and on settings such as the load balancing algorithm and persistence method configured for the application unit.

If the request does not satisfy the rule, the request is evaluated against the rule for the next topmost application unit. In this order, the request is evaluated against each application unit rule until the request satisfies a rule. If the request does not satisfy any of the configured rules, it is processed by the default application unit, which is always the last application unit.

You can configure multiple public endpoints for an AppExpert application. In such a configuration, by default, each application unit processes requests received by all the public endpoints and load balances all the services that are configured for the application. However, you can specify that an application unit processes traffic from only a subset of the public endpoints and load balances only a subset of the services that are configured for the AppExpert application.

## Getting Started with an AppExpert Application

The process of setting up an AppExpert application begins with downloading the appropriate AppExpert application template from the Citrix Community Web site at "<http://community.citrix.com/display/ns/AppExpert+Templates>." The template that you need depends on the NetScaler release running on your appliance.

After you download the template, you must import the template to the NetScaler appliance, configure deployment settings, and then verify the configuration to make sure that the AppExpert application is working as expected.

## Importing an AppExpert Application Template

Updated: 2013-08-30

You can either import the template file directly from your local computer or upload the template to the appliance and then import it. For more information about uploading a template to the NetScaler appliance, see "[Uploading and Downloading Template Files](#)."

During import, along with the template file that you specify in the AppExpert Template Wizard, you can include a deployment file that contains deployment details. If you choose to include a deployment file, you do not have to provide any additional information. All application-configuration information is imported from the template file, and all deployment-specific information for the application is imported from the deployment file. The NetScaler appliance imports all configuration settings from the deployment file through the NITRO API, and the wizard displays the configuration summary screen for your verification. If you do not include a deployment file, the wizard displays screens on which you can specify deployment information. During import, if an error occurs, any changes are automatically rolled back, preserving the configuration that was in place before you attempted to import the AppExpert application. For more information about the format of application templates and deployment files, see "[Understanding NetScaler Application Templates and Deployment Files](#)." For more information about how the template and deployment files are imported through the NITRO API, see "[NITRO API](#)."

AppExpert applications and the deployment files created from them support two or more endpoints. However, when importing an AppExpert template file, if you do not include a deployment file, the AppExpert Template Wizard displays a screen on which you can configure a maximum of two public endpoints—one endpoint of type HTTP and one endpoint of type HTTPS—for the application. So, if you want more than two endpoints, you have to configure additional endpoints after you create the application. You can then export the application to obtain a deployment file that contains all the configured endpoints.

If the public endpoint that you configure during import uses the HTTPS protocol, and you are not providing a deployment file, you must also specify a server certificate for the public endpoint. Additionally, if variables have been configured for the template, a Specify Variable Values page appears in the wizard. On this page, you can choose to specify new values for the variables. Configuring deployment settings (a minimum of one public endpoint and one or more services) during template import is not mandatory; you can choose to skip these steps during import and, instead, configure these settings after you import the template. Note, however, that you can configure additional AppExpert application features, such as AAA, only after you specify at least one public endpoint.

For more information about configuring endpoints after you import a template, see "[Configuring Public Endpoints](#)." For more information about configuring services and service groups after you import a template, see "[Configuring Services and Service Groups](#)." For more information about configuring variables for a NetScaler application, see "[Creating Variables in Application Templates](#)."

### To import an AppExpert application template to the NetScaler appliance

1. Navigate to AppExpert > Applications.
2. In the details pane, click Applications, and then click Import.
3. Follow the instructions in the AppExpert Template Wizard.

## Verifying and Testing the Configuration




Updated: 2013-08-30

Verification is an important step in the process of setting up the NetScaler application. Before you proceed with other configuration tasks, you must verify that the state of the entities, such as endpoints and application units, are UP, and then test the entities for correct processing.

### Verifying the Configuration

The graphical user interface (GUI) includes icons that indicate the states of the entities in the AppExpert application. These icons are displayed for applications and application units and are based on the health checks that the NetScaler appliance performs periodically on services and entities. The following table lists the icons and describes their meanings.

Table 1. Descriptions of State Indicator Icons

Icon	Entity	Indicates that
	Application	At least one public endpoint is up. The application will accept client requests from the public endpoints that are up.
	Application unit	The application unit is up. The application unit is up when at least one service or service group is up.
	Application	The public endpoint is out of service (disabled). This indicator is displayed when only one public endpoint is configured for the AppExpert application.
	Application	All the endpoints that are configured for the application are out of service. This indicator is displayed only when multiple endpoints are configured for the application.
	Application unit	All the services configured for the application unit are down.

You must ensure that the icons for each application and its application units are green at all times. If the icon that is displayed for an application is not green, verify that you have configured the public endpoints correctly. If the icon that is displayed for an application unit is not green, verify that the services are configured correctly. However, note that a green indicator does not mean that the state of all associated entities is UP. It only means that the application has sufficient resources (endpoints and services) to serve client requests. To verify that the state of all associated entities is UP, check the health of all the entities on the statistics page for the application. For more information about viewing the application statistics page, see "[Viewing Application Statistics](#)."

## Testing the Configuration by Using Hit Counters

You can test the configuration by sending test HTTP requests for web application content through the NetScaler appliance, and then verifying that the requests are being processed by the right application units, by viewing the hit counters for the various AppExpert application entities. For example, to verify that the endpoint is receiving requests, you can view the hit counter for the AppExpert application. To verify that the configured application unit rules are being matched as expected, you can view the hit counters for the AppExpert application units.

Note: To view hit counters for policies and actions that are configured for AppExpert applications, you must go to the associated feature node. For example, to view the hit counter for a Rewrite policy that is configured for an AppExpert application, you must go to the Rewrite feature node in the NetScaler configuration utility.

For a test example, consider an AppExpert application that includes an application unit called "WebPages" for processing web page content, and an application unit called "Images" for processing images. In this example, the rule that is configured for the WebPages application unit includes an expression that checks whether an HTML file is being requested. The Images application unit includes an expression that checks whether an image file is being requested.

Consider an HTML file called sitehome.html, located at /var/www/html/myapplicationpages/, on a backend server with an IP address of 192.0.2.10. In addition to HTML content, the HTML file also references images stored on the server. An HTTP request for the HTML file, sent directly to the server, would be as follows:

```
http://192.0.2.10/myapplicationpages/sitehome.html
```

To send a test request for this file through the NetScaler appliance, in the URL, replace the IP address of the server with the IP address of the public endpoint that is configured for the AppExpert application. For example, if the IP address of the public endpoint is 192.0.2.11, your test URL would be as follows:

```
http://192.0.2.11/myapplicationpages/sitehome.html
```

After you send the request, you must view the hit counter for the application to verify that the public endpoint received the request, view the hit counter for the WebPages application unit to verify that the request for the HTML file matched the rule configured for the application unit, and view the hit counter for the Images application unit to verify that the requests for the images matched the rule configured for the application unit.

For the application, the Hits dialog box displays the total number of requests received by each configured public endpoint. For an application unit, the Hits dialog box displays the number of requests that the application unit processed from each of the public endpoints, and the total hit count.

### **To view the hit counter for an application or application unit**

1. Navigate to AppExpert > Applications.
2. In the details pane, click the application or application unit for which you want to view the hit counter.
3. Click Hits.

## Customizing the Configuration

After you verify that the AppExpert application is working correctly, you can customize the configuration to suit your requirements.

You can configure public endpoints and services for the AppExpert application and specify only a subset of the endpoints and services for each application unit. When you want the AppExpert application to manage a traffic subset that is not included in the template, you can either add an application unit for the new traffic subset or modify an existing application unit rule. You can also specify the order of evaluation of the traffic subsets that the AppExpert application manages.

Finally, you can modify the policies that the template provided. If the AppExpert application template does not include policies for a particular NetScaler feature, such as Rewrite or application firewall, you can configure your own policies.

The order in which you perform these tasks depends on your requirement. However, before you configure a service for an application, you must configure the service for the parent application.



## Configuring Public Endpoints

If you did not specify a public endpoint when importing an AppExpert application, you can specify public endpoints after you create the application. You can configure one public endpoint of type HTTP and one public endpoint of type HTTPS for your AppExpert application.

If endpoints are already configured for the application, you can dissociate endpoints from the AppExpert application and delete any endpoints that you no longer need. Note that when you dissociate a public endpoint from the AppExpert application, the endpoint is automatically unbound from the associated application unit, but it is not deleted from the system.

### To configure public endpoints for an AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application for which you want to configure public endpoints, and then click **Configure Public Endpoints**.
3. In the Choose Public Endpoints dialog box for the application, do one of the following:
  - o If the endpoints you want are listed in the dialog box, click the corresponding check boxes.
  - o If you want to specify all the public endpoints, click **Activate All**.
  - o If you want to dissociate endpoints from the AppExpert application, clear the corresponding check boxes.
  - o If you want to create a new public endpoint, click **Add**. Then, in the Create public endpoint dialog box, configure endpoint settings, and then click **OK**.

In the Create public endpoint dialog box, you can specify only the name, IP address, port, and protocol for the endpoint. You can specify additional endpoint settings after you create the public endpoint. To specify additional endpoint settings, after you create the endpoint, in the Choose Public Endpoints dialog box, click the endpoint, and then click **Open**. Then, in the Configure Public Endpoint dialog box, provide additional settings, and then click **OK**.

For more information about the parameters in the Create public endpoint and Configure Public Endpoint dialog boxes, see "[Content Switching](#)."

- o If you want to modify a public endpoint, click the endpoint, and then click **Open**. Then, in the Configure Public Endpoint dialog box, modify settings for the endpoint, and then click **OK**.

For more information about the parameters in the Configure Public Endpoint dialog box, see "[Content Switching](#)."

4. Click **Close**.

## Configuring Endpoints for an Application Unit

When you configure multiple public endpoints for an AppExpert application, by default, all endpoints are bound to each application unit, and each application unit processes the requests received by all the endpoints. However, you can specify that a given application unit manages the traffic that is received by only a subset of the endpoints that are configured for the AppExpert application.

### To configure endpoints for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application unit for which you want to specify public endpoints, and then click Configure Public Endpoints.
3. In the Choose Public Endpoints dialog box for the application unit, do one of the following:
  - If you are specifying endpoints for the application unit for the first time, clear the check boxes that correspond to the endpoints that you do not want to be bound to the application unit.
  - If you want to specify endpoints that are listed in the dialog box but not currently bound to the application unit, click the corresponding check boxes.
4. Click OK.

## Configuring Services and Service Groups

When you configure a service or service group, you either modify an existing service or service group, or add new services to the AppExpert application. You add services or service groups if you did not specify them when you imported the application template. You also add services and service groups when you increase the number of servers that host instances of the application. You can configure a service and service group for an application unit only after you configure the service or service group for the AppExpert application.

### To configure a service or service group for the AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application for which you want to configure services or service groups, and then click Configure Backend Services.
3. In the Configure Backend Services dialog box, do one of the following:
  - o To configure services, click the Services tab.
  - o To configure service groups, click the Service Groups tab.
4. On the Service or Service Groups tab, do one of the following:
  - o If the services or service groups that you want are listed on the tab, click the corresponding check boxes.
  - o If you want to specify all the services or service groups, click Activate All.
  - o If you want to create a new service or service group, click Add. Then, in the Create Service dialog box or Create Service Group dialog box, configure settings for the service or service group, respectively, and then click Create.
  - o If you want to modify a service, click the service, and then click Open. Then, in the Configure Service dialog box or Create Service Group dialog box, configure settings for the service or service group, respectively, and then click OK.

For information about the settings in the Create Service, Configure Service, and Create Service Group dialog boxes, see "[Load Balancing](#)."

## Configuring Services, Service Groups, and Load Balancing Parameters for an Application Unit

When you configure services and service groups for an AppExpert application, by default, all the services and service groups are bound to each application unit. However, depending on how you have configured your web application, the application resources that are managed by an application unit might be hosted on only some of the servers that are configured as services for the AppExpert application. Or, a set of servers might host content that is meant for the requests received at one or more specific public endpoints. In such scenarios, if all the services and service groups that are configured for the AppExpert application are associated with the application unit, a request that is forwarded to a server that does not host the requested content might not be served or might be served incorrect content. Therefore, you must ensure that each application unit is configured to manage only those services that can serve the requested content.

When configuring services and service groups for an application unit, you might choose to specify load balancing settings such as the weights that services must be assigned and the desired load balancing, persistence, and spillover methods. For more information about these settings, see [Load Balancing](#).

### To configure services or service groups for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application unit for which you want to configure a service or service group, and then click Configure Backend Services.
3. In the Configure Backend Services dialog box, do one of the following:
  - To configure services, click the Services tab.
  - To configure service groups, click the Service Groups tab.
4. In the Services or Service Groups tab, do one of the following:
  - Clear the check boxes that correspond to the services or service groups that you do not want configured for the application unit. Make sure that the check boxes that correspond to the services or service groups that you want configured for the application unit are selected. Then, in the Weight column, specify the weight that you want to assign to each configured service.
  - To specify all services or service groups, click Activate All.
5. On the Method and Persistence and Advanced tabs, specify the desired parameters.
6. Click OK.

## Creating Application Units

You might need to add application units for traffic subsets that are either specific to your web application implementation or not defined in the template. When creating an application unit, you must configure a rule for the application unit.

### To create an application unit for the AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application for which you want to add an application unit, and then click Add.
3. Click Create.

## Configuring Application Unit Rules

You might want to configure an application unit rule to include or exclude certain types of traffic. When you configure the rule, you can also define the syntax of the expression.

### To configure an application unit rule

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Applications.
2. In the details pane, right-click the application unit for which you want to modify the rule, and then click Open.
3. In the Configure Application Unit dialog box, do the following:
  - a. To specify the format of the new expression, do one of the following:
    - To specify that you want to configure a classic expression in the Rule box, click Classic Syntax.
    - To specify that you want to configure an advanced expression in the Rule box, click Default Syntax.
  - b. In the Rule box, configure the expression.
4. Click OK.

## Specifying the Order of Evaluation of Application Units

Application unit rules are evaluated in the order in which they are placed in the graphical user interface (GUI). The rule that is configured for the topmost application unit is always configured first, followed by the rule that is configured for the second topmost application unit, and so on. The default application unit is always evaluated last.

When a request matches the rule that is configured for an application unit, the request is processed by the application unit, and no further matching is performed. Therefore, the order of evaluation of application units becomes an important factor if the traffic subsets for two or more application units overlap. If the traffic subsets for two or more application units overlap, you must specify the order in which an incoming request is matched against the application unit rules.

### To specify the order of evaluation of application units

1. Navigate to AppExpert > Applications.
2. In the details pane, do the following:
  - To move an application unit up by one step, right-click the application unit, and then click Move Up.
  - To move an application unit down by one step, right-click the application unit, and then click Move Down.

## Configuring Policies for Application Units

For an AppExpert application, you can configure policies for Compression, Caching, Rewrite, Responder, and Application Firewall. The templates that you download from the Citrix Community web site provide you with a set of policies that fulfill the most common application management requirements. You might want to fine-tune or customize these policies. If the set of policies provided for a given application unit does not include policies for a particular feature, you can create and bind your own policies for that feature.

If you create an AppExpert application without using a template, you must configure all the policies that the web application needs.

The GUI uses various icons to indicate whether or not policies are configured for a feature. For an application unit, if a policy is configured for a given feature, an icon that represents the feature is displayed. For example, if a compression policy is configured for an application unit, a compression icon is displayed in the Compression column for the application unit. For features for which no policy is configured, an icon depicting a plus sign (+) is displayed.

Note: When configuring policies for application units, you might need to configure policies and expressions that are either in the classic or default syntax. Additionally, when you configure default syntax policies, you might need to specify parameters such as Goto expressions and invoke policy banks. For information about configuring policies and expressions in both formats, see "[Policies and Expressions](#)."

## Configuring Compression Policies

You can use either classic policies or advanced policies to configure compression, but you cannot bind compression policies of both types to the same application unit.

### To configure a compression policy for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, in the row for the application unit you want to configure, click the icon provided in the Compressor column.
3. In the Configure Compression Policies dialog box, do one or more of the following, depending on the configuration tasks you want to perform:
  - Click Switch to Default Syntax if you want to configure a default syntax compression policy. If you want to bind or configure classic compression policies, and if you are in the default syntax view, you can click Switch to Classic Syntax to return to the classic policy view and begin modifying bound classic policies or create and bind new classic compression policies.  
Important: This setting also determines what policies are displayed when you want to insert a policy. For example, if you are in the default syntax view, when you click Insert Policy, the list that appears in the Policy Name column will include only default syntax policies. You cannot bind policies of both types to an application unit.
  - If you want to configure classic policies, click either Request or Response, depending on whether you want the policy to be evaluated at request-time or at response-time.

You can configure both request-time and response-time classic compression policies for an application unit. After evaluating all of the request-time policies, if no match is found, the appliance evaluates response-time policies.

- To modify a compression policy that is already bound to the application unit, click the name of the policy, and then click Modify Policy. Then, in the Configure Compression Policy dialog box, modify the policy, and then click OK.

For information about modifying a compression policy, see "[Compression](#)."

- To unbind a policy, click the name of the policy, and then click Unbind Policy.
- To modify the priority assigned to a policy, double-click the priority value, and then enter a new value.
- To regenerate assigned priorities, click Regenerate Priorities.
- To insert a new policy, click Insert Policy and, in the list that is displayed in the Policy Name column, click New Policy. Then, in the Create Compression Policy dialog box, configure the policy, and then click Create.

For information about modifying a compression policy, see "[Compression](#)."

- If you are configuring a default syntax expression, do the following:
  - In the Goto Expression column, select a Goto expression.
  - In the Invoke column, specify the policy bank that you want to invoke if the current policy evaluates to TRUE.



4. Click Apply Changes, and then click Close.

## Configuring Caching Policies

You can use only default syntax policies and expressions to configure Caching policies.

### To configure Caching policies for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, in the row for the application unit you want to configure, click the icon provided in the Caching column.
3. In the Configure Cache Policies dialog box, do one or more of the following, depending on the configuration tasks you want to perform:
  - o Click either Request or Response, depending on whether you want the policy to be evaluated at request-time or at response-time.

You can configure both request-time and response-time Caching policies for an application unit. After evaluating all of the request-time policies, if no match is found, the appliance evaluates response-time policies.

- o To modify a Caching policy that is already bound to the application unit, click the name of the policy, and then click Modify Policy. Then, in the Configure Cache Policy dialog box, modify the policy, and then click OK.

For information about modifying a Caching policy, see ["Integrated Caching."](#)

- o To unbind a policy, click the name of the policy, and then click Unbind Policy.
- o To modify the priority assigned to a policy, double-click the priority value, and then enter a new value.
- o To regenerate assigned priorities, click Regenerate Priorities.
- o To insert a new policy, click Insert Policy and, in the list that is displayed in the Policy Name column, click New Policy. Then, in the Create Cache Policy dialog box, configure the policy, and then click Create.

For information about modifying a Caching policy, see ["Integrated Caching."](#)

- o In the Goto Expression column, select a Goto expression.
- o In the Invoke column, specify the policy bank that you want to invoke if the current policy evaluates to TRUE.

4. Click Apply Changes, and then click Close.

## Configuring Rewrite Policies

You can use only default syntax policies and expressions to configure Rewrite policies.

### To configure Rewrite policies for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, in the row for the application unit you want to configure, click the icon provided in the Rewrite column.
3. In the Configure Rewrite Policies dialog box, do one or more of the following, depending on the configuration tasks you want to perform:
  - o Click either Request or Response, depending on whether you want the policy to be evaluated at request-time or at response-time.

You can configure both request-time and response-time Rewrite policies for an application unit. After evaluating all of the request-time policies, if no match is found, the appliance evaluates response-time policies.

- o To modify a Rewrite policy that is already bound to the application unit, click the name of the policy, and then click Modify Policy. Then, in the Configure Rewrite Policy dialog box, modify the policy, and then click OK.

For information about modifying a Rewrite policy, see ["Rewrite."](#)

- o To unbind a policy, click the name of the policy, and then click Unbind Policy.
- o To modify the priority assigned to a policy, double-click the priority value, and then enter a new value.
- o To regenerate assigned priorities, click Regenerate Priorities.

- To insert a new policy, click Insert Policy and, in the list that is displayed in the Policy Name column, click New Policy. Then, in the Create Rewrite Policy dialog box, configure the policy, and then click Create.

For information about modifying a Rewrite policy, see "[Rewrite](#)."

- In the Goto Expression column, select a Goto expression.
- In the Invoke column, specify the policy bank that you want to invoke if the current policy evaluates to TRUE.

4. Click Apply Changes, and then click Close.

## Configuring Responder Policies

You can use only default syntax policies and expressions to configure Responder policies.

### To configure Responder policies for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, in the row for the application unit you want to configure, click the icon provided in the Responder column.
3. In the Configure Responder Policies dialog box, do one or more of the following, depending on the configuration tasks you want to perform:
  - To modify a Filter policy that is already bound to the application unit, click the name of the policy, and then click Modify Policy. Then, in the Configure Responder Policy dialog box, modify the policy, and then click OK.

For information about modifying a Responder policy, see "[Responder](#)."

- To unbind a policy, click the name of the policy, and then click Unbind Policy.
- To modify the priority assigned to a policy, double-click the priority value, and then enter a new value.
- To regenerate assigned priorities, click Regenerate Priorities.
- To insert a new policy, click Insert Policy and, in the list that is displayed in the Policy Name column, click New Policy. Then, in the Create Responder Policy dialog box, configure the policy, and then click Create.

For information about modifying a Responder policy, see "[Responder](#)."

- In the Goto Expression column, select a Goto expression.
- In the Invoke column, specify the policy bank that you want to invoke if the current policy evaluates to TRUE.

4. Click Apply Changes, and then click Close.

## Configuring Application Firewall Policies

You can configure both classic and default syntax policies and expressions for Application Firewall. However, if a policy of one type is already bound globally or to a virtual server that is configured on the appliance, you cannot bind a policy of the other type to an application unit. For example, if a default syntax policy is already bound either globally or to a virtual server, you cannot bind a classic policy to an application unit.

### To configure Application Firewall policies for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, in the row for the application unit you want to configure, click the icon provided in the Application Firewall column.
3. In the Configure Application Firewall Policies dialog box, do one or more of the following, depending on the configuration tasks you want to perform:
  - Click either Classic Expression or Advanced Expression depending on the type of expression you want to configure for the Application Firewall policy.  
Important: This setting also determines what policies are displayed when you want to insert a policy. For example, if you select Advanced Expression, when you click Insert Policy, the list that appears in the Policy Name column will include only default syntax policies. You cannot bind policies of both types to an application unit. This option is not available if a policy of either type is already bound either globally or to a virtual server.
  - To modify an application firewall policy that is already bound to the application unit, click the name of the policy, and then click Modify Policy. Then, in the Configure Application Firewall Policy dialog box, modify the policy, and then click OK.

For information about modifying a application firewall policy, see "[Policies](#)."

- To unbind a policy, click the name of the policy, and then click Unbind Policy.
- To modify the priority assigned to a policy, double-click the priority value, and then enter a new value.
- To regenerate assigned priorities, click Regenerate Priorities.
- To insert a new policy, click Insert Policy and, in the list that is displayed in the Policy Name column, click New Policy. Then, in the Create Application Firewall Policy dialog box, configure the policy, and then click Create.

For information about modifying a application firewall policy, see "[Policies](#)."

4. Click Apply Changes, and then click Close.

## Configuring Persistency Groups for Application Units

You can configure a persistency group for the application units in an AppExpert application. In the context of an AppExpert application, a persistency group is a group of application units that you can treat as a single entity for the purpose of applying common persistence settings. When the application is exported to an application template file, the persistency group settings are included, and they are automatically applied to the application units when you import the AppExpert application.

### To configure a persistency group for an application by using the configuration utility

1. Navigate to AppExpert > Applications.
2. In the Applications View dialog box, click the name of the application for whose application units you want to configure a persistency group, and then click Configure Persistency Groups.
3. In the Configure Persistency Groups dialog box, do one of the following:
  - o To add a persistency group, click Add.
  - o To modify a persistency group, click Open.
4. In the Create Persistency Group or Configure Persistency Group dialog box, set the following parameters:
  - o Group Name – Name of the persistency group. For the NetScaler appliance to recognize the persistency group as part of the application's configuration, the name of the AppExpert application must be included in the name of the persistency group, as a prefix. Therefore, by default, the appliance displays the prefix in the Group Name box, and you cannot remove that prefix. Enter a name of your choice after the prefix.
  - o Persistence – Type of persistence for the virtual server. If you select SOURCEIP, in the IPv4 Netmask box, enter a network mask that specifies the number of bits that the appliance must consider when creating persistence sessions. If you select COOKIEINSERT, in the Cookie Domain and Cookie Name boxes, specify a domain attribute to send in the Set-Cookie directive, and a name for the cookie, respectively.
  - o Timeout – Time period for which a persistence session is in effect.
  - o Backup Persistence – Type of backup persistence for the group.
  - o Backup Timeout – Time period, in minutes, for which backup persistence is in effect.
  - o Application Units – To add an application unit to the persistency group, in the Available Application Units box, click the application unit, and then click Add. To remove an application unit from the persistency group, in the Configured Application Units box, click the application unit, and then click Remove.
5. Click OK.

## Viewing AppExpert Applications and Configuring Entities by Using the Application Visualizer

The Application Visualizer is a graphical representation of an AppExpert application. The Visualizer displays the public endpoints, application units, backend services, and policies that are configured for the application. You can use the Visualizer to obtain a visual overview of an AppExpert application's configuration and configure some of the displayed entities. By default, the Visualizer displays application units, services, and monitors for the selected application.

### To view an AppExpert application by using the Application Visualizer

1. Navigate to AppExpert > Applications.
2. In the details pane, click the name of the application that you want to view, and then click Visualizer.
3. Do one or more of the following:
  - To optimize the display area, choose Best Fit, Zoom In, or Zoom out. If an item that you want to see disappears from view after zooming in, you can click and drag the viewable area.
  - To save the graph as an image file, click Save Image.
  - To find a particular entity, in the Search in field, type an entity name. In the view area, the entity names that begin with the search string are highlighted. To restrict the search, click the drop-down menu and select the specific entity that you want to search for.
  - To view the policies that an application uses, click one or more icons to display feature-specific policies. The policy types are Compression, Filter, Rewrite, Responder, Cache, application firewall, Authorization, Auditing, HTML Injection, SureConnect, Priority Queuing, and Traffic.
  - To view the rule that is configured for an application unit, click the curve that connects the public endpoint to the application unit. The rule is displayed on the Related Tasks tab.
  - To view the binding information for an application unit, policy, or monitor, click the displayed icon, click the Related Tasks tab, and then click Show Bindings
  - To view member services, click the icon for the service, click the Related Tasks tab, and then click Show Member Services.
  - To view detailed statistics for a public endpoint or application unit, click the icon that is displayed, click the Related Tasks tab, and then click Statistics.
  - To view the Load Balancing Visualizer for an application unit, click the application unit, click Related Tasks, and then click Visualizer.
  - To view the number of requests received per second at a given point in time by the load balancing virtual server and the number of hits per second at a given point in time for rewrite, responder, and cache policies, click Show Stats. The statistical information is displayed on the respective nodes in the Visualizer. This information is not updated in real time and has to be refreshed manually. To refresh this information, click Refresh Stats.

### To configure and view entities in an AppExpert application by using the Application Visualizer

1. Navigate to AppExpert > Applications.
2. In the details pane, click the name of the application that you want to configure or view, and then click Visualizer.
3. Do one or more of the following:
  - To configure an entity that is displayed in the viewing area, click the icon for the entity, click the Related Tasks tab, and then click Modify public endpoint.

In the Application Visualizer, you can modify only public endpoints and services.

- To bind additional monitors to a service, click the Available Resources tab, select Monitors from the drop-down list, and then click and drag a monitor to a service.
- To unbind a service from an application unit, click the curve that connects the application unit and the service, click Related Tasks, and then click Unbind.
- To unbind a monitor from a service, click the curve that connects the service and the monitor, click Related Tasks, and then click Unbind.
- To modify a monitor, click the monitor, click Related Tasks, and then click Open.
- To modify the binding parameters for a monitor, click the curve that connects the monitor to the associated service, click Related Tasks, and then click Modify Parameters.
- To apply a common service configuration across multiple service containers that are displayed when the services bound to a vservice do not have the same configuration, click the service container whose

configuration you want to apply to all the containers, and then, in Related Tasks, click Apply Configuration.

- o To view a comparative list of the parameters whose values differ across service containers, click the icon for a container, click the Related Tasks tab, and then click Service Attributes Diff. The comparative list helps you determine which service container has the service configuration that you want to apply to all the containers. After you determine which service container has the configuration you want, right-click the container, and then click Apply this Configuration.
- o To copy the configuration of an entity (other than the configuration of the AppExpert application) to the local computer's clipboard, click the entity, click the Related Tasks tab, and then click Copy Properties. You can then paste the configuration information in a word processing document or spreadsheet.

## Monitoring a NetScaler Application

After you customize the AppExpert application, you can view application statistics to make sure that the application and all its entities are working correctly. You can also use the Application Visualizer to monitor statistics associated with certain entities such as policies and virtual servers.

You can also view the hit counters for various entities at regular intervals to make sure that counters are being updated.

### Viewing Application Statistics

Updated: 2013-08-30

In the Applications node, you can select an application and view the Statistics page for the application. On the Statistics page, you can monitor the health and states of public endpoints and application units, and view the following statistical information:

- Requests and responses per second for each of the public endpoints and application units.
- Bytes per second, at each endpoint, for incoming and outgoing traffic.
- Application unit hit counters and the number of client and server connections for each application unit.
- Statistics for the services that are bound to the application units.

On the Statistics page, you can also view CPU usage, memory usage, and system logs.

#### To view statistics for an application

1. Navigate to AppExpert > Applications.
2. In the details pane, click the application for which you want to view statistics, and then click Statistics.

## Monitoring an Application by Using the Application Visualizer

Updated: 2013-08-30

You can use the Application Visualizer to monitor the number of requests received per second at a given point in time by the vservers and the number of hits per second at a given point in time for Rewrite, Responder, and Cache policies.

#### To view statistical information for vservers, Rewrite policies, Responder policies, and Cache policies in the Visualizer

1. Navigate to AppExpert > Applications.
2. In the details pane, select the application for which you want to view statistical information, and then click Visualizer.
3. In the Application Visualizer window, do the following:

- To view the statistics, click Show Stats.

The statistical information is displayed on the respective nodes in the Visualizer. This information is not updated in real time and has to be refreshed manually.

- To refresh the statistical information, click Refresh Stats.

### Viewing Hits

Updated: 2013-08-30

The hit counters that are provided for various AppExpert application entities enable you to monitor the functioning of public endpoints and application units. For an application, the Hits dialog box displays the total number of requests received by each configured public endpoint. For an application unit, the Hits dialog box displays the number of requests that the application unit processed from each of the public endpoints and the total hit count. For instructions on viewing hit counters, see ["Verifying and Testing the Configuration."](#)

## Deleting an Application

If you no longer need an application and its application units, you can delete it. When you delete an AppExpert application, backend services are not deleted, and any public endpoints that the application used become available for use by other applications.

When deleting an application, you are also prompted to specify whether you want to delete any bound policies and actions that are not used elsewhere.

### To delete an application

1. Navigate to AppExpert > Applications.
2. In the details pane, click the name of the application that you want to delete, and then click Remove.



## Configuring Authentication, Authorization, and Auditing

You can configure Authentication, Authorization, and Auditing (AAA) for the applications that you configure on the appliance. An authentication policy that is configured for an application defines the type of authentication to apply when a user or group attempts to access the application. If external authentication is used, the policy also specifies the external authentication server. Authorization policies configured for an application specify whether a particular user or group can access the application. Auditing policies define the audit log type, the level at which logging is performed, and other audit server settings. Authentication and auditing policies use the classic policy format.

Authentication policies, authorization policies, and auditing policies can be configured in any order. However, before you configure AAA for an application, you must configure a public endpoint for the application.

This document includes the following details:

- [Configuring Authentication](#)
- [Configuring Authorization](#)
- [Configuring Auditing](#)
- [Disabling AAA for an Application](#)

## Configuring Authentication

Updated: 2013-08-30

Configuring authentication for an application involves specifying an authentication FQDN, an authentication virtual server, a server certificate, and authentication and session policies. Authentication policies are automatically bound to the authentication virtual server specified for the application.

### To configure authentication for an AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, click the name of the application for which you want to configure authentication, and then click Authentication.
3. In the Authentication Wizard, on the Introduction page, click Next.
4. Follow the instructions in the Authentication Wizard.

## Configuring Authorization

Updated: 2013-08-30

You can configure authorization for users and groups to enable them to access an AppExpert application. If the AAA user or group for which you want to configure permissions has not already been created, you can create it from AppExpert and then configure permissions for application access.

### To configure permissions for a AAA user or group to access an AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, click the AppExpert application for which you want to configure user or group access, and then click Authorization.
3. Do one of the following:
  - If the AAA user or group for which you want to configure permissions is already in the Groups/Users tree, drag the user or group from the Groups/Users tree to the Users or Groups node in the application tree. Then, right-click the user or group and click Allow.
  - If the AAA user or group for which you want to configure permissions is not configured on the appliance, in the application tree, right-click Users or Groups, and then click Add. In the Create AAA Group or Create AAA User dialog box, fill in the values, click Create, and then click Close.

The user or group is created with the permission set to Allow. To change the permission setting, right-click the group or user, and then click the permission setting.

4. Click Close.

## Configuring Auditing

Updated: 2013-08-30

When you configure auditing policies for an application, you must specify the server to which the log messages must be directed, the format of the messages logged, and the log level. Optionally, you can configure other settings, such as the log facility and date format. Auditing policies are automatically bound to all the AppExpert application's public endpoints.

## To configure auditing policies for an application

1. Navigate to AppExpert > Applications.
2. In the details pane, click the application for which you want to configure auditing policies, and then click Auditing.
3. In the Configure Auditing Policies dialog box, click Insert Policy.
  - o To specify an existing auditing policy, under Policy Name, click the name of the policy, and then do the following:
    - To modify the priority that is assigned to the policy by default, under Priority, double-click the priority, and then type a new priority value.
    - To modify the settings of the audit server, under Server, double-click the name of the server, and then, in the Configure Auditing Server dialog box, modify the settings as appropriate. You can modify all the settings in this dialog box except the name of the audit server and the audit type. For more information about the settings in the Configure Auditing Server dialog box, see ["Auditing Policies."](#)
  - o To create a new auditing policy, under Policy Name, click New Policy, and then, in the Create Auditing Policy dialog box, do the following:
    - In the Name box, type a name for the policy.
    - The Name box already contains the string that is required at the beginning of the server name. You cannot modify the string.
    - From the Auditing Type list, select the auditing type (either SYSLOG or NSLOG).
    - If the audit server you want to specify is already listed in the Server list, select the server from the list, and then, if you want to modify the server settings, click Modify. In the Configure Auditing Server dialog box, modify the settings as appropriate, and then click OK. For more information about the settings in the Configure Auditing Server dialog box, see ["Auditing Policies."](#)
    - If you want to configure a new audit server, click New, and then, in the Create Auditing Server dialog box, type a name for the server, specify the server IP address, port number, and other settings as appropriate. When finished, click OK.
    - Click Create.
  - o To change the priorities for the new auditing policies you created, under Priority, for each policy for which you want to change the priority, double-click the priority value and type new priority value.
  - o To regenerate priorities, click Regenerate Priorities.
  - o To unbind a policy, click the policy, and then click Unbind Policy.
  - o To modify a policy, click the policy, and then click Modify Policy.
4. Click Apply Changes, and then click Close.

## Disabling AAA for an Application

Updated: 2013-08-30

After you configure AAA for an application, you can disable the AAA configuration for that application. When you disable AAA for an application, the configuration is not lost. You can enable AAA for the application when you want to reapply the configuration.

### To enable or disable AAA for an application

1. Navigate to AppExpert > Applications.
2. In the details pane, click the application for which you want to enable or disable AAA, and then do one of the following
  - o To disable AAA for the application, click Turn Off AAA.
  - o To enable AAA for the application, click Turn On AAA.

# Setting Up a Custom NetScaler Application

If an AppExpert application template is not available for the Web application that you want to manage through the NetScaler appliance, or if available AppExpert application templates do not suit your requirements, you can create an AppExpert application without a template.

To create an AppExpert application without a template, you must first create an application and application units. Then, you configure public endpoints, services, and service groups. Finally, you configure the policies that determine how application traffic is evaluated and processed.

After you create the application and application units and configure policies, you must verify the configuration and test it to make sure that it is working correctly, just as you would when you configure an application by using a prebuilt AppExpert application template. Then, you must monitor the application to make sure that the application and its entities are working correctly.

This document includes the following details:

- [Creating an Application](#)
- [Creating Application Units](#)
- [Configuring Public Endpoints for an AppExpert Application](#)
- [Configuring Public Endpoints for an Application Unit](#)
- [Configuring Services and Service Groups for an AppExpert Application](#)
- [Configuring Services and Service Groups for an Application Unit](#)
- [Configuring Policies](#)

## Creating an Application

Updated: 2013-08-30

When you create an AppExpert application, the appliance creates a container to which you can add application units. The *default* application unit is not created until you create the first application unit.

### To create an AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click Applications, and then click Add.
3. In the Create Application dialog box, in Name, enter a name for the application, and then click OK.

## Creating Application Units

Updated: 2013-08-30

For each subset of traffic associated with your web application, you must create an application unit.

### To create an application unit for the AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application for which you want to add an application unit, and then click Add.
3. Click Create.

## Configuring Public Endpoints for an AppExpert Application

Updated: 2013-08-30

After you have created all the application units that you require, you must configure one or more public endpoints to enable clients to access the web application through the NetScaler appliance.

### To configure public endpoints for an AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application for which you want to configure public endpoints, and then click Configure Public Endpoints.
3. In the Choose Public Endpoints dialog box for the application, do one of the following:
  - If the endpoints you want are listed in the dialog box, click the corresponding check boxes.
  - If you want to specify all the public endpoints, click Activate All.
  - If you want to dissociate endpoints from the AppExpert application, clear the corresponding check boxes.

- If you want to create a new public endpoint, click Add. Then, in the Create public endpoint dialog box, configure endpoint settings, and then click OK.

In the Create public endpoint dialog box, you can specify only the name, IP address, port, and protocol for the endpoint. You can specify additional endpoint settings after you create the public endpoint. To specify additional endpoint settings, after you create the endpoint, in the Choose Public Endpoints dialog box, click the endpoint, and then click Open. Then, in the Configure Public Endpoint dialog box, provide additional settings, and then click OK.

For more information about the parameters in the Create public endpoint and Configure Public Endpoint dialog boxes, see "[Content Switching](#)."

- If you want to modify a public endpoint, click the endpoint, and then click Open. Then, in the Configure Public Endpoint dialog box, modify settings for the endpoint, and then click OK.

For more information about the parameters in the Configure Public Endpoint dialog box, see "[Content Switching](#)."

4. Click Close.

## Configuring Public Endpoints for an Application Unit

Updated: 2013-08-30

For an application unit, you specify public endpoints in the same way as you would specify public endpoints for an application that is created from an AppExpert application template. For more information about specifying a subset of the endpoints for an application unit, see "[Configuring Endpoints for an Application Unit](#)."

### To configure endpoints for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application unit for which you want to specify public endpoints, and then click Configure Public Endpoints.
3. In the Choose Public Endpoints dialog box for the application unit, do one of the following:
  - If you are specifying endpoints for the application unit for the first time, clear the check boxes that correspond to the endpoints that you do not want to be bound to the application unit.
  - If you want to specify endpoints that are listed in the dialog box but not currently bound to the application unit, click the corresponding check boxes.
4. Click OK.

## Configuring Services and Service Groups for an AppExpert Application

Updated: 2013-08-30

Services and service groups are available for application units only after you configure the services and service groups for the AppExpert application. Therefore, you must configure services and service groups for the AppExpert application before you configure the services for the application units. All the services and service groups that you configure for an AppExpert application must use the same protocol (either HTTP or HTTPS). The procedure for configuring services and service groups for an AppExpert application that is not created from a template is the same as that for an application created from a template.

### To configure a service or service group for the AppExpert application

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application for which you want to configure services or service groups, and then click Configure Backend Services.
3. In the Configure Backend Services dialog box, do one of the following:
  - To configure services, click the Services tab.
  - To configure service groups, click the Service Groups tab.
4. On the Service or Service Groups tab, do one of the following:
  - If the services or service groups that you want are listed on the tab, click the corresponding check boxes.
  - If you want to specify all the services or service groups, click Activate All.
  - If you want to create a new service or service group, click Add. Then, in the Create Service dialog box or Create Service Group dialog box, configure settings for the service or service group, respectively, and then click Create.

- If you want to modify a service, click the service, and then click Open. Then, in the Configure Service dialog box or Create Service Group dialog box, configure settings for the service or service group, respectively, and then click OK.

For information about the settings in the Create Service, Configure Service, and Create Service Group dialog boxes, see ["Load Balancing."](#)

## Configuring Services and Service Groups for an Application Unit

Updated: 2013-08-30

After you configure services and service groups, you must configure services and service groups for each application unit. However, this step is not necessary if each backend service hosts all the content associated with the web application. You configure services and service groups for an application unit if the content associated with the application unit is hosted on only a subset of the backend servers.

### To configure services or service groups for an application unit

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application unit for which you want to configure a service or service group, and then click Configure Backend Services.
3. In the Configure Backend Services dialog box, do one of the following:
  - To configure services, click the Services tab.
  - To configure service groups, click the Service Groups tab.
4. In the Services or Service Groups tab, do one of the following:
  - Clear the check boxes that correspond to the services or service groups that you do not want configured for the application unit. Make sure that the check boxes that correspond to the services or service groups that you want configured for the application unit are selected. Then, in the Weight column, specify the weight that you want to assign to each configured service.
  - To specify all services or service groups, click Activate All.
5. On the Method and Persistence and Advanced tabs, specify the desired parameters.
6. Click OK.

## Configuring Policies

Updated: 2013-08-30

The procedures for configuring policies for an AppExpert application that is created without using a template are the same as those for an AppExpert application that was created from a template. For more information, see ["Configuring Policies for Application Units."](#)

## Creating and Managing Template Files

After you set up an AppExpert application and customize it to suit your requirements, you can create a template from the application and then share the template with other administrators. Or, you can create a template and then import the template to other NetScaler appliances that require a similar AppExpert application configuration. This simplifies and expedites the process of setting up similar applications on other appliances. You can also export a content switching configuration to a template file. When creating a template file, you can configure variables in the policy expressions and actions that are configured for an application.

AppExpert application template files can be exported either to the template directory on the NetScaler appliance or to a folder on your local computer. You can then upload and download the templates to and from the NetScaler appliance and rename the templates that are stored in the AppExpert application templates directory on your appliance.

This document includes the following information:

- [Exporting an AppExpert Application to a Template File](#)
- [Exporting a Content Switching Virtual Server Configuration to a Template File](#)
- [Creating Variables in Application Templates](#)
- [Uploading and Downloading Template Files](#)
- [Understanding NetScaler Application Templates and Deployment Files](#)

## Exporting an AppExpert Application to a Template File

When you export an AppExpert application, all application-configuration information is exported to a template file and all deployment-specific information is exported to a deployment file. The string `_deployment` is automatically appended to the name of the template file to create the name of the deployment file. Both files are in XML format. If you choose to export the application template file to the NetScaler appliance, the template file is stored in the `/nsconfig/nstemplates/applications` directory on the NetScaler appliance and the deployment file is stored in the `/nsconfig/nstemplates/applications/deployment_files/` directory. For more information about the format of application templates and deployment files, see ["Understanding NetScaler Application Templates and Deployment Files"](#). If you have configured NetScaler Gateway application, when you export the AppExpert configuration, you can choose to include the NetScaler Gateway policies in the template.

### To export an AppExpert application to a template file

1. Navigate to AppExpert > Applications.
2. In the details pane, click the name of the application that you want to export as a template file, and then click Export.
3. In the Export...as Template dialog box, do the following:
  - a. In the Name box, modify the name of the template, if necessary.
  - b. If you want to configure variables for the template, click Configure Variables, and then, in the Configure Variables dialog box, configure the variables that you want.

For more information about configuring variables in application templates, see ["Creating Variables in Application Templates"](#).

- c. If you want to export the template file to the application templates directory on the appliance, make sure that Browse (Appliance) is displayed.
- d. If you want to export the template file to your computer, click the Browse (Appliance) drop-down menu, click Local, browse to the location to which you want to save the file, and then click Save.
- e. Provide the following information:
  - o **Introduction Description**—Any text that introduces the AppExpert application template during import. This text is displayed on the Specify Application Name page of the AppExpert Template Wizard when the template is imported.
  - o **Summary Description**—Any summary that you might want to display on the Summary page of the AppExpert Template Wizard when the template is imported.
  - o **Author**—The name of the author of the template.
  - o **Major**—The major version number of the template.
  - o **Minor**—The minor version number of the template. This number is appended to the major version number and displayed on the Summary page of the AppExpert Template Wizard, during import, in the format Major.Minor.
- f. Click OK.

If NetScaler Gateway policies have been configured for the application, you will be prompted to include the NetScaler Gateway configuration in the application template. If you want to include the NetScaler Gateway configuration in the template, at the prompt, click Yes.



## Exporting a Content Switching Virtual Server Configuration to a Template File

You can also export a content switching configuration as an application template. You can export a content switching virtual server configuration to an application template either from the Content Switching Virtual Servers pane or from the Content Switching Visualizer. Configuration information, which includes the content switching virtual server, all associated load balancing virtual servers, services, service groups, and policies, is exported to a template file and all deployment-specific information is exported to a deployment file. The string "\_deployment" is automatically appended to the name of the template file to create the name of the deployment file. Both files are in XML format. If you choose to export the application template file to the NetScaler appliance, the template file is stored in the /nsconfig/nstemplates/applications directory on the NetScaler appliance and the deployment file is stored in the /nsconfig/nstemplates/applications/deployment\_files/ directory. For more information about the format of application templates and deployment files, see ["Understanding NetScaler Application Templates and Deployment Files."](#) The configuration information that is exported includes the content switching virtual server, all associated load balancing virtual servers, services, service groups, and policies.

However, if the content switching virtual server is already configured as the public endpoint for an AppExpert application, you cannot export the configuration to a template file. In this scenario, you must export the associated AppExpert application to a template. For more information about exporting an AppExpert application to a template file, see ["Exporting an AppExpert Application to a Template File."](#)

### To export a content switching configuration to an application template file from the Content Switching Visualizer

1. Navigate to Traffic Management > Content Switching > Virtual Servers.
2. In the details pane, click the name of the content switching virtual server whose configuration you want to export as a template file, and then click Visualizer.
3. In the Content Switching Visualizer, click the icon for the content switching vserver, click Related Tasks, and then click Create Template.
4. In the Export...as Template dialog box, enter a name for the template file, and then do one of the following:
  - o To export the template file to the appliance, make sure that Browse (Appliance) is displayed.
  - o To export the template file to your computer, click the Browse (Appliance) drop-down menu, click Local, browse to the location to which you want to save the file, and then click Save.
5. Provide the following information:
  - o **Introduction Description**—Any text that introduces the AppExpert application template during import. This text is displayed on the Specify Application Name page of the AppExpert Template Wizard when the template is imported.
  - o **Summary Description**—Any summary that you might want to display on the Summary page of the AppExpert Template Wizard when the template is imported.
  - o **Author**—The name of the author of the template.
  - o **Major**—The major version number of the template.
  - o **Minor**—The minor version number of the template. This number is appended to the major version number and displayed on the Summary page of the AppExpert Template Wizard, during import, in the format Major.Minor.
6. Click OK.

### To export a content switching configuration to an application template file from the Content Switching Virtual Servers pane

1. Navigate to Traffic Management > Content Switching > Virtual Servers.
2. In the details pane, click the name of the content switching virtual server whose configuration you want to export as a template file, and then click Create AppExpert Template.
3. Perform steps 4 through 6 described in ["To export a content switching configuration to an application template file from the Content Switching Visualizer"](#).



## Creating Variables in Application Templates

Application templates support the declaration of variables in the policy expressions and actions that are configured for an application. The ability to declare variables in policy expressions and actions enables you to replace preconfigured values in expressions (for example, configurable parameters such as the host name of a server or the target for a Rewrite action) with values that suit the environment into which you are importing the template. If variables have been configured for an AppExpert application template, the AppExpert Template Wizard, which appears when you import an AppExpert application template, includes a Specify Variable Values page on which you can specify appropriate values for the variables that are configured for the template.


As an example, consider the following policy expression that is configured to evaluate the value of the Host header in an HTTP request:

```
HTTP.REQUEST.HEADER( "Host" ).CONTAINS( "server1" )
```

If you want the server name to be configurable at import time, you can specify the string "server1" as a variable. When importing the template, you can specify a new value for the variable on the Variables tab.


After you create a variable, you can do the following:

- Assign additional strings to an existing variable. After you create a variable for a string, you can select and assign other parts of the same or different expression to the variable. The strings you assign to a variable need not be the same. At import time, all the strings that are assigned to the variable are replaced with the value that you provide.
- View the string or strings that are assigned to the variable.
- View a list of all the entities and parameters that use the variable.

In the export application template wizard, you can define variables in certain fields (fields with an adjacent  button) for the following entities:

- Cache policies
- Rewrite policies
- Rewrite actions
- Responder policies
- Responder actions

## To configure a variable in a policy expression or action

1. Navigate to AppExpert > Applications.
2. In the details pane, right-click the application that you want to export to a template file, and then click Export.
3. In the Export...as Template dialog box, modify the default template file name if required, specify the location where you want to save the template, and then click Configure Variables.
4. In the Configure Variables dialog box, click the tab that lists the policy expression or action for which you want to configure a variable, select the expression, and then click Configure Variables.
5. In the Variables dialog box, click the  button next to the expression or value in which you want to create a variable.
6. In the Variables dialog box, do the following:
  - To create a variable, in the text box that displays the configured expression or value, select the string that you want to be configurable at import time, and then click Add. In the Add Variable dialog box, specify a name and a description for the variable, and then click Create.
  - The name of the variable, its value, and the description you provided appear in the Available Variables listing in the dialog box. The name you provide will be the name of the associated field in the template import wizard, and the description will appear as alt text when the user positions the mouse pointer over the field.
  - To modify a variable, in the Available Variables list, click the variable, and then click Open. In the Add Variable dialog box, modify the value and the description, and then click OK.
  - To view all the strings that are assigned to a given variable, in the Available Variables listing, click the name of the variable. The strings that are assigned to the variable are highlighted.
  - To view a list of all the entities and parameters in which the variable is used, in the Available Variables listing, click the variable whose references you want to view, and then click Show References.

- To assign a string to an existing variable, in the text box that displays the expression you configured, select the string you want to assign to an existing variable, right-click the selection, click Use Existing Selection, and then click the name of the variable to which you want to assign the string.

If a variable has multiple strings assigned to it, when you specify a new value for the variable during import, all strings assigned to the variable are replaced with the new value.

7. Click Close.

## Uploading and Downloading Template Files

Template files can be uploaded from your local computer to the NetScaler appliance or downloaded from the appliance to your local computer. On the appliance, AppExpert application templates are always stored in the AppExpert application templates directory, which is `/nsconfig/nstemplates/applications/`.

### To upload an AppExpert application template from your local computer to the NetScaler appliance

1. Navigate to AppExpert > Templates.
2. In the details pane, click Manage Templates.
3. In the Manage Application Templates dialog box, click Application Templates, and then click Upload.
4. In the Upload Application Template dialog box, browse to the directory in which the template file is stored, click the template file, and then click Select.

The template file is uploaded to the AppExpert application template directory on the appliance.

### To download an AppExpert application template from the NetScaler appliance to your local computer

1. Navigate to AppExpert > Templates.
2. In the details pane, click Manage Templates.
3. In the Manage Application Templates dialog box, click the AppExpert application template that you want to download, and click Download.
4. In the Download Application Template dialog box, browse to the location to which you want to save the file, and then click Save.

# Understanding NetScaler Application Templates and Deployment Files

When you export a NetScaler application, the following two files are automatically created:

- **NetScaler application template file.** Contains application-configuration information such as application units, rules, and configured policies.
- **Deployment file.** Contains deployment-specific information such as public endpoints, services, associated IP addresses, and configured variables.

In the application template and deployment file, each unit of application-configuration information is encapsulated in a specific XML element that is meant for that unit type. For example, each public endpoint and associated endpoint details are encapsulated within the `<appendpoint>` and `</appendpoint>` tags, and all the endpoint elements are encapsulated within the `<appendpoint_list>` and `</appendpoint_list>` tags.

Note: After you export a NetScaler application, you can add elements, remove elements, and modify existing elements before importing the application to a NetScaler appliance.

## Example of a NetScaler Application Template

Following is an example of a template file that was created from a NetScaler application called "SharePoint\_Team\_Site":

```
<?xml version="1.0" encoding="UTF-8" ?>
<template>
<template_info>
  <application_name>SharePoint_Team_Site</application_name>
  <templateversion_major>1</templateversion_major>
  <templateversion_minor>1</templateversion_minor>
  <author>Ed</author>
  <introduction>An application for managing a SharePoint team site with images, reports, an
  <summary>This template includes variables</summary>
  <version_major>9</version_major>
  <version_minor>3</version_minor>
  <build_number>38</build_number>
</template_info>
<apptemplate>
  <rewrite>
    <rewriteaction_list>
      <rewriteaction>
        <name>Rw_name</name>
        <type>replace</type>
        <target>HTTP.REQ.BODY(10000).AFTER_REGEX(re/number/).BEFORE_REGEX(re/address/)</
        <stringbuilderexpr>"NA"</stringbuilderexpr>
        <allow_unsafe_pil>NO</allow_unsafe_pil>
      </rewriteaction>
      <rewriteaction>
        .
        .
        .
      </rewriteaction>
      .
      .
      .
    </rewriteaction_list>
    <rewritepolicy_list>
      <rewritepolicy>
        <name>Rw_number_NA</name>
        <rule>HTTP.REQ.BODY(10000).CONTAINS("admin")</rule>
        <action>Rw_name</action>
      </rewritepolicy>
      <rewritepolicy>
        .
        .
        .
      </rewritepolicy>
      .
      .
      .
    </rewritepolicy_list>
  </rewrite>
</appunit_list>
```

```

<appunit>
  <name>SharePoint_Team_Sitedefault</name>
  <rule />
  <expressiontype>PE</expressiontype>
  <servicetype>HTTP</servicetype>
  <ipv46>0.0.0.0</ipv46>
  <ipmask>*</ipmask>
  <port>0</port>
  <range>1</range>
  <persistencetype>NONE</persistencetype>
  <timeout>2</timeout>
  <persistencebackup>NONE</persistencebackup>
  <backupperpersistencetimeout>2</backupperpersistencetimeout>
  <lbmethod>LEASTCONNECTION</lbmethod>
  <persistmask>255.255.255.255</persistmask>
  <v6persistmasklen>128</v6persistmasklen>
  <pq>OFF</pq>
  <sc>OFF</sc>
  <m>IP</m>
  <datalength>0</datalength>
  <dataoffset>0</dataoffset>
  <sessionless>DISABLED</sessionless>
  <state>ENABLED</state>
  <connfailover>DISABLED</connfailover>
  <clttimeout>180</clttimeout>
  <somethod>NONE</somethod>
  <sopersistence>DISABLED</sopersistence>
  <redirectportrewrite>DISABLED</redirectportrewrite>
  <downstateflush>DISABLED</downstateflush>
  <gt2gb>DISABLED</gt2gb>
  <ipmapping>0.0.0.0</ipmapping>
  <disableprimaryondown>DISABLED</disableprimaryondown>
  <insertvserveripport>OFF</insertvserveripport>
  <authentication>OFF</authentication>
  <authn401>OFF</authn401>
  <push>DISABLED</push>
  <pushlabel>none</pushlabel>
  <l2conn>OFF</l2conn>
</appunit>
<appunit>
  .
  .
  .
</appunit>
.
.
.
</appunit_list>
</apptemplate>
<parameters>
  <property_list>
    <property>
      <variable_definition_list>
        <variable_definition>
          <name>body_size</name>
          <defaultvalue>10000</defaultvalue>
          <description>Evaluation Scope</description>
          <startindex>14</startindex>
          <length>5</length>
        </variable_definition>
        .
        .
        .
      </variable_definition_list>
      <object_type>rewriteaction</object_type>
      <object_name>Rw_name</object_name>
      <name>target</name>
    </property>
    .
    .
    .
  </property_list>
</parameters>
</template>

```

## Example of a Deployment File

Following is the deployment file associated with the "SharePoint\_Team\_Site" application in the preceding example:

```
<?xml version="1.0" encoding="UTF8" ?>
<template_deployment>
  <template_info>
    <application_name>SharePoint_Team_Site</application_name>
    <templateversion_major>1</templateversion_major>
    <templateversion_minor>1</templateversion_minor>
    <author>Ed</author>
    <introduction>An application for managing a SharePoint team site with images, reports,
    <summary>This template includes variables</summary>
    <version_major>9</version_major>
    <version_minor>3</version_minor>
    <build_number>38</build_number>
  </template_info>
  <appendpoint_list>
    <appendpoint>
      <ipv46>10.111.111.1</ipv46>
      <port>80</port>
      <servicetype>HTTP</servicetype>
    </appendpoint>
  </appendpoint_list>
  <service_list>
    <service>
      <ip>10.102.29.5</ip>
      <port>80</port>
      <servicetype>HTTP</servicetype>
    </service>
    <service>
      .
      .
      .
    </service>
    .
    .
    .
  </service_list>
  <variable_list>
    <variable>
      <name>body_size</name>
      <description>Evaluation Scope</description>
      <value>10000</value>
    </variable>
    <variable>
      .
      .
      .
    </variable>
    .
    .
    .
  </variable_list>
</template_deployment>
```

## NetScaler Gateway Applications

When you configure an AppExpert application to manage a web application through the Citrix® NetScaler® appliance, you also create a set of application units and configure a set of traffic optimization and security policies for each unit. The policies that you configure for each application unit (policies for features such as Compression, Caching, and Rewrite) evaluate traffic that is meant only for that unit. In addition to these policies, you might want to configure Access Gateway policies for the application as a whole to optimize the application traffic when accessed through the Access Gateway. The Access Gateway Applications feature enables you to configure Access Gateway policies (Authorization, Traffic, Clientless Access, and TCP Compression) for an AppExpert application. After you configure NetScaler Gateway policies for AppExpert applications, you can include the policy configuration in the AppExpert application templates that you create.

You can also configure NetScaler Gateway policies for intranet subnets, file shares, and other network resources.

Finally, you can create bookmarks for AppExpert applications and certain resources if you want users to be able to access them from the NetScaler Gateway home page.

You can configure the entities in the NetScaler Gateway Applications feature only by using the configuration utility.

## How an NetScaler Gateway Application Works

Updated: 2013-07-17

When you create an AppExpert application in the Applications node in the configuration utility, a corresponding Access Gateway application is automatically created in the Access Gateway Applications node. Additionally, a rule that uses the AppExpert application's configured public endpoint is automatically created for the Access Gateway application entry. If multiple endpoints are configured for the AppExpert application, the rule includes all the configured public endpoints. The NetScaler appliance uses this rule to apply any configured Access Gateway policies to the traffic received at the AppExpert application's public endpoint. Traffic received at the AppExpert application's public endpoint is first evaluated against the NetScaler Gateway policies and then evaluated against the policies configured for AppExpert application's application units.

The rule that is created for the Clientless Access policies for an Access Gateway application is an advanced expression that also uses the public endpoint that is configured for the AppExpert application. Therefore, before you configure NetScaler Gateway policies for an AppExpert application, you must configure public endpoints for the AppExpert application.

When you include the NetScaler Gateway configuration in an application template, deployment-specific information, such as IP address and port information, and the rule that is created from this information are not included in the template.

## How a NetScaler Configuration for a File Share Works

On the NetScaler appliance, you can configure Authorization policies for a file share that is hosted on your organization's network.

When you create a file share, you specify a name for the file share and the network path to the file share. In the network path, you can specify either the name of the server or the server IP address. A rule that uses the components of the file share path is automatically created for the file share. This rule enables the appliance to identify requests for files hosted on the file share server. Any Authorization policies that are configured for the file share are applied to incoming requests.

The NetScaler configuration for a file share cannot be saved in AppExpert application templates.

## How a NetScaler Configuration for an Intranet Subnet Works

For the intranet subnets that form a part of your network, you can configure policies for Authorization, Traffic, and TCP Compression on the NetScaler appliance. When adding an intranet subnet, you specify the IP address and the netmask of the intranet subnet. A rule that uses these two parameters is automatically created for the intranet subnet. The appliance applies the configured policies to any request that has a destination IP address and netmask set to the subnet's IP address and netmask, respectively.

The NetScaler configuration for an intranet subnet cannot be saved in AppExpert application templates.

## How the Other Resources Category Works

Updated: 2013-07-18

The Other Resources category enables you to configure Access Gateway policies for any network resource by using a rule of your choice. When you configure the NetScaler appliance to process requests for the network resource, you configure a classic expression to identify the requests that are associated with the network resource. You can configure Authorization, Traffic, Clientless Access, and TCP Compression policies for a network resource in Other Resources. The NetScaler appliance applies the configured NetScaler Gateway policies to any requests that match the configured rule.

The NetScaler configuration for a network resource in Other Resources cannot be saved in AppExpert application templates.

## Entity Naming Conventions

The NetScaler Gateway Applications feature enforces a naming convention for some of the entities that you create in this feature. For example, the names of the profiles that you create for Traffic policies for an intranet subnet always begin with a string that consists of the name of the intranet subnet followed by an underscore (\_). The name that you provide for the entity is appended to this string. If the name of a subnet is "subnet1," the name of the profile begins with "subnet1\_." When such a naming convention is required (in the text box in which you type the name of an entity, for example), the user interface automatically inserts the string with which the name of the entity must begin and does not allow you to modify it.



## Adding File Shares

When creating a file share, you provide the network path to the file share. Any policies that you configure for a file share use the rule that is automatically created when you created the file share.

### To configure a file share

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, click File Shares, and then do one of the following:
  - o To add a file share, click File Shares, and then click Add.
  - o To modify a file share, click File Shares, and then click Open.
3. In the Create File Share or Configure File Share dialog box, do the following:
  - a. In the Name box, type a name for the file share you are adding. This parameter cannot be changed for an existing file share.
  - b. In the Path box, type the path to the file share.

The path to the file share may use either the name of the server or the IP address of the server.

- c. In Bookmark, in the Text to Display box, type a name for the file share as you would want it to appear on the Access Gateway home page.
- d. Click Create or OK, and then click Close.

## Adding Intranet Subnets

You can specify authorization and Traffic policies for traffic that is bound for the intranet subnets that are configured in your network. The rules for these policies are automatically created by using the parameters you specify for the subnet.

### To configure an intranet subnet

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, do one of the following:
  - To add an intranet subnet, click Intranet Subnets, and then click Add.
  - To modify an intranet subnet, click an intranet subnet, and then click Open.
3. In the Create Intranet Subnet or Configure Intranet Subnet dialog box, do the following:
  - a. In the Name box, type a name for the intranet subnet you are adding. This parameter cannot be changed for an existing intranet subnet.
  - b. In the IP Address box, type the IP address of the intranet subnet.
  - c. In the Netmask box, type the netmask that will be used for the intranet subnet.
  - d. Click Create or OK, and then click Close.

## Adding Other Resources

For a network resource that you add to Other Resources, you must configure a classic expression that identifies the subset of traffic associated with the resource. For more information about configuring a classic expression, see the .

### To configure a resource in Other Resources

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, do one of the following:
  - To add a resource, click Other Resources, and then click Add.
  - To modify a resource, click a resource, and then click Open.
3. In the Create Resource or Configure Resource dialog box, do the following:
  - a. In the Name box, type a name for the resource you are adding. This parameter cannot be changed for an existing resource.
  - b. In the Rule box, type the rule that will identify the subset of traffic that is associated with the resource you are adding.

Alternatively, click Configure, and then create the rule in the Create Expression dialog box.

- c. Click Create or OK, and then click Close.

## Configuring Authorization Policies

You can configure NetScaler Gateway authorization policies for AAA users and groups to access a resource.

### To configure permissions for a AAA user or group to access a resource

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, in the Authorization column, click the icon for the application, file share, intranet subnet, or resource for which you want to configure authorization policies for AAA users and groups.
3. Do one of the following:
  - If the AAA user or group for which you want to configure permissions is already in the Groups/Users tree, drag the user or group from the Groups/Users tree to the Users or Groups node in the <application name> tree. Then, right-click the user or group and click Allow.
  - If the AAA user or group for which you want to configure permissions is not configured on the appliance, in the <application name> tree, right-click Users or Groups, and then click Add. In the Create AAA Group or Create AAA User dialog box, fill in the values, click Create, and then click Close.

The user or group is created with the permission set to Allow. To change the permission setting, right-click the group or user, and then click the permission setting.

4. Click Close.

## Configuring Traffic Policies

The traffic policies that you configure for the resources in the NetScaler Gateway Applications node control client connections to the application. You do not have to configure a rule for the resource. The rule created automatically when you create the resource. You only need to associate a request profile with the traffic policy. In the traffic profile, you specify parameters such as the protocol, application time-out, and file type association.

### To configure traffic policies for a resource

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, in the Traffic column, click the icon provided for the application, file share, intranet subnet, or resource for which you want to configure traffic policies.
3. In the Configure Traffic Policies dialog box, do the following:
  - o To specify an existing traffic policy, click Insert Policy, and then, in the Policy Name column, click the name of the policy.
  - o To configure a new policy, click Insert Policy, and then, in the Policy Name column, click New Policy. In the Create Traffic Policy dialog box, in the Name box, after the underscore (\_), type a name for the policy. Then, in Request Profile, either select an existing request profile or click New to configure a new request profile. You can also select an existing profile and then click Modify to modify the profile.

For more information about configuring a traffic policy or profile, see NetScaler Gateway , Enterprise Edition at <http://edocs.citrix.com/>.

- o To modify a policy that you have inserted, in the Policy Name column, click the policy name, and then click Modify Policy. To modify only the associated profile, in the Profile column, click the name of the profile, and then click Modify Profile.
  - o To regenerate the priorities assigned to the policies, click Regenerate Priorities.
  - o To specify a new priority value for a policy, in the Priority column, double-click the assigned priority, and then enter the value you want.
  - o To unbind a policy, click the policy, and then click Unbind Policy.
4. Click Apply Changes, and then click Close.

## Configuring Clientless Access Policies

Clientless access, when configured for a resource on the NetScaler appliance, allows end-users to access the resource without using the NetScaler Gateway client software. Users can use web browsers to access resources such as Outlook Web Access. You configure clientless access for a resource by configuring a clientless access policy that is associated with a clientless access profile.

### To configure a clientless access policy for a resource in the NetScaler Gateway Applications node

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, in the Clientless Access column, click the icon for the application, file share, intranet subnet, or resource for which you want to configure a clientless access policy.
3. In the Configure Clientless Access Policies dialog box, do the following:

- To specify an existing clientless access policy, click Insert Policy, and then, in the Policy Name column, click the name of the policy.
- To configure a new clientless access policy, click Insert Policy, and then, in the Policy Name column, click New Policy. In the Create Clientless Access Policy dialog box, in the Name box, after the underscore (\_), type a name for the policy. Then, in Profile, either select an existing profile or click New to configure a new profile. You can also select an existing profile and then click Modify to modify the profile.

For more information about configuring a clientless access policy or profile, see NetScaler Gateway , Enterprise Edition at <http://edocs.citrix.com/>.

- To modify a policy that you have inserted, in the Policy Name column, click the policy name, and then click Modify Policy. To modify only the associated profile, in the Profile column, click the name of the profile, and then click Modify Profile.
  - To specify a new priority value for a policy, in the Priority column, double-click the assigned priority, and then enter the value you want.
  - To unbind a policy, click the policy, and then click Unbind Policy.
4. Click Apply Changes, and then click Close.

## Configuring TCP Compression Policies

You can configure TCP compression policies for an application to increase the performance of the application. TCP compression reduces network latency, reduces bandwidth requirements, and increases the speed of transmission. When configuring a TCP compression policy, you associate a compression action with the policy. The compression action specifies either Compress, GZIP, Deflate, or NoCompress as the compression type. For more information about the compression policies, and compression actions, see NetScaler Gateway , Enterprise Edition at <http://edocs.citrix.com/>.

### To configure a TCP compression policy for a resource in the NetScaler Gateway Applications node

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, in the TCP Compression column, click the icon for the application, file share, intranet subnet, or resource for which you want to configure a TCP compression policy.
3. In the Configure TCP Compression Policies dialog box, do the following:
  - o To specify an existing TCP compression policy, click Insert Policy, and then, in the Policy Name column, click the name of the policy.
  - o To create a new TCP compression policy, click Insert Policy, and then, in the Policy Name column, click New Policy. In the Create TCP Compression Policy dialog box, in the Policy Name box, after the underscore (â€œ\_â€•), type a name for the policy. Then, in Action, either select an existing action or click New and configure a new action. You can also click View to view the configured compression type.

For more information about configuring a TCP compression policy or action, see NetScaler Gateway , Enterprise Edition at <http://edocs.citrix.com/>.

- o To modify a policy that you have inserted, in the Policy Name column, click the policy name, and then click Modify Policy.
  - o To regenerate the priorities assigned to the policies, click Regenerate Priorities.
  - o To specify a new priority value for a policy, in the Priority column, double-click the assigned priority, and then enter the value you want.
  - o To unbind a policy, click the policy, and then click Unbind Policy.
4. Click Apply Changes, and then click Close.

## Configuring Bookmarks

You can configure bookmarks for an application or for a resource that you configure in the Other Resources category if you want the application or resource to be accessible from the NetScaler Gateway home page.

### To configure a bookmark for an NetScaler Gateway application or a resource in the Other Resources category

1. In the navigation pane of the NetScaler configuration utility, expand AppExpert, and then click Access Gateway Applications.
2. In the details pane, click the application or resource for which you want to configure a bookmark, and then click Configure Bookmark.
3. In the Create Bookmark dialog box, configure values for the parameters.

For more information about the parameters in the Create Bookmark dialog box, see NetScaler Gateway , Enterprise Edition at <http://edocs.citrix.com/>.

4. Click Create, and then click Close.



## AppQoE

Application level Quality of Experience (AppQoE) integrates several existing policy-based security features of the NetScaler appliance into a single integrated feature that takes advantage of a new queuing mechanism, fair queuing. Fair queuing manages requests to load-balanced web servers and applications at the virtual server level instead of at the service level, allowing it to handle queuing of all requests to a web site or application as one group before load balancing, instead of as separate streams after load balancing.

The features that are integrated into AppQoE are , , and . Collectively these services provide protection against a number of problems:

- **Simple overload.** Any server, no matter how robust, can accept only a limited number of connections at one time. When a protected web site or application receives too many requests at once, the Surge Protection feature detects the overload and queues the excess connections til the server can accept them. The Priority Queuing feature ensures that whoever most needs access to a resource is provided access without having to wait behind other lower-priority requests. The SureConnect feature displays an alternate web page that notifies users that the resource that they requested is not available.
- **Denial-of-Service (DOS) attacks.** Any public-facing resource is vulnerable to attacks whose purpose is to bring that service down and deny legitimate users access to it. The Surge Protection, Priority Queuing, and SureConnect features help manage DOS attacks as well as other types of high load. In addition, the HTTP Denial-of-Service Protection feature targets DOS attacks against your web sites, sending challenges to suspected attackers and dropping connections if the clients do not send an appropriate response.

Until the current version of the NetScaler operating system, these features were implemented at the service level, which means that each service was assigned its own queues. While service-level queues work, they also have some disadvantages, most of which are due to the NetScaler appliance having to load balance requests before implementing any of the protection features that rely on queuing. Implementing protection features before queuing has a number of advantages, some of which are listed below:

- Absolute priority of connections as configured in the priority queuing feature can be maintained.
- Connections are not flushed if a service transitions state, as they are in a service-level queue.
- During periods of high load, such as a denial-of-service attack, HTTP DoS and SureConnect come into play before load balancing, allowing these features to detect and divert unwanted or lower-priority traffic from the load balancer before the load balancer must cope with it.

In addition to implementing fair queuing, AppQoE integrates a set of features that each provide a different set of tools to achieve a common goal: protecting your networked resources from excessive or inappropriate demand. Putting these features into a common framework enables you to configure and implement them more easily.

# Enabling AppQoE

To configure AppQoE, you must first enable the feature.

## To enable AppQoE by using the command line

At the command prompt, type the following commands:

- enable ns feature appqoe
- show ns feature

### Example

```
> enable ns feature appqoe
Done
> show ns feature
```

	Feature	Acronym	Status
	-----	-----	-----
1)	Web Logging	WL	ON
2)	Surge Protection	SP	ON
3)	Load Balancing	LB	ON
...			
29)	<b>AppQoE</b>	<b>AppQoE</b>	<b>ON</b>
Done			

## To enable AppQoE by using the configuration utility

1. Navigate to System > Settings.
2. In the details pane, click Configure Advanced Features.
3. In the Configure Advanced Features dialog box, select the AppQoE check box.
4. Click OK.

## AppQoE Actions

After enabling the AppQoE feature, you must configure one or more actions for handling requests.

Important: No specific individual parameters are required to create an action, but you must include at least one parameter or you cannot create the action.

### To configure an AppQoE action by using the command line

At the command prompt, type the following commands:

- add appqoe action <name> [-priority <priority>] [-respondWith (ACS|NS) [<customfile>] [-altContentSvcName <string>] [-altContentPath <string>] [-maxConn <positive\_integer>] [-delay <usecs>] [-polqDepth <positive\_integer>] [-priqDepth <positive\_integer>] [-dosTrigExpression <expression>] [-dosAction ( SimpleResponse | HICResponse )]
- show appqoe action

#### Example

To configure priority queuing with policy queue depths of 10 and 1000 for medium and lowest priority queues, respectively:

```
> add appqoe action appqoe-act-basic-prhigh -priority HIGH
Done

> add appqoe action appqoe-act-basic-prmedium -priority MEDIUM -polqDepth 10
Done

> add appqoe action appqoe-act-basic-prlow -priority LOW -polqDepth 1000
Done

> show appqoe action
1)      Name: appqoe-act-basic-prhigh
        ActionType: PRIORITY_QUEUEING
        Priority: HIGH
        PolicyQdepth: 0
        Qdepth: 0

2)      Name: appqoe-act-basic-prmedium
        ActionType: PRIORITY_QUEUEING
        Priority: MEDIUM
        PolicyQdepth: 10
        Qdepth: 0

3)      Name: appqoe-act-basic-prlow
        ActionType: PRIORITY_QUEUEING
        Priority: LOW
        PolicyQdepth: 1000
        Qdepth: 0

Done
```

### To modify an existing AppQoE action by using the command line

At the command prompt, type the following commands:

- set appqoe action <name> [-priority <priority>] [-altContentSvcName <string>] [-altContentPath <string>] [-polqDepth <positive\_integer>] [-priqDepth <positive\_integer>] [-maxConn <positive\_integer>] [-delay <usecs>] [-dosTrigExpression <expression>] [-dosAction ( SimpleResponse | HICResponse )]
- show appqoe action

### To remove an AppQoE action by using the command line

At the command prompt, type the following commands:

- rm appqoe action <name>
- show appqoe action

## Parameters for configuring an AppQoE action

### name

A name for the new action, or the name of the existing action that you want to modify. The name can begin with a letter, number, or the underscore symbol, and can consist of from one to letters, numbers, and the hyphen (-), period (.) pound (#), space ( ), at sign (@), equals (=), colon (:), and underscore (\_) symbols.

### priority

The priority queue to which the request is assigned. When a protected web server or application is heavily loaded and cannot accept additional requests, specifies the order in which waiting requests are to be fulfilled when resources are available. The choices are:

1. **HIGH.** Fulfills the request as soon as resources are available.
2. **MEDIUM.** Fulfills the request after it has fulfilled all requests in the HIGH priority queue.
3. **LOW.** Fulfills the request after it has fulfilled all requests in the HIGH and MEDIUM priority queues.
4. **LOWEST.** Fulfills the request only after it has fulfilled all requests in higher-priority queues.

If priority is not configured, then the NetScaler appliance assigns the request to the LOWEST priority queue by default.

### respondWith

Configures the NetScaler ADC to take the specified Responder action when the specified threshold is reached. Must be used with one of the following settings:

- o **ACS:** Serves content from an alternate content service. Threshold: maxConn (maximum connections) or delay.
- o **NS:** Serves a built-in response from the NetScaler ADC. Threshold: maxConn (maximum connections) or delay.
- o **NO ACTION:** Serves no alternative content. Assigns connections to the LOWEST priority queue if the maxConn (maximum connections) or delay threshold is reached.

### altContentSvcName

If `-respondWith ACS` is specified, the name of the alternative content service, usually an absolute URL to the web server that hosts the alternate content.

### altContentPath

If `-respondWith ( ACS | NS )` is specified, the path to the alternative content.

### polqDepth

Policy queue depth threshold value for the policy queue associated with this action. When the number of connections in the policy queue associated with this action increases to the specified number, subsequent requests are assigned to the LOWEST policy queue. Minimum value: 1 Maximum value: 4,294,967,294

### priqDepth

Policy queue depth threshold value for the specified priority queue. If the number of requests in the specified queue or the virtual server to which the policy associated with the current action is bound increases to the specified number, subsequent requests are assigned to the LOWEST priority queue. Minimum value: 1 Maximum value: 4,294,967,294

### maxConn

The maximum number of connections that can be open for requests that match the policy rule. Minimum value: 1 Maximum value: 4,294,967,294

### delay

The delay threshold, in microseconds, for requests that match the policy rule. If a matching request has been delayed for longer than the threshold, the NetScaler appliance performs the specified action. If NO ACTION is specified, then the appliance assigns requests to the LOWEST priority queue. Minimum value: 1 Maximum value: 599999,999

### dosTrigExpression

Adds an optional second-level check to trigger DoS actions.

### dosAction

Action to take when the ADC determines that it or a protected server is under DoS attack. Possible values: SimpleResponse, HICResponse

## To configure an AppQoE action by using the configuration utility

1. Navigate to App-Expert > AppQoE > Actions.
2. In the details pane, do one of the following:
  - o To create a new action, click Add.
  - o To modify an existing action, select the action, and then click Edit.
3. In the Create AppQoE Action or the Configure AppQoE Action screen, type or select values for the parameters. The contents of the dialog box correspond to the parameters described in "Parameters for configuring the AppQoE Action" as follows (asterisk indicates a required parameter):
  - o Nameâ€"name
  - o Action typeâ€"respondWith

- o Priorityâ€™priority
  - o Policy Queue Depthâ€™polqDepth
  - o Queue Depthâ€™priqDepth
  - o DOS Actionâ€™dosAction
4. Click Create or OK.

## AppQoE Parameters

In the AppQoE parameters, you configure the session life of an AppQoE session, the file name of the file containing the customized response, and the number of client connections that can be placed in a queue.

### To configure the AppQoE parameter settings by using the command line

At the command prompt, type the following commands:

- set appqoe parameter [-sessionLife <secs>] [-avgwaitingclient <positive\_integer>] [-MaxAltRespBandWidth <positive\_integer>] [-dosAttackThresh <positive\_integer>]
- show appqoe parameter

### Parameters for configuring the AppQoE parameters

sessionLife

Number of seconds to wait after displaying alternate content before the ADC displays the same content again.  
Default value: 300 Minimum value: 1 Maximum value: 4,294,967,294

avgwaitingclient

The average number of client requests that can be in the service waiting queue. Default value: 1000000 Maximum value: 4,294,967,294

MaxAltRespBandWidth

The maximum bandwidth to consume when sending alternate responses. If the maximum is reached, the ADC quits sending the alternate content til bandwidth consumption drops. Default value: 100 Minimum value: 1 Maximum value: 4,294,967,294

dosAtckThrsh

The denial-of-service attack threshold. The number of connections that must be waiting in queues before the ADC responds with DoS protection measures. Default value: 2000 Minimum value: 0 Maximum value: 4,294,967,294

### To configure the AppQoE parameter settings by using the configuration utility

1. Navigate to AppExpert > AppQoE.
2. In the details pane, click Configure AppQoE Parameters.
3. In the Configure AppQoE params screen, type or select values for the parameters. The contents of the dialog box correspond to the parameters described in "Parameters for configuring the AppQoE Parameters" as follows (asterisk indicates a required parameter):
  - Session Life (secs)â€™sessionLife
  - Average waiting clientâ€™avgwaitingclient
  - Alternate Response Bandwidth Limit(Mbps) â€™MaxAltRespBandWidth
  - DOS Attack Threshold â€™dosAttackThresh
4. Click OK.

## AppQoE Policies

To implement AppQoE, you must configure at least one policy to tell your NetScaler ADC how to distinguish the connections to be queued in a specific queue.

### To configure an AppQoE policy by using the command line

At the command prompt, type the following command:

```
add appqoe policy <name> -rule <expression> -action <string>
```

#### Example

The following example selects requests with a User-Agent header that contains "Android", and assigns them to the medium priority queue. These requests come from smartphones and tablets that run the Google Android operating system.

```
> add appqoe action appqoe-act-primd -priority MEDIUM
Done
> add appqoe policy appqoe-pol-primd -rule "HTTP.REQ.HEADER(\"User-Agent\").CONTAINS(\"Andro
Done
> sh appqoe policy appqoe-pol-primd
    Name: appqoe-pol-primd
    Rule: HTTP.REQ.HEADER("User-Agent").CONTAINS("Android")
    Action: appqoe-act-primd
    Hits: 0
```

Done

### Parameters for configuring an AppQoE policy

#### name

A name for the AppQoE policy. The name can begin with a letter, number, or the underscore symbol, and can consist of from one to 127 letters, numbers, and the hyphen (-), period (.), pound (#), space ( ), at sign (@), equals (=), colon (:), and underscore (\_) symbols. You should choose a name that helps identify the type of action.

#### rule

A NetScaler expression that tells the appliance which connections it should handle. For complete information about policy expressions, see the *Citrix NetScaler Policy Configuration and Reference Guide* at .

#### action

The AppQoE action to perform when a connection matches the policy.

### To configure an AppQoE policy by using the configuration utility

1. Navigate to App-Expert > AppQoE > Policies.
2. In the details pane, do one of the following:
  - To create a new policy, click Add.
  - To modify an existing policy, select the policy, and then click Edit.
3. If you are creating a new policy, in the Create AppQoE Policy dialog, in the Name text box, type a name for your new policy.

The name can begin with a letter, number, or the underscore symbol, and can consist of from one to 127 letters, numbers, and the hyphen (-), period (.), pound (#), space ( ), at sign (@), equals (=), colon (:), and underscore (\_) symbols. You should choose a name that helps identify the purpose and effect of this policy.

If you are modifying an existing policy, skip this step. You cannot change the name of an existing policy.

4. In the Action drop-down list, choose the AppQoE action to perform when the policy matches a connection. Click the plus (+) to open the Add AppQoE Action dialog and add a new action.
5. In the Rule text box, either enter the policy expression directly, or click New to create a policy expression. If you click New, perform the following steps:
  - a. In the Create Expression dialog box, click Add.
  - b. In the Add Expression dialog box, select a common expression from the Frequently Used Expressions drop-down list, or use the Construct Expression drop-down lists to create the expression that defines which traffic to filter.

If you choose to create your own expression, you start by selecting the first term from the first drop-down list on the left side of the Construct Expression area. The choices in that list are:

- HTTP: All traffic to port 80 and port 443.

- o SYS:
- o CLIENT:
- o SERVER:
- o ANALYTICS:
- o TEXT:

The default choice is HTTP. After you make a choice in the first drop-down list (or accept the default), you can choose the next term in your expression from the drop-down list to the right of it. The terms in that list and other lists that follow change depending on your previous choices; the lists offer only terms that are valid choices. Continue to select terms until you have finished the expression.

Use the Help and Preview Expression areas for assistance when creating the expression. For a complete description of the available choices, see the *Citrix NetScaler Policy Configuration and Reference Guide* at .

- c. When you have created the expression that you want, click OK. The expression is added in the Expression text box.
6. Click Create. The expression appears in the Rule text box.



## Entity Templates

An entity template is a collection of configuration information for an individual entity on a Citrix® NetScaler® appliance. It provides a specification and a set of defaults for a configurable NetScaler entity, such as a policy, virtual server, service, or action. By using a template that defines a set of defaults, you can quickly configure multiple entities that require a similar configuration while eliminating several configuration steps.

Entity templates are available only in the configuration utility. You use the NetScaler configuration utility to create, manage, and use any type of entity template. You can share entity templates with other administrators and manage local folders that contain the templates. You can also import entity templates from and export entity templates to your local computer.

Before creating a template, you should be familiar with the configuration of the entity.

Note: You use entity templates to configure individual entities. To configure multiple entities related to a particular Web application, you must use an application template. For more information, see "[AppExpert Applications and Templates](#)."

### How Entity Templates Work

When you create a template for a NetScaler entity, you specify default values for the entity. You specify what values must be read-only, what values must not be displayed, and what values users can configure. You also configure the pages that compose the template import wizard. All the information and settings you provide are stored in the template file.

When a user imports the entity template to a NetScaler appliance, a wizard guides the user through the various pages that you configured for the template. The wizard displays the read-only parameter values and prompts the user to specify values for the configurable parameters. After the user follows the instructions in the wizard, the appliance creates the entity with the configured values.

For example, you can create an entity template for HTTP services that provides a text box for a service name and assigns preset values for the service protocol, timeouts, thresholds, and monitors. Later, when you use the template to create new HTTP services, a wizard prompts you for a service name and supplies the preset values that you would otherwise have configured manually.

The procedure for creating entity templates for load balancing virtual servers is different than the AppExpert procedure for creating other entity templates. For more information, see "[Creating an Entity Template](#)."

In addition, the procedure for using the template to create the load balancing virtual server entity is different. For more information, see "[Creating an Entity from a Template](#)."

## Configuring an Entity Template

You can create or modify an entity template either from the AppExpert feature node or from the associated NetScaler feature node for the entity. For example, you can create a content switching virtual server entity template in either the AppExpert feature node or the content switching feature node in the configuration utility.

If you create a template that is not based on an existing entity, you can specify the following options and settings for the template:

- The default value of a parameter.
- Whether the default values are visible to users.
- Whether the default values can be changed by users.
- The number of pages in the entity import wizard, including the page names, text, and available parameters.
- The entities that must be bound to the entity for which the template is being created.

For example, when you are creating a cache redirection virtual server template, you can specify the policies that you want to bind to the cache redirection virtual servers that you create from the template. However, only binding information is included in the template. The bound entities are not included. If the entity template is imported to another NetScaler appliance, the bound entities must exist on the appliance at import time for the binding to succeed. If none of the bound entities exist on the target appliance, the entity (for which the template was configured) is created without any bindings. If only a subset of the bound entities exist on the target appliance, they are bound to the entity that is created from the template.

When you create a template based on an existing entity, the configuration settings of the entity appear in the template. All bound entities are selected by default, but you can modify bindings as necessary. As in the case of a template that is not based on an existing entity, only binding information is included and not the entities. You can either save the template with the existing configuration settings or use the settings as a basis for creating a new configuration for a template.

## Creating an Entity Template

You can create entity templates in either the AppExpert node or the NetScaler feature node that corresponds to the type of entity. For example, you can create a content switching virtual server template from the entity templates tab of the AppExpert feature's Templates node or in the Content Switching node. You can also specify the parameters that you want the template to store, and specify whether you want the template import wizard to prompt the user for certain parameter values.

However, when creating load balancing virtual servers, you do not have the option of specifying parameter values that you want stored in the template. You create a load balancing virtual server template by selecting an existing load balancing virtual server and configuring any variables that you might want to create in existing parameters and bound policies. The variables can be assigned values when you create a load balancing virtual server from the template. The template stores load balancing parameters such as the virtual server's IP address and port number, bound policies, actions, and variable definitions. A deployment file is also created, automatically, from the load balancing configuration. The deployment file stores deployment-specific information, such as information about bound services, service groups, and the name-value pairs of variables. If the bound entities that are included in the template are already configured on the NetScaler appliance to which the template is imported, duplicates are created, with names that are generated automatically in a particular format. The duplicate entities are based on the parameter information stored in the entity template.

When you create a load balancing virtual server template from the AppExpert node, the template is always saved to the `/nsconfig/nstemplates/entities/lb_vserver/` folder. If you want to save the template to a different folder, create the template from the Virtual Servers pane in the Load Balancing node. The deployment file is created with the name with which you save the template file, but with the string `_deployment` appended to the name. The deployment file is saved to the `/nsconfig/nstemplates/entities/lb_vserver/deployment_files/` folder. For more information about deployment files for load balancing virtual server templates, see "[Understanding Load Balancing Entity Templates and Deployment Files](#)."

Note: You can use either of the first two procedures for creating any template, except for a load balancing virtual server template. For creating a load balancing virtual server template, use the third or fourth procedure.

### To create an entity template by using the AppExpert feature node

1. Navigate to AppExpert > Templates.
2. In the details pane, on the Entity Templates tab, do one of the following:
  - o To create a new template, click Add. In the Select the Template Type dialog box, select the template type, and then click OK.
  - o To create a duplicate of an existing entity template, in the details pane, select the entity template, and then click Add.
3. In the Create...Template dialog box, follow the instructions to create a template.

If you are creating a duplicate of an existing entity template, in the Create...Template dialog box, on the Specify Template Name page, you must change the name of the entity template.

4. Click Finish, and then click Exit.

### To create an entity template by using its corresponding feature node

1. Navigate to Traffic Management, and select the feature (for example, Content Switching), and then select the entity (for example, Virtual Servers), for which you want to create the entity template.
2. At the top of the details pane, click Entity Templates, and then click Create Template.
3. In the Create...Template dialog box, follow the instructions to create a template.
4. Click Finish, and then click Exit.

### To create a load balancing virtual server template from the AppExpert node

1. Navigate to AppExpert > Templates.
2. In the details pane, on the LB Templates tab, click Add.
3. In the Select Load Balancing Virtual Server dialog box, select the load balancing virtual server whose configuration you want to save to a template file, and then click OK.
4. In the Create Template dialog box, provide the following information:
  - o Name. The name of the template.  
Note: The Folder field shows the location to which the template will be saved. You cannot modify the path that is displayed.
  - o Configure Variables. Configure variables for the load balancing template. For more information, see "[Configuring Variables in Load Balancing Virtual Server Templates](#)."
  - o Introduction Description. A description of the virtual server for which you are creating a template.

- Summary Description. A summary of the configuration or additional instructions for other administrators, such as a description of any additional steps that need to be followed after the entity is successfully created.
- Author. The creator of the template.
- Major. An optional major version number of your choice, to be specified if you want to maintain versions of your template.
- Minor. An optional minor version number of your choice, to be specified if you want to maintain minor versions of your template.

You can maintain versions by incrementing one or both of the version numbers each time you maintain the template. The Entity Template Wizard concatenates and displays the major and minor version numbers during import. For example, if the major version number is 1 and the minor version is 1, the Entity Template Wizard displays a version number of 1.1.

5. Click OK.

## To create a load balancing virtual server template from the Load Balancing Virtual Servers pane

1. Navigate to Traffic Management > Load Balancing > Virtual Servers.
2. In the details pane, select the virtual server on which to base the template,, and then click Create Template. You might have to click the scroll arrow at the bottom right of the pane to bring the Create Template button into view.
3. In the Create Template dialog box, provide the following information:
  - Name. The name of the template.
  - Folder. The location to which the template will be saved.  
Note: If you want to save the template to the appliance, you can save it only to the `/nsconfig/nstemplates/entities/lb_vserver/` directory (the path displayed by default in Folder. If you want to save the template file to a folder on your computer, click the down-arrow on the Browse button, click Local, and then select a folder.
  - Configure Variables. Configure variables for the load balancing template. For more information, see "[Configuring Variables in Load Balancing Virtual Server Templates.](#)"
  - Introduction Description. A description of the virtual server for which you are creating a template.
  - Summary Description. A summary of the configuration or additional instructions for other administrators, such as a description of any additional steps that need to be followed after the entity is successfully created.
  - Author. The creator of the template.
  - Major. An optional major version number of your choice, to be specified if you want to maintain versions of your template.
  - Minor. An optional minor version number of your choice, to be specified if you want to maintain minor versions of your template.

You can maintain versions by incrementing one or both of the version numbers each time you maintain the template. The Entity Template Wizard concatenates and displays the major and minor version numbers during import. For example, if the major version number is 1 and the minor version is 1, the Entity Template Wizard displays a version number of 1.1.

4. Click OK.

## Configuring Variables in Load Balancing Virtual Server Templates

Load balancing virtual server templates support the declaration of variables in the configured load balancing parameters and in bound policies and actions. The ability to declare variables enables you to replace preconfigured values with values that suit the environment into which you are importing the template. The Entity Template Wizard, which appears when you import a template, includes a Specify Variable Values page on which you can specify appropriate values for the variables that are configured for the entity template. This wizard page appears only when you import a template that is configured with existing variables.

As an example, consider the following expression configured for a policy that is bound to a load balancing virtual server for which you are creating a template. The expression evaluates the value of the Accept-Language header in an HTTP request.


```
HTTP.REQUEST.HEADER( "Accept-Language" ).CONTAINS( "en-us" )
```

If you want the value of the header to be configurable at import time, you can specify the string `en-us` as a variable. When importing the template, you can specify a new value for the variable on the Specify Variable Values page.

After you create a variable, you can do the following:

- Assign additional strings to an existing variable. After you create a variable for a string, you can select and assign other parts of the same or different expression to the variable. The strings you assign to a variable need not be the same. At import time, all the strings that are assigned to the variable are replaced with the value that you provide.
- View the string or strings that are assigned to the variable.
- View a list of all the entities and parameters that use the variable.

## To configure variables in a load balancing virtual server template

1. Navigate to Traffic Management > Load Balancing > Virtual Servers.
2. In the details pane, right-click the virtual server that you want to export to a template file, and then click Create Template.
3. In the Create Template dialog box, modify the default template file name if required, specify the location where you want to save the template, and then click Configure Variables.
4. In the Configure Variables dialog box, click the tab that lists the entity for which you want to configure a variable, select the entity, and then click Configure Variables.
5. In the Variables for <Entity Type>: <Entity Name> dialog box, click the  button next to the parameter value or expression in which you want to create a variable.
6. In the Variables for <Field Name> dialog box, do the following:

- To create a variable, in the text box that displays the configured expression or value, select the string that you want to be configurable at import time, and then click Add. In the Create Variable dialog box, specify a name and a description for the variable, and then click Create.

The name of the variable, its value, and the description you provided appear in the Available Variables listing in the dialog box. The name you provide will be the name of the associated field in the template import wizard, and the description will appear as alt text when the user positions the mouse pointer over the field.

- To modify a variable, in the Available Variables list, click the variable, and then click Open. In the Create Variable dialog box, modify the value and the description, and then click OK.

The new value that you specify will not replace the text selected in the text box that displays the configured expression or value. However, when you import the template, the new value will be displayed as the default value for the variable in the template import wizard.

- To view all the strings that are assigned to a given variable, in the Available Variables listing, click the name of the variable. The strings that are assigned to the variable are highlighted.
- To view a list of all the parameters, expressions, and actions in which the variable is used, in the Available Variables listing, click the variable whose references you want to view, and then click Show References.

- To assign a string to an existing variable, in the text box that displays the expression you configured, select the string you want to assign to an existing variable, right-click the selection, click Use existing Variable, and then click the name of the variable to which you want to assign the string.

If a variable has multiple strings assigned to it, when you specify a new value for the variable during import, all strings assigned to the variable are replaced with the new value.

7. Click Close.

## Modifying an Entity Template

You can modify only the parameters, bindings, and pages configured for a template. The name and location of the template specified when the template was created cannot be changed. The NetScaler appliance does not provide you with the option of modifying a load balancing virtual server template.

### To modify an entity template by using the AppExpert feature node

1. Navigate to AppExpert > Templates.
2. In the details pane, on the Entity Templates tab, select the template you want to change, and then click Open.
3. In the Modify...Template dialog box, follow the instructions to modify a template.
4. Click Finish, and then click Exit.

### To modify an entity template by using its corresponding feature node

1. Navigate to Traffic Management, select the feature (for example, Content Switching), and then select the entity (for example, Virtual Servers) for which you want to modify the entity template.
2. At the top of the details pane, click Entity Templates, and then click Manage Template.
3. In the Manage <feature entity name> Entity Templates dialog box, select the template that you want to modify, and then click Modify.
4. In the Modify <template name> Template dialog box, follow the instructions to modify a template.
5. Click Finish, and then click Exit.
6. Click Close.

## Deleting an Entity Template

Deleting an entity template does not affect any objects that have been created by using the template. You can delete a load balancing virtual server template only from the AppExpert feature node.

### To delete an entity template by using the AppExpert feature node

1. Navigate to AppExpert > Templates.
2. In the details pane, on the Entity Templates tab, click the template you want to delete, and then click Remove.

### To delete an entity template by using its corresponding feature node

1. Navigate to Traffic Management, and select the feature (for example, Content Switching) and then select the entity (for example, Virtual Servers), for which you want to delete the entity template.
2. At the top of the details pane, click Entity Templates, and then click Manage Template.
3. In the Manage...Entity Templates dialog box, select the template that you want to delete, and then click Delete.



## Creating an Entity from a Template

You can create an entity from an entity template either from the AppExpert feature node in the NetScaler configuration utility or from the NetScaler feature node that corresponds to the type of entity that you want to create. For example, you can create a content switching virtual server from a template with either the AppExpert feature node or the content switching feature node in the configuration utility.

The procedure for creating a load balancing virtual server from a template is different than the AppExpert procedure for creating other entities from templates.

After you create an instance of an entity using an entity template, you can configure it in the same way that you would any other object of that type, such as by using the configuration utility or the command line.

### To create an entity from a template by using the AppExpert feature node

1. Navigate to AppExpert > Templates.
2. In the details pane, do one of the following:
  - a. To create any entity other than a load balancing virtual server from a template, on the Entity Templates tab, click the template that you want to use, and then click Use Template.
  - b. To create a load balancing virtual server from a template, on the LB Templates tab, click the template that you want to use, and then click Use Template.
3. In the <Entity Template Name> wizard, follow the instructions to create the entity on the NetScaler.
4. Click Finish, and then click Exit.

### To create an entity from a template by using its corresponding feature node

1. Navigate to Traffic Management, and expand a feature node (for example, Content Switching), and then click an entity subnode (for example, Virtual Servers).
2. At the top of the details pane, click Entity Templates, and then click Use Template.
3. Click the name of the template that you want to use.
4. In the Use <template name> Template wizard, follow the instructions to create the entity.

Only templates that match the current context are displayed. For example, in the details pane for content switching virtual servers, only entity templates for content switching virtual servers appear, if configured.

5. Click Finish, and then click Exit.

### To create a load balancing virtual server by using a load balancing virtual server template

1. Navigate to Traffic Management > Load Balancing > Virtual Servers.
2. In the details pane, click Use Template.
3. In the Entity Template Wizard, follow the instructions to create a load balancing virtual server on the NetScaler.

Only templates that match the current context are displayed. For example, when you click Browse (Appliance), only entity templates for load balancing virtual servers appear, if configured.

4. Click Finish, and then click Exit.

Note: The Entity Template Wizard includes a Specify Variable Values page on which you can specify new values for variables. For more information about configuring variables in load balancing virtual server templates, see "[Configuring Variables in Load Balancing Virtual Server Templates](#)."

## Managing Entity Template Folders

You can organize only load balancing virtual server template folders.

### To organize load balancing virtual server template folders

1. Navigate to AppExpert > Templates > LB Templates.
2. In the Manage LB Templates dialog box, do one of the following:

- To change the name of a folder, select the folder and click Rename.

You can also click the folder that you want to rename, and then press F2. You cannot rename the top-level default folder.

- To remove the folder, select the folder and click Delete.

You can also click the folder that you want to remove, and then press the Delete key. You cannot remove the top-level default folder.

3. Click Close.

## Uploading and Downloading Entity Templates

You can import the entity templates that are stored on your local computer. You can also download entity templates from the NetScaler appliance to your local computer and then import them to other NetScaler appliances.

Note: You cannot upload or download load balancing virtual server templates.

### To upload an entity template to the NetScaler appliance

1. Navigate to Traffic Management, and expand a feature node (for example, Content Switching), and then click a subnode (for example, Virtual Servers) for which you want to upload an entity template.
2. At the top of the details pane, click Entity Templates, and then click Manage Template.
3. In the Manage...Entity Templates dialog box, click the top-level folder, and then click Upload.
4. In the Upload Entity Template dialog box, navigate to the template file that you want to upload, and then click Select.
5. Click Close.

### To download an entity template from the NetScaler appliance

1. Navigate to Traffic Management, and expand a feature node (for example, Content Switching), and then click a subnode (for example, Virtual Servers) for which you want to upload an entity template.
2. At the top of the details pane, click Entity Templates, and then click Manage Template.
3. In the Manage...Entity Templates dialog box, click the template that you want to download, and then click Download.
4. In the Download Entity Template dialog box, navigate to the location at which you want to save the template on your local computer, enter a file name, and then click Save.
5. Click Close.

## Understanding Load Balancing Entity Templates and Deployment Files

Load balancing entity templates are created in the same way that NetScaler application templates are created. When you export a load balancing virtual server to a template file, the following two files are automatically created:

- **Load balancing virtual server template file.** Contains XML elements that store the values of the parameters that are configured for the load balancing virtual server. The file also contains XML elements for storing information about bound policies.
- **Deployment file.** Contains XML elements that store deployment-specific information such as services, service groups, and configured variables.

In the template and deployment files, each unit of configuration information is encapsulated in a specific XML element that is meant for that unit type. For example, the load balancing method parameter, `lbMethod`, is encapsulated within the `<lbmethod>` and `</lbmethod>` tags.

Note: After you export a load balancing virtual server, you can add elements, remove elements, and modify existing elements before importing the configuration information to a NetScaler appliance.

### Example of a Load Balancing Virtual Server Template

Following is an example of a template file that was created from a load balancing virtual server called "Lbvip":

```
<?xml version="1.0" encoding="UTF-8" ?>
<template>
  <template_info>
    <entity_name>Lbvip</entity_name>
    <version_major>10</version_major>
    <version_minor>0</version_minor>
    <build_number>40.406</build_number>
  </template_info>
  <entitytemplate>
    <lbvserver_list>
      <lbvserver>
        <name>Lbvip</name>
        <servicetype>HTTP</servicetype>
        <ipv46>0.0.0.0</ipv46>
        <ipmask>*</ipmask>
        <port>0</port>
        <range>1</range>
        <persistencetype>NONE</persistencetype>
        <timeout>2</timeout>
        <persistencebackup>NONE</persistencebackup>
        <backuppersistencetimeout>2</backuppersistencetimeout>
        <lbmethod>LEASTCONNECTION</lbmethod>
        <persistmask>255.255.255.255</persistmask>
        <v6persistmasklen>128</v6persistmasklen>
        <pq>OFF</pq>
        <sc>OFF</sc>
        <m>IP</m>
        <datalength>0</datalength>
        <dataoffset>0</dataoffset>
        <sessionless>DISABLED</sessionless>
        <state>ENABLED</state>
        <connfailover>DISABLED</connfailover>
        <clttimeout>180</clttimeout>
        <somethod>NONE</somethod>
        <sopersistence>DISABLED</sopersistence>
        <sopersistencetimeout>2</sopersistencetimeout>
        <redirectportrewrite>DISABLED</redirectportrewrite>
        <downstateflush>DISABLED</downstateflush>
        <gt2gb>DISABLED</gt2gb>
        <ipmapping>0.0.0.0</ipmapping>
        <disableprimaryondown>DISABLED</disableprimaryondown>
        <insertvserveripport>OFF</insertvserveripport>
        <authentication>OFF</authentication>
        <authn401>OFF</authn401>
        <push>DISABLED</push>
        <pushlabel>none</pushlabel>
        <l2conn>OFF</l2conn>
        <appflowlog>DISABLED</appflowlog>
      </lbvserver>
    </lbvserver_list>
  </entitytemplate>
</template>
```

```

    <icmpvsrresponse>PASSIVE</icmpvsrresponse>
    <lbvserver_cmppolicy_binding_list>
      <lbvserver_cmppolicy_binding>
        <name>Lbvip</name>
        <polycyname>NOPOLICY-COMPRESSION</polycyname>
        <priority>100</priority>
        <gotopriorityexpression>END</gotopriorityexpression>
        <bindpoint>REQUEST</bindpoint>
      </lbvserver_cmppolicy_binding>
    </lbvserver_cmppolicy_binding_list>
  </lbvserver>
</lbvserver_list>
</entitytemplate>
</template>

```

## Example of a Deployment File

Following is the deployment file associated with the virtual server in the preceding example:

```

<?xml version="1.0" encoding="UTF-8" ?>
<template_deployment>
  <template_info>
    <entity_name>Lbvip</entity_name>
    <version_major>10</version_major>
    <version_minor>0</version_minor>
    <build_number>40.406</build_number>
  </template_info>
  <service_list>
    <service>
      <ip>1.2.3.4</ip>
      <port>80</port>
      <servicetype>HTTP</servicetype>
    </service>
  </service_list>
  <servicegroup_list>
    <servicegroup>
      <name>svcgrp</name>
      <servicetype>HTTP</servicetype>
      <servicegroup_servicegroupmember_binding_list>
        <servicegroup_servicegroupmember_binding>
          <ip>1.2.3.90</ip>
          <port>80</port>
        </servicegroup_servicegroupmember_binding>
        <servicegroup_servicegroupmember_binding>
          <ip>1.2.8.0</ip>
          <port>80</port>
        </servicegroup_servicegroupmember_binding>
        <servicegroup_servicegroupmember_binding>
          <ip>1.2.8.1</ip>
          <port>80</port>
        </servicegroup_servicegroupmember_binding>
        <servicegroup_servicegroupmember_binding>
          <ip>1.2.9.0</ip>
          <port>80</port>
        </servicegroup_servicegroupmember_binding>
      </servicegroup_servicegroupmember_binding_list>
    </servicegroup>
  </servicegroup_list>
</template_deployment>

```

## HTTP Callouts

For certain types of requests, or when certain criteria are met during policy evaluation, you might want to stall policy evaluation briefly, retrieve information from a server, and then perform a specific action that depends on the information that is retrieved. At other times, when you receive certain types of requests, you might want to update a database or the content hosted on a Web server. HTTP callouts enable you to perform all these tasks.

An HTTP callout is an HTTP or HTTPS request that the NetScaler appliance generates and sends to an external application when certain criteria are met during policy evaluation. The information that is retrieved from the server can be analyzed by default syntax policy expressions, and an appropriate action can be performed. You can configure HTTP callouts for HTTP content switching, TCP content switching, rewrite, responder, and for the token-based method of load balancing.

Before you configure an HTTP callout, you must set up an application on the server to which the callout will be sent. The application, which is called the *HTTP callout agent*, must be configured to respond to the HTTP callout request with the required information. The HTTP callout agent can also be a Web server that serves the data for which the NetScaler appliance sends the callout. You must make sure that the format of the response to an HTTP callout does not change from one invocation to another.

After you set up the HTTP callout agent, you configure the HTTP callout on the NetScaler appliance. Finally, to invoke the callout, you include the callout in a default syntax policy in the appropriate NetScaler feature and then bind the policy to the bind point at which you want the policy to be evaluated.

After you have configured the HTTP callout, you must verify the configuration to make sure that the callout is working correctly.

## How an HTTP Callout Works

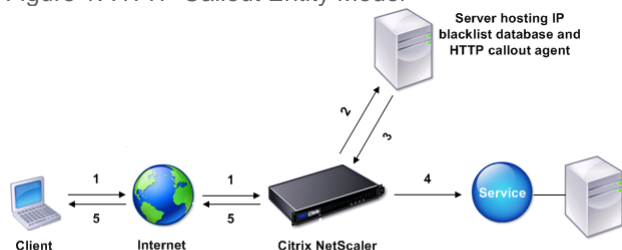
When the NetScaler appliance receives a client request, the appliance evaluates the request against the policies bound to various bind points. During this evaluation, if the appliance encounters the HTTP callout expression, `SYS.HTTP_CALLOUT(<name>)`, it stalls policy evaluation briefly and sends a request to the HTTP callout agent by using the parameters configured for the specified HTTP callout. Upon receiving the response, the appliance inspects the specified portion of the response, and then either performs an action or evaluates the next policy, depending on whether the evaluation of the response from the HTTP callout agent evaluates to TRUE or FALSE, respectively. For example, if the HTTP callout is included in a responder policy, if the evaluation of the response evaluates to TRUE, the appliance performs the action associated with the responder policy.

If the HTTP callout configuration is incorrect or incomplete, or if the callout invokes itself recursively, the appliance raises an UNDEF condition, and updates the undefined hits counter.

If the HTTP callout configuration is incorrect or incomplete, or if the callout invokes itself recursively, the appliance raises an UNDEF condition, and updates the undefined hits counter.

The following figure illustrates the working of an HTTP callout that is invoked from a globally bound responder policy. The HTTP callout is configured to include the IP address of the client that is associated with an incoming request. When the NetScaler appliance receives a request from a client, the appliance generates the callout request and sends it to the callout server, which hosts a database of blacklisted IP addresses and an HTTP callout agent that checks whether the client's IP address is listed in the database. The HTTP callout agent receives the callout request, checks whether the client's IP address is listed, and sends a response that the NetScaler appliance evaluates. If the response indicates that the client's IP address is not blacklisted, the appliance forwards the response to the configured service. If the client's IP address is blacklisted, the appliance resets the client connection

Figure 1. HTTP Callout Entity Model



- 1: Client request
- 2: HTTP callout request to check whether the client is blacklisted
- 3: Response from HTTP callout agent
- 4: Request forwarded to service if 3 indicates a safe IP address
- 5: Connection RESET if 3 indicates a bad IP address

## Notes on the Format of HTTP Requests and Responses

The NetScaler appliance does not check for the validity of the HTTP callout request. Therefore, before you configure HTTP callouts, you must know the format of an HTTP request. You must also know the format of an HTTP response, because configuring an HTTP callout involves configuring expressions that evaluate the response from the HTTP callout agent.

This document includes the following details:

- [Format of an HTTP Request](#)
- [Format of an HTTP Response](#)

### Format of an HTTP Request

An HTTP request contains a series of lines that each end with a carriage return and a line feed, represented as either <CR><LF> or \r\n.

The first line of a request (the *message line*) contains the HTTP method and target. For example, a message line for a GET request contains the keyword GET and a string that represents the object that is to be fetched, as shown in the following example:

```
GET /mysite/mydirectory/index.html HTTP/1.1\r\n
```

The rest of the request contains HTTP headers, including a required Host header and, if applicable, a message body.

The request ends with a blank line (an extra <CR><LF> or \r\n).

Following is an example of a request:

```
Get /mysite/index.html HTTP/1.1\r\n
Host: 10.101.101.10\r\n
Accept: */*\r\n
\r\n
```

### Format of an HTTP Response

An HTTP response contains a status message, response HTTP headers, and the requested object or, if the requested object cannot be served, an error message.

Following is an example of a response:

```
HTTP/1.1 200 OK\r\n
Content-Length: 55\r\n
Content-Type: text/html\r\n
Last-Modified: Wed, 12 Aug 1998 15:03:50 GMT\r\n
Accept-Ranges: bytes\r\n
ETag: "04f97692cbd1:377"\r\n
Date: Thu, 19 Jun 2008 19:29:07 GMT\r\n
\r\n
<55-character response>
```



## Configuring an HTTP Callout

When configuring an HTTP callout, you specify the type of request (HTTP or HTTPS), destination and format of the request, the expected format of the response, and, finally, the portion of the response that you want to analyze.

For the destination, you either specify the IP address and port of the HTTP callout agent or engage a load balancing, content switching, or cache redirection virtual server to manage the HTTP callout requests. In the first case, the HTTP callout requests will be sent directly to the HTTP callout agent. In the second case, the HTTP callout requests will be sent to the virtual IP address (VIP) of the specified virtual server. The virtual server will then process the request in the same way as it processes a client request. For example, if you expect a large number of callouts to be generated, you can configure instances of the HTTP callout agent on multiple servers, bind these instances (as services) to a load balancing virtual server, and then specify the load balancing virtual server in the HTTP callout configuration. The load balancing virtual server then balances the load on those configured instances as determined by the load balancing algorithm.

For the format of the HTTP callout request, you can specify the individual attributes of the HTTP callout request (an attribute-based HTTP callout), or you can specify the entire HTTP callout request as a default syntax expression (an expression-based HTTP callout).

Note: The appliance does not check for the validity of the request. You must make sure that the request is a valid request. An incorrect or incomplete HTTP callout configuration results in a runtime UNDEF condition that is not associated with an action. The UNDEF condition merely updates the Undefined Hits counter, which enables you to troubleshoot an incorrectly configured HTTP callout. However, the appliance parses the HTTP callout request to enable you to configure certain NetScaler features for the callout. This can lead to an HTTP callout invoking itself. For information about callout recursion and how you can avoid it, see ["Avoiding HTTP Callout Recursion."](#)

Finally, regardless of whether you use HTTP request attributes or an expression to define the format of the HTTP callout request, you must specify the format of the response from the HTTP callout agent and the portion of the response that you want to evaluate. The response can be a Boolean value, a number, or text. The portion of the response that you want to evaluate is specified by an expression. For example, if you specify that the response contains text, you can use `HTTP.RES.BODY(<unit>)` to specify that the appliance must evaluate only the first <unit> bytes of the response from the callout agent.

At the command line, you first create an HTTP callout by using the `add` command. When you add a callout, all parameters are set to a default value of NONE, except the HTTP method, which is set to a default value of GET. You then configure the callout's parameters by using the `set` command. The `set` command is used to configure both types of callouts (attribute-based and expression-based). The difference lies in the parameters that are used for configuring the two types of callouts. Accordingly, the command-line instructions that follow include a `set` command for configuring an attribute-based callout and a `set` command for configuring an expression-based callout. In the configuration utility, all of these configuration tasks are performed in a single dialog box.

Note: Before you put an HTTP callout into a policy, you can modify all configured parameters except the return type. Once an HTTP callout is in a policy, you cannot completely modify an expression that is configured in the callout. For example, you cannot change `HTTP.REQ.HEADER("myval")` to `CLIENT.IP.SRC`. However, you can modify the operators and arguments that are passed to the expression. For example, you can change `HTTP.REQ.HEADER("myVal1")` to `HTTP.REQ.HEADER("myVal2")`, or `HTTP.REQ.HEADER("myVal")` to `HTTP.REQ.HEADER("myVal").AFTER_STR(<string>)`. If the `set` command fails, create a new HTTP callout.

HTTP callout configuration involves configuring default syntax expressions. For more information about configuring default syntax expressions, see ["Configuring Default Syntax Expressions: Getting Started."](#)

## To configure an HTTP callout by using the command line interface

At the command prompt, do the following:

1. Create a HTTP callout.

```
add policy httpCallout <name>
```

### Example

```
> add policy httpCallout mycallout
```

2. Configure the details of the HTTP callout.

- o To configure an attribute-based HTTP callout, type:

```
set policy httpCallout <name> [-IPAddress <ip_addr|ipv6_addr|*>] [-port <port|*>] [-vServer <string>] [-returnType <returnType>] [-httpMethod ( GET | POST )] [-hostExpr <string>] [-urlStemExpr <string>] [-headers <name(value)> ...] [-parameters <name(value)> ...] [-resultExpr <string>]
```

### Example

```
> set policy httpCallout mycallout -vserver lbv1 -returnType num -httpMethod GET -urlStemExpr "http.req.url" -parameters Name("My Name") -headers Name("MyHeader") -resultExpr "http.res.body(10000).length"
```

- o To configure an expression-based HTTP callout, type:

```
set policy httpCallout <name> [-vServer <string>] [-returnType <returnType>] [-httpMethod ( GET | POST )] [-fullReqExpr <string>] [-resultExpr <string>]
```

### Example

```
> set policy httpCallout mycallout1 -vserver lbv1 -returnType num -httpMethod GET -fullReqExpr q{"GET " + http.req.url + "HTTP/" + http.req.version.major + " " + http.req.host + "\r\nHost: 10.101.10.10\r\nAccept: */*\r\n\r\n"}
```

3. Verify the configurations of the HTTP callout.

```
show policy httpCallout <name>
```




## To configure an HTTP callout by using the configuration utility

1. Navigate to AppExpert > HTTP Callouts.
2. In the details pane, click Add.
3. In the Create HTTP Callout dialog box, configure the parameters of the HTTP callout. For a description of the parameter, hover the mouse cursor over the check box.
4. Click Create and then click Close.

## Verifying the Configuration

For an HTTP callout to work correctly, all the HTTP callout parameters and the entities associated with the callout must be configured correctly. While the NetScaler appliance does not check the validity of the HTTP callout parameters, it indicates the state of the bound entities, namely the server or virtual server to which the HTTP callout is sent. The following table lists the icons and describes the conditions under which the icons are displayed.

Table 1. Icons That Indicate the States of Entities Bound to an HTTP Callout

Icon	Indicates that
	The state of the server that hosts the HTTP callout agent, or the load balancing, content switching, or cache redirection virtual server to which the HTTP callout is sent is UP.
	The state of the server that hosts the HTTP callout agent, or the load balancing, content switching, or cache redirection virtual server to which the HTTP callout is sent is OUT OF SERVICE.
	The state of the server that hosts the HTTP callout agent, or the load balancing, content switching, or cache redirection virtual server to which the HTTP callout is sent is DOWN.

For an HTTP callout to function correctly, the icon must be green at all times. If the icon is not green, check the state of the callout server or virtual server to which the HTTP callout is sent. If the HTTP callout is not working as expected even though the icon is green, check the parameters configured for the callout.

You can also verify the configuration by sending test requests that match the policy from which the HTTP callout is invoked, checking the hits counter for the policy and the HTTP callout, and verifying the responses that the NetScaler appliance sends to the client.

Note: An HTTP callout can sometimes invoke itself recursively a second time. If this happens, the hits counter is incremented by two counts for each callout that is generated by the appliance. For the hits counter to display the correct value, you must configure the HTTP callout in such a way that it does not invoke itself a second time. For more information about how you can avoid HTTP callout recursion, see ["Avoiding HTTP Callout Recursion."](#)

### To view the hits counter for an HTTP callout

1. Navigate to AppExpert > HTTP Callouts.
2. In the details pane, click the HTTP callout for which you want to view the hits counter, and then view the hits in the Details area.

## Invoking an HTTP Callout

After you configure an HTTP callout, you invoke the callout by including the `SYS.HTTP_CALLOUT(<name>)` expression in a default syntax policy rule. In this expression, `<name>` is the name of the HTTP callout that you want to invoke.

You can use default syntax expression operators with the callout expression to process the response and then perform an appropriate action. The return type of the response from the HTTP callout agent determines the set of operators that you can use on the response. If the part of the response that you want to analyze is text, you can use a text operator to analyze the response. For example, you can use the `CONTAINS(<string>)` operator to check whether the specified portion of the response contains a particular string, as in the following example:

```
SYS.HTTP_CALLOUT(myCallout).contains("Good IP address")
```

If you use the preceding expression in a responder policy, you can configure an appropriate responder action.

Similarly, if the part of the response that you want to evaluate is a number, you can use a numeric operator such as `GT(int)`. If the response contains a Boolean value, you can use a Boolean operator.

**Note:** An HTTP callout can invoke itself recursively. HTTP callout recursion can be avoided by combining the HTTP callout expression with a default syntax expression that prevents recursion. For information about how you can avoid HTTP callout recursion, see ["Avoiding HTTP Callout Recursion."](#)

You can also cascade HTTP callouts by configuring policies that each invoke a callout after evaluating previously generated callouts. In this scenario, after one policy invokes a callout, when the NetScaler appliance is parsing the callout before sending the callout to the callout server, a second set of policies can evaluate the callout and invoke additional callouts, which can in turn be evaluated by a third set of policies, and so on. Such an implementation is described in the following example.

First, you could configure an HTTP callout called `myCallout1`, and then configure a responder policy, `Pol1`, to invoke `myCallout1`. Then, you could configure a second HTTP callout, `myCallout2`, and a responder policy, `Pol2`. You configure `Pol2` to evaluate `myCallout1` and invoke `myCallout2`. You bind both responder policies globally.

To avoid HTTP callout recursion, `myCallout1` is configured with a unique custom HTTP header called `"Request1."` `Pol1` is configured to avoid HTTP callout recursion by using the default syntax expression,

```
HTTP.REQ.HEADER("\Request1\").EQ("\Callout Request\").NOT.
```

`Pol2` uses the same default syntax expression, but excludes the `.NOT` operator so that the policy evaluates `myCallout1` when the NetScaler appliance is parsing it. Note that `myCallout2` identifies its own unique header called `"Request2,"` and `Pol2` includes a default syntax expression to prevent `myCallout2` from invoking itself recursively.

### Example

```
> add policy httpCallout myCallout1
```

Done

```
> set policy httpCallout myCallout1 -IPAddress 10.102.3.95 -port 80 -returnType TEXT -hostEx  
"\10.102.3.95\" -urlStemExpr "\/cgi-bin/check_clnt_from_database.pl\" -headers Request1  
("Callout Request") -parameters cip(CLIENT.IP.SRC) -resultExpr "HTTP.RES.BODY(100)"
```

Done

```
> add responder policy Pol1 "HTTP.REQ.HEADER("\Request1\").EQ("\Callout Request\").NOT &&  
SYS.HTTP_CALLOUT(myCallout1).CONTAINS("\IP Matched\")" RESET
```

Done

```
> bind responder global Pol1 100 END -type OVERRIDE
```

Done

```
> add policy httpCallout myCallout2
```

Done

```
> set policy httpCallout myCallout2 -IPAddress 10.102.3.96 -port 80 -returnType TEXT -hostEx  
"\10.102.3.96\" -urlStemExpr "\/cgi-bin/check_clnt_location_from_database.pl\" -headers  
("Callout Request") -parameters cip(CLIENT.IP.SRC) -resultExpr "HTTP.RES.BODY(200)"
```

Done

```
> add responder policy Pol2 "HTTP.REQ.HEADER(\"Request2\").EQ(\"Callout Request\").NOT &&  
HTTP.REQ.HEADER(\"Request1\").EQ(\"Callout Request\") && SYS.HTTP_CALLOUT(myCallout2).CONTA  
(\"APAC\")" RESET
```

Done

```
> bind responder global Pol2 110 END -type OVERRIDE
```

Done

## Avoiding HTTP Callout Recursion

Even though the NetScaler appliance does not check for the validity of the HTTP callout request, it parses the request once before it sends the request to the HTTP callout agent. This parsing allows the appliance to treat the callout request as any other incoming request, which in turn allows you to configure several useful NetScaler features (such as integrated caching, SureConnect, and Priority Queuing) to work on the callout request.

However, during this parsing, the HTTP callout request can hit the same policy and therefore invoke itself recursively. The appliance detects the recursive invocation and raises an undefined (UNDEF) condition. However, the recursive invocation results in the policy and HTTP callout hit counters being incremented by two counts each instead of one count each.

To prevent a callout from invoking itself, you must identify at least one unique characteristic of the HTTP callout request, and then exclude all requests with this characteristic from being processed by the policy rule that invokes the callout. You can do so by including another default syntax expression in the policy rule. The expression must precede the `SYS.HTTP_CALLOUT (<name>)` expression so that it is evaluated before the callout expression is evaluated. For example:

```
<Expression that prevents callout recursion> && SYS.HTTP_CALLOUT(<name>)
```

When you configure a policy rule in this way, when the appliance generates the request and parses it, the compound rule evaluates to FALSE, the callout is not generated a second time, and the hit counters are incremented correctly.

One way by which you can assign a unique characteristic to an HTTP callout request is to include a unique custom HTTP header when you configure the callout. Following is an example of an HTTP callout called "myCallout." The callout generates an HTTP request that checks whether a client's IP address is present in a database of blacklisted IP addresses. The callout includes a custom header called "Request," which is set to the value "Callout Request." A globally bound responder policy, "Pol1," invokes the HTTP callout but excludes all requests whose Request header is set to this value, thus preventing a second invocation of myCallout. The expression that prevents a second invocation is `HTTP.REQ.HEADER(\"Request\").EQ(\"Callout Request\").NOT`.

### Example

```
> add policy httpCallout myCallout
Done

> set policy httpCallout myCallout -IPAddress 10.102.3.95 -port 80 -returnType TEXT -hostExp
Done

> add responder policy Pol1 "HTTP.REQ.HEADER(\"Request\").EQ(\"Callout Request\").NOT && SYS
Done

> bind responder global Pol1 100 END -type OVERRIDE
Done
```

Note: You can also configure an expression to check whether the URL of the request includes the URL stem expression that is configured for the HTTP callout. If you want to implement this scenario, make sure that the HTTP callout agent is dedicated to respond only to HTTP callouts and not to other client requests directed through the appliance. If the HTTP callout agent is an application or Web server that serves other client requests, such an expression will prevent the appliance from processing those client requests. Instead, use a unique custom header as described earlier.

## Caching HTTP Callout Responses

For improved performance while using callouts, you can use the integrated caching feature to cache callout responses. The responses are stored in an integrated caching content group named `calloutContentGroup` for a specified time duration.

Note: To cache callout responses, make sure that the integrated caching feature is enabled.

### To set the cache duration by using the command line interface

At the command prompt, type:

```
set policy httpCallout <name> -cacheForSecs <secs>
```

#### Example

```
> set httpcallout httpcallout1 -cacheForSecs 120
```

### To set the cache duration by using the configuration utility

1. Navigate to AppExpert > HTTP Callouts.
2. In the details pane, select the HTTP callout for which you want to set the cache duration and click Open.
3. In the Configure HTTP Callout dialog box, specify the Cache Expiration Time.
4. Verify that you have entered the correct time duration, and then click OK.

## Use Case: Filtering Clients by Using an IP Blacklist

HTTP callouts can be used to block requests from clients that are blacklisted by the administrator. The list of clients can be a publicly known blacklist, a blacklist that you maintain for your organization, or a combination of both.

The NetScaler appliance checks the IP address of the client against the pre-configured blacklist and blocks the transaction if the IP address has been blacklisted. If the IP address is not in the list, the appliance processes the transaction.

To implement this configuration, you must perform the following tasks:

1. Enable responder on the NetScaler appliance.
2. Create an HTTP callout on the NetScaler appliance and configure it with details about the external server and other required parameters.
3. Configure a responder policy to analyze the response to the HTTP callout, and then bind the policy globally.
4. Create an HTTP callout agent on the remote server.

## Enabling Responder

Updated: 2013-08-30

You must enable responder before you can use it.

### To enable responder by using the configuration utility

1. Make sure that you have installed the responder license.
2. In the configuration utility, expand AppExpert, and right-click Responder, and then click Enable Responder feature.

## Creating an HTTP Callout on the NetScaler Appliance

Updated: 2013-08-30

Create an HTTP callout, HTTP-Callout-1, with the parameter settings shown in the following table. For more information about creating an HTTP callout, see ["Configuring an HTTP Callout."](#)

Table 1. Parameters and Values for HTTP-Callout-1

Parameter	Value
Name	HTTP-Callout-1
<b>Server to receive callout request</b>	
IP Address	10.103.9.95
Port	80
<b>Request to send to the server</b>	
Method	GET
Host Expression	10.102.3.95
URL Stem Expression	"/cgi-bin/check_clnt_from_database.pl"
<b>Headers</b>	
Name	Request
Value-expression	Callout Request
<b>Parameters</b>	
Name	Cip
Value-expression	CLIENT.IP.SRC
<b>Server Response</b>	
Return Type	TEXT
Expression to extract data from the response	HTTP.RES.BODY(100)

## Configuring a Responder Policy and Binding it Globally



Updated: 2013-08-30

After you configure the HTTP callout, verify the callout configuration, and then configure a responder policy to invoke the callout. While you can create a responder policy in the Policies sub-node and then bind it globally by using the Responder Policy Manager, this demonstration uses the Responder Policy Manager to create the responder policy and bind the policy globally.

## To create a responder policy and bind it globally by using the configuration utility

1. Navigate to AppExpert > Responder.
2. In the details pane, under Policy Manager, click Policy Manager.
3. In the Responder Policy Manager dialog box, click Override Global.
4. Click Insert Policy, and then, under Policy Name, click New Policy.
5. In the Create Responder Policy dialog box, do the following:
  - a. In Name, type Policy-Responder-1.
  - b. In Action, select RESET.
  - c. In Undefined-Result Action, select Global undefined-result action.
  - d. In Expression, type the following default syntax expression:

```
"HTTP.REQ.HEADER(\"Request\").EQ(\"Callout Request\").NOT && SYS.HTTP_CALLOUT(HTTP
```

- e. Click Create, and then click Close.
6. Click Apply Changes, and then click Close.

## Creating an HTTP Callout Agent on the Remote Server

You must now create an HTTP callout agent on the remote callout server that will receive callout requests from the NetScaler appliance and respond appropriately. The HTTP callout agent is a script that is different for each deployment and must be written with the server specifications in mind, such as the type of database and the scripting language supported.

Following is a sample callout agent that verifies whether the given IP address is part of an IP blacklist. The agent has been written in the Perl scripting language and uses a MYSQL database.

The following CGI script checks for a given IP address on the callout server.

```
#!/usr/bin/perl -w
print "Content-type: text/html\n\n";
    use DBI();
    use CGI qw(:standard);
#Take the Client IP address from the request query
    my $ip_to_check = param('cip');
# Where a MYSQL database is running
    my $dsn = 'DBI:mysql:BAD_CLIENT:localhost';
# Database username to connect with
    my $db_user_name = 'dbuser';
# Database password to connect with
    my $db_password = 'dbpassword';
    my ($id, $password);
# Connecting to the database
    my $dbh = DBI->connect($dsn, $db_user_name, $db_password);
    my $sth = $dbh->prepare(qq{ select * from bad_clnt });
    $sth->execute();
    while (my ($ip_in_database) = $sth->fetchrow_array()) {
        chomp($ip_in_database);
# Check for IP match
        if ($ip_in_database eq $ip_to_check) {
            print "\n IP Matched\n";
        }
    }
    print "\n IP Failed\n";
    $sth->finish();
    exit;
```

e

## Use Case: ESI Support for Fetching and Updating Content Dynamically

Edge Side Includes (ESI) is a markup language for edge-level dynamic Web content assembly. It helps in accelerating dynamic Web-based applications by defining a simple markup language to describe cacheable and non-cacheable Web page components that can be aggregated, assembled, and delivered at the network edge. By using HTTP callouts on the NetScaler appliance, you can read through the ESI constructs and aggregate or assemble content dynamically.

To implement this configuration, you must perform the following tasks:

1. Enable rewrite on the NetScaler appliance.
2. Create an HTTP callout on the appliance and configure it with details about the external server and other required parameters.
3. Configure a rewrite action to replace the ESI content with the callout response body.
4. Configure a rewrite policy to specify the conditions under which the action is performed, and then bind the rewrite policy globally.

### Enabling Rewrite

Updated: 2013-08-30

Rewrite must be enabled before it is used on the NetScaler appliance. The following procedure describes the steps to enable the rewrite feature.

#### To enable rewrite by using the configuration utility

1. Make sure that you have installed the rewrite license.
2. In the configuration utility, expand AppExpert, and right-click Rewrite, and then click Enable Rewrite feature.

## Creating an HTTP Callout on the NetScaler Appliance

Updated: 2013-08-30

Create an HTTP callout, HTTP-Callout-2, with the parameter settings shown in the following table. For more information about creating an HTTP callout, see ["Configuring an HTTP Callout."](#)

Table 1. Parameters and Values for HTTP-Callout-2

Parameter	Value
Name	HTTP-Callout-2
<b>Server to receive callout request</b>	
IP Address	10.102.56.51
Port	80
<b>Request to send to the server</b>	
Method	GET
Host Expression	10.102.56.51:80
URL Stem Expression	"HTTP.RES.BODY(500).AFTER_STR(\\\"src=\\\").BEFORE_STR(\\\"/>\\\")"
<b>Headers</b>	
Name	Name
Value-expression	Callout
<b>Server Response</b>	
Return Type	TEXT
Expression to extract data from the response	HTTP.RES.BODY(100)

### Configuring the Rewrite Action

Updated: 2013-08-30

Create a rewrite action, Action-Rewrite-1, to replace the ESI content with the callout response body. Use the parameter settings shown in the following table.

Table 2. Parameters and Values for Action-Rewrite-1

Parameter	Value
Name	Action-Rewrite-1
Type	Replace
Expression to choose target text reference	"HTTP.RES.BODY(500).AFTER_STR (\ " <example> \ ").BEFORE_STR (\ "</example> \ ") "
String expression for replacement text	"SYS.HTTP_CALLOUT(HTTP-Callout-2)"

## To configure the rewrite action by using the configuration utility

1. Navigate to AppExpert > Rewrite > Actions.
2. In the details pane, click Add.
3. In the Create Rewrite Action dialog box, in Name, type Action-Rewrite-1.
4. In Type, select REPLACE.
5. In Expression to choose target text reference, type the following default syntax expression:

```
"HTTP.RES.BODY( 500 ) .AFTER_STR( \ " <example> \ " ) .BEFORE_STR( \ "</example> \ " ) "
```

6. In the String expression for replacement text, type the following string expression:

```
"SYS.HTTP_CALLOUT( HTTP-Callout-2 ) "
```

7. Click Create, and then click Close.

## Creating the Rewrite Policy and Binding it Globally

Updated: 2013-08-30

Create a rewrite policy, Policy-Rewrite-1, with the parameter settings shown in the following table. You can create a rewrite policy in the Policies subnode and then bind it globally by using the Rewrite Policy Manager. Alternatively, you can use the Rewrite Policy Manager to perform both these tasks simultaneously. This demonstration uses the Rewrite Policy Manager to perform both tasks.

Table 3. Parameters and Values for Policy-Rewrite-1

Parameter	Value
Name	Policy-Rewrite-1
Action	Action_Rewrite-1
Undefined Result Action	-Global undefined-result action-
Expression	"HTTP.REQ.HEADER(\ "Name\ ").CONTAINS ( \ "Callout\ " ).NOT"

## To configure a rewrite policy and bind it globally by using the configuration utility

1. Navigate to AppExpert > Rewrite.
2. In the details pane, under Policy Manager, click Rewrite Policy Manager.
3. In the Rewrite Policy Manager dialog box, click Override Global.
4. Click Insert Policy, and then, in the Policy Name column, click New Policy.
5. In the Create Rewrite Policy dialog box, do the following:
  - a. In Name, type Policy-Rewrite-1.
  - b. In Action, select Action-Rewrite-1.
  - c. In Undefined-Result Action, select Global undefined-result action.
  - d. In Expression, type the following default syntax expression:

```
"HTTP.REQ.HEADER( \ "Name\ " ) .CONTAINS( \ "Callout\ " ) .NOT"
```

- e. Click Create, and then click Close.
6. Click Apply Changes, and then click Close.

## Use Case: Access Control and Authentication

In high security zones, it is mandatory to externally authenticate the user before a resource is accessed by clients. On the NetScaler appliance, you can use HTTP callouts to externally authenticate the user by evaluating the credentials supplied. In this example, the assumption is that the client is sending the user name and password through HTTP headers in the request. However, the same information could be fetched from the URL or the HTTP body.

To implement this configuration, you must perform the following tasks:

1. Enable the responder feature on the NetScaler appliance.
2. Create an HTTP callout on the appliance and configure it with details about the external server and other required parameters.
3. Configure a responder policy to analyze the response, and then bind the policy globally.
4. Create a callout agent on the remote server.

## Enabling Responder

Updated: 2013-08-30

The responder feature must be enabled before it is used on the NetScaler appliance.

### To enable responder by using the configuration utility

1. Make sure that the responder license is installed.
2. In the configuration utility, expand AppExpert, and right-click Responder, and then click Enable Responder feature.

## Creating an HTTP Callout on the NetScaler Appliance

Updated: 2013-08-30

Create an HTTP callout, HTTP-Callout-3, with the parameter settings shown in the following table. For more information about creating an HTTP callout, see "[Configuring an HTTP Callout](#)."

Table 1. Parameters and Values for HTTP-Callout-3

Parameter	Value
Name	HTTP-Callout-3
<b>Server to receive callout request</b>	
IP Address	10.103.9.95
Port	80
<b>Request to send to the server</b>	
Method	GET
Host Expression	10.102.3.95
URL Stem Expression	âœœcgi-bin/authenticate.plâœ•
<b>Headers</b>	
Name	Request
Value-expression	Callout Request
<b>Parameters</b>	
Name	Username
Value-expression	HTTP.REQ.HEADER("Username").VALUE(0)
Name	Password
Value-expression	HTTP.REQ.HEADER("Password").VALUE(0)
<b>Server Response</b>	
Return Type	TEXT
Expression to extract data from the response	HTTP.RES.BODY(100)

## Creating a Responder Policy to Analyze the Response

Updated: 2013-08-30

Create a responder policy, Policy-Responder-3, that will check the response from the callout server and RESET the connection if the source IP address has been blacklisted. Create the policy with the parameters settings shown in the following table. While you can create a responder policy in the Policies subnode and then bind it globally by using the Responder Policy Manager, this demonstration uses the Responder Policy Manager to create the responder policy and bind the policy globally.

Table 2. Parameters and Values for Policy-Responder-3

Parameter	Value
Name	Policy-Responder-3
Action	RESET
Undefined-Result-Action	-Global undefined-result action-
Expression	"HTTP.REQ.HEADER("Request").EQ("Callout Request").NOT && SYS.HTTP_CALLOUT(HTTP-Callout-3).CONTAINS("Authentication Failed")"

### To create a responder policy and bind it globally by using the configuration utility

1. Navigate to AppExpert > Responder.
2. In the details pane, under Policy Manager, click Responder Policy Manager.
3. In the Responder Policy Manager dialog box, click Override Global.
4. Click Insert Policy, and then, in the Policy Name column, click New Policy.
5. In the Create Responder Policy dialog box, do the following:
  - a. In Name, type Policy-Responder-3.
  - b. In Action, select RESET.
  - c. In Undefined-Result Action, select Global undefined-result action.
  - d. In the Expression text box, type:

```
"HTTP.REQ.HEADER("Request").EQ("Callout Request").NOT && SYS.HTTP_CALLOUT(HTTP-Callout-3).CONTAINS("Authentication Failed")"
```

- e. Click Create, and then click Close.
6. Click Apply Changes, and then click Close.

## Creating an HTTP Callout Agent on the Remote Server

You now need to create an HTTP callout agent on the remote callout server. The HTTP callout agent receives callout requests from the NetScaler appliance and responds appropriately. The callout agent is a script that is different for each deployment and must be written with server specifications in mind, such as the type of database and the scripting language supported.

Following is sample callout agent pseudo-code that verifies whether the supplied user name and password are valid. The agent can be implemented in any programming language of your choice. The pseudo-code is to be used only as a guideline for developing the callout agent. You can build additional functionality into the program.

### To verify the supplied user name and password by using pseudo-code

1. Accept the user name and password supplied in the request and format them appropriately.
2. Connect to the database that contains all the valid user names and passwords.
3. Check the supplied credentials against your database.
4. Format the response as required by the HTTP callout.
5. Send the response to the NetScaler appliance.

## Use Case: OWA-Based Spam Filtering

Spam filtering is the ability to dynamically block emails that are not from a known or trusted source or that have inappropriate content. Spam filtering requires an associated business logic that indicates that a particular kind of message is spam. When the NetScaler appliance processes Outlook Web Access (OWA) messages based on the HTTP protocol, HTTP callouts can be used to filter spam.

You can use HTTP callouts to extract any portion of the incoming message and check with an external callout server that has been configured with rules that are meant for determining whether a message is legitimate or spam. In case of spam email, for security reasons, the NetScaler appliance does not notify the sender that the email is marked as spam.

The following example conducts a very basic check for various listed keywords in the email subject. These checks can be more complex in a production environment.

To implement this configuration, you must perform the following tasks:

1. Enable the responder feature on the NetScaler appliance.
2. Create an HTTP callout on the NetScaler appliance and configure it with details about the external server and other required parameters.
3. Create a responder policy to analyze the response, and then bind the policy globally.
4. Create a callout agent on the remote server.

## Enabling Responder

Updated: 2013-08-30

The responder feature must be enabled before it can be used on the NetScaler appliance.

### To enable responder by using the configuration utility

1. Make sure that the responder license is installed.
2. In the configuration utility, expand AppExpert, and right-click Responder, and then click Enable Responder feature.

## Creating an HTTP Callout on the NetScaler Appliance

Updated: 2013-08-30

Create an HTTP callout, HTTP-Callout-4, with the parameter settings shown in the following table. For more information about creating an HTTP callout, see ["Configuring an HTTP Callout."](#)

Table 1. Parameters and Values for HTTP-Callout-4

Parameter	Value
Name	HTTP-Callout-4
<b>Server to receive callout request</b>	
IP Address	10.103.56.51
Port	80
<b>Request to send to the server</b>	
Method	POST
Host Expression	ffffff
URL Stem Expression	"/cgi-bin/Callout/spam_filter.pl"
<b>Headers</b>	
Name	Request
Value-expression	Callout Request
<b>Parameters</b>	
Name	Subject
Value-expression	("\" + HTTP.REQ.BODY(1000).AFTER_STR("urn:schemas:httpmail:subject=").BEFORE_STR("\n").TO_LOWER + "\"")
<b>Server Response</b>	

Return Type	BOOL
Expression to extract data from the response	HTTP.RES.BODY(100) .CONTAINS(\"Matched\")

## Creating a Responder Action

Updated: 2013-08-30

Create a responder action, Action-Responder-4. Create the action with the parameter settings shown in the following table.

Table 2. Parameters and Values for Action-Responder-4

Parameter	Value
Name	Action-Responder-4
Type	Respond with
Target	"\"HTTP/1.1 200 OK\r\nServer: Microsoft-IIS/6.0\r\nX-Powered-By: ASP.NET\r\nContent-Length: 0\r\nMS-WebStorage: 6.5.6944\r\nCache-Control: no-cache\r\n\r\n\""

### To create a responder action by using the configuration utility

1. Navigate to AppExpert > Responder > Actions.
2. In the details pane, click Add.
3. In the Create Responder Action dialog box, in Name, type Action-Responder-4.
4. In Type, click Respond with.
5. In Target, type:

```
"\"HTTP/1.1 200 OK\r\nServer: Microsoft-IIS/6.0\r\nX-Powered-By: ASP.NET\r\nContent-Ler
```

6. Click Create, and then click Close.

## Creating a Responder Policy to Invoke the HTTP Callout

Updated: 2013-08-30

Create a responder policy, Policy-Responder-4, that will check the request body and, if the body contains the word *subject*, invoke the HTTP callout to verify the email. Create the policy with the parameter settings shown in the following table. While you can create a responder policy in the Policies subnode and then bind it globally by using the Responder Policy Manager, this demonstration uses the Responder Policy Manager to create the responder policy and bind it globally.

Table 3. Parameters and Values for Policy-Responder-4

Parameter	Value
Name	Policy-Responder-4
Action	Action-Responder-4
Undefined-Result-Action	-Global undefined-result action-
Expression	"HTTP.REQ.BODY(1000).CONTAINS(\"urn:schemas:httpmail:subject\") && SYS.HTTP_CALLOUT(HTTP-Callout-4)"

### To create a responder policy by using the configuration utility

1. Navigate to AppExpert > Responder.
2. In the details pane, under Policy Manager, click Responder policy manager.
3. In the Responder Policy Manger dialog box, click Override Global.
4. Click Insert Policy, and then, in the Policy Name column, click New Policy.
5. In the Create Responder Policy dialog box, do the following:
  - a. In Name, type Policy-Responder-4.
  - b. In Action, click Action-Responder-4.
  - c. In Undefined-Result Action, click Global undefined-result action.
  - d. In the Expression text box, type:

```
"HTTP.REQ.BODY(1000).CONTAINS(\"urn:schemas:httpmail:subject\") && SYS.HTTP_CALLOI
```

- e. Click Create, and then click Close.
6. Click Apply Changes, and then click Close.

## Creating an HTTP Callout Agent on the Remote Server

You will now need to create an HTTP callout agent on the remote callout server. The HTTP callout agent receives callout requests from the NetScaler appliance and responds accordingly. The callout agent is a script that is different for each deployment and must be written with server specifications in mind, such as the type of database and the scripting language supported.

The following pseudo-code provides instructions for creating a callout agent that checks a list of words that are generally understood to indicate spam mails. The agent can be implemented in any programming language of your choice. The pseudo-code is to be used only as a guideline for developing the callout agent. You can build additional functionality into the program.

### To identify spam email by using pseudo-code

1. Accept the email subject provided by the NetScaler appliance.
2. Connect to the database that contains all the terms against which the email subject is checked.
3. Check the words in the email subject against the spam word list.
4. Format the response as required by the HTTP callout.
5. Send the response to the NetScaler appliance.



## Use Case: Dynamic Content Switching

This use case provides dynamic content switching by using an HTTP callout to get the name of the load balancing virtual server to which the request is forwarded.

1. Add a content switching virtual server.

```
> add cs vserver cs_vserver1 HTTP 10.102.29.196 80
```

2. Create an HTTP callout.

```
> add policy httpCallout http_callout1
```

3. Configure the HTTP callout to respond with the name of the load balancing virtual server from a request that contains the client IP address in the HTTP header "X-CLIENT-IP".

```
> set policy httpCallout http_callout1 -IPAddress 10.217.14.23 -port 80 -returnType TEXT
```

4. Configure the content switching action to retrieve the callout response.

```
> add cs action cs_action1 -targetVserverExpr 'SYS.HTTP_CALLOUT(http_callout1)'
```

Note: You must bind a load balancing virtual server to the content switching virtual server to account for:

- The non-availability of the load balancing virtual server that the callout resolves to.
- A UNDEF condition that results from the execution of the callout.

```
> bind cs vserver cs_vserver1 -lbvserver default_lbvip
```

5. Configure the content switching policy.

```
> add cs policy cs_policy1 -rule true -action cs_action1
```

6. Binding the content switching policy to the content switching virtual server.

```
> bind cs vserver cs_vserver1 -policyName cs_policy1 -priority 10
```

## Pattern Sets and Data Sets

Policy expressions for string matching operations on a large set of string patterns tend to become long and complex. Resources consumed by the evaluation of such complex expressions are significant in terms of processing cycles, memory, and configuration size. You can create simpler, less resource-intensive expressions by using pattern matching. Depending on the type of patterns that you want to match, you can use one of the following features to implement pattern matching:

- A pattern set is an array of indexed patterns used for string matching during default syntax policy evaluation. Example of a pattern set: `imagetypes {svg, bmp, png, gif, tiff, jpg}`.
- A data set is a specialized form of pattern set. It is an array of patterns of types number (integer), IPv4 address, or IPv6 address.

In many cases, you can use either pattern sets or data sets. However, in cases where you want specific matches for numerical data or IPv4 and IPv6 addresses, you must use data sets.

Note: Pattern sets and data sets can be used only in default syntax policies.

To use pattern sets or data sets, first create the pattern set or data set and bind patterns to it. Then, when you configure a policy for comparing a string in a packet, use an appropriate operator and pass the name of the pattern set or data set as an argument.

## How String Matching works with Pattern Sets and Data Sets

A pattern set or data set contains a set of patterns, and each pattern is assigned a unique index. When a policy is applied to a packet, an expression identifies a string to be evaluated, and the operator compares the string to the patterns defined in the pattern set or data set until a match is found or all patterns have been compared. Then, depending on its function, the operator returns either a boolean value that indicates whether or not a matching pattern was found or the index of the pattern that matches the string.

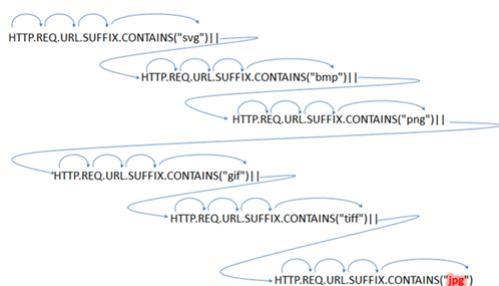
Note: This topic explains the working of a pattern set. Data sets work the same way. The only difference between pattern sets and data sets is the type of patterns defined in the set.

Consider the following use case to understand how patterns can be used for string matching.

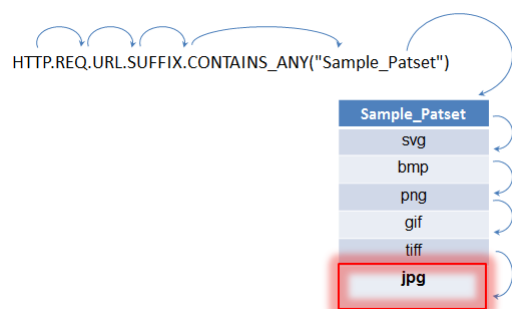
You want to determine whether the URL suffix (target text) contains any of the image file extensions. Without using pattern sets, you would have to define a complex expression, as follows:

```
HTTP.REQ.URL.SUFFIX.CONTAINS("svg") &|&| HTTP.REQ.URL.SUFFIX.CONTAINS("bmp") &|&| HTTP.REQ.U  
HTTP.REQ.URL.SUFFIX.CONTAINS("gif") &|&| HTTP.REQ.URL.SUFFIX.CONTAINS("tiff") &|&| HTTP.REQ.
```

If the URL has a suffix of "jpg," with the above compound expression, the NetScaler appliance has to iterate through the entire compound expression sequentially, from one sub expression to the next, to determine that the request refers to a jpg image. The following figure shows the steps in the process.



When a compound expression includes hundreds of sub expressions, the above process is resource intensive. A better alternative is an expression that invokes a pattern set, as shown in the following figure.



During policy evaluation as shown above, the operator (CONTAINS\_ANY) compares the string identified in the request with the patterns defined in the pattern set until a match is found. With the `Sample_Patset` expression, the multiple iterations through six sub expressions are reduced to just one.

By eliminating the need to configure compound expressions that perform string matching with multiple OR operations, pattern sets or data sets simplify configuration and accelerate processing of requests and responses.

## Configuring a Pattern Set

To configure a pattern set, you must specify the strings that are to serve as patterns. You can manually assign a unique index value to each of these patterns, or you can allow the index values to be assigned automatically.

Note: Pattern sets are case sensitive (unless you specify the expression to ignore case). Therefore, the string pattern "product1," for example, is not the same as the string pattern "Product1."

### Points to remember about index values

- You cannot bind the same index value to more than one pattern.
- An automatically assigned index value is one number larger than the highest index value of the existing patterns within the pattern set. For example, if the highest index value of existing patterns in a pattern set is 104, the next automatically assigned index value will be 105.
- If you do not specify an index for the first pattern, index value 1 is automatically assigned to that pattern.
- Index values are not regenerated automatically if one or more patterns are deleted or modified. For example, if the set contains five patterns, with indexes from 1 through 5, and if the pattern with an index of 3 is deleted, the other index values in the pattern set are not automatically regenerated to produce values from 1 through 4.
- The maximum index value that can be assigned to a pattern is 4294967290. If that value is already assigned to a pattern in the set, you must manually assign index values to any newly added patterns. An unused index value that is lower than a currently used value cannot be assigned automatically.

## To configure a pattern set by using the command line interface

At the command prompt, do the following:

1. Create a pattern set.

```
add policy patset <name>
```

#### Example:

```
> add policy patset samplepatset
```

2. Bind patterns to the pattern set.

```
bind policy patset <name> <string> [-index <positive_integer>]
```

#### Example:

```
> bind policy patset samplepatset product1 -index 1
```

Note: Repeat this step for all the patterns you want to bind to the pattern set.

3. Verify the configuration.

```
show policy patset <name>
```

## To configure a pattern set by using the configuration utility

1. Navigate to AppExpert > Pattern Sets.
2. In the details pane, click Add to open the Create Pattern Set dialog box.
3. Specify a name for the pattern set in the Name text box.
4. Under Specify Pattern, type the first pattern and, optionally, specify values for the following parameters:
  - Treat back slash as escape character—Select this check box to specify that any backslash characters that you might include in the pattern are to be treated as escape characters.
  - Index—A user assigned index value, from 1 through 4294967290.
5. Verify that you have entered the correct characters, and then click Add.
6. Repeat steps 4 and 5 to add additional patterns, and then click Create.

## Configuring a Data Set

To configure a data set, you must specify the strings that are to serve as patterns, and assign a type (number, IPv4 address, or IPv6 address) to each pattern. You can manually assign a unique index value to each of these patterns, or you can allow the index values to be assigned automatically.

Note: Data sets are case sensitive (unless you specify the expression to ignore case). Therefore, the string pattern "product1," for example, is not the same as the string pattern "Product1."

The rules applied for index values of data sets are the same as those applied for pattern sets. For information about index values, see "[Configuring a Pattern Set](#)."

## To configure a data set by using the command line interface

At the command prompt, do the following:

1. Create a data set.

```
add policy dataset <name> <type>
```

**Example:**

```
> add policy dataset sampledataset ipv4
```

2. Bind patterns to the data set.

```
bind policy dataset <name> <value> [-index <positive_integer>]
```

**Example:**

```
> bind policy dataset sampledataset 10.102.29.1 -index 1
```

Note: Repeat this step for all the patterns you want to bind to the data set.

3. Verify the configuration.

```
show policy dataset <name>
```

## To configure a data set by using the configuration utility

Navigate to AppExpert > Data Sets, click Add and specify the relevant details.

## Using Pattern Sets and Data Sets

Default syntax policy expressions that take pattern sets or data sets as an argument can be used to perform string matching operations.

The usage is as follows:

```
<text>.<operator>(" <name> ")
```

where,

- <text> is the expression that identifies a string in a packet. Example: `HTTP.REQ.HEADER("Host")`.
- <operator> is one of the operators described in the following table.

Table 1. Operators for pattern sets and data sets

Operator	Description
<code>&lt;text&gt;.CONTAINS_ANY(&lt;name&gt;)</code>	Returns true if the target text contains one or more of the patterns defined in the specified pattern set or data set.
<code>&lt;text&gt;.SUBSTR_ANY(&lt;name&gt;)</code>	Returns the first string that matches any pattern defined in the specified pattern set or data set.
<code>&lt;text&gt;.BEFORE_STR_ANY(&lt;name&gt;)</code>	Returns the text that is present before the first occurrence of any of the patterns defined in the specified pattern set or data set.
<code>&lt;text&gt;.AFTER_STR_ANY(&lt;name&gt;)</code>	Returns the text that is present after the first occurrence of any of the patterns defined in the specified pattern set or data set.
<code>&lt;text&gt;.EQUALS_ANY (&lt;name&gt;)</code>	Returns true if the target text exactly matches any of the patterns defined in the specified pattern set or data set.
<code>&lt;text&gt;.ENDSWITH_ANY(&lt;name&gt;)</code>	Returns true if the target text ends with any of the patterns that are defined in the specified pattern set or data set.
<code>&lt;text&gt;.STARTSWITH_ANY(&lt;name&gt;)</code>	Returns true if the target text starts with any of the patterns that are defined in the specified pattern set or data set.
<code>&lt;text&gt;.STARTSWITH_INDEX(&lt;name&gt;)</code>	Evaluates whether the target text starts with any of the patterns that are defined in the specified pattern set or data set. If a match is found, the index of the matching pattern is returned. Otherwise, 0 is returned.
<code>&lt;text&gt;.ENDSWITH_INDEX(&lt;name&gt;)</code>	Evaluates whether the target text ends with any of the patterns that are defined in the specified pattern set or data set. If a match is found, the index of the matching pattern is returned. Otherwise, 0 is returned.
<code>&lt;text&gt;.CONTAINS_INDEX(&lt;name&gt;)</code>	Evaluates whether the target text contains any of the patterns that are defined in the specified pattern set or data set. If a match is found, the index of the matching pattern is returned. Otherwise, 0 is returned.
<code>&lt;text&gt;.EQUALS_INDEX(&lt;name&gt;)</code>	Evaluates whether the target text exactly matches any of the patterns that are defined in the specified pattern set or data set. If an exact match is found, the index of the pattern is returned. Otherwise, 0 is returned.

- <name> is the name of the pattern set or data set

For sample usage, see ["Sample Usage."](#)

## Sample Usage

To understand the usage of pattern sets in expressions, consider the example of a pattern set named "imagetypes."

Table 1. Pattern set "imagetypes"

Patterns	Index value
svg	1
bmp	2
png	3
gif	4
tiff	5
jpg	6

**Example 1:** Determine whether the suffix of an HTTP request is one of the file extensions defined in the "imagetypes" pattern set.

- **Expression.** `HTTP.REQ.URL.SUFFIX.EQUALS_ANY("imagetypes")`
- **Sample URL.** `http://www.example.com/homepageicon.jpg`
- **Result.** `TRUE`

**Example 2:** Determine whether the suffix of an HTTP request is one of the file extensions defined in the "imagetypes" pattern set, and return the index of that pattern.

- **Expression.** `HTTP.REQ.URL.SUFFIX.EQUALS_INDEX("imagetypes")`
- **Sample URL.** `http://www.example.com/mylogo.gif`
- **Result.** `4` (The index value of the pattern "gif".)

**Example 3:** Use the index value of a pattern to determine whether the URL suffix is within a specified index-value range.

- **Expression.** `HTTP.REQ.URL.SUFFIX.EQUALS_INDEX("imagetypes").GE(3) && HTTP.REQ.URL.SUFFIX.EQUALS_INDEX("imagetypes").LE(5)`
- **Sample URL.** `http://www.example.com/mylogo.gif`
- **Result.** `TRUE` (The index value of gif file types is 4.)

**Example 4:** Implement one set of policies for file extensions bmp, jpg, and png, and a different set of policies for gif, tiff, and svg files.

An expression that returns the index of a matched pattern can be used to define traffic subsets for a web application. The following two expressions could be used in content switching policies for a content switching virtual server:

- `HTTP.REQ.URL.SUFFIX.EQUALS_INDEX("imagetypes").LE(3)`
- `HTTP.REQ.URL.SUFFIX.EQUALS_INDEX("imagetypes").GE(4)`

## Variables

Variables are named objects that store information in the form of tokens. These tokens are used within and across different transactions on the NetScaler Appliance for internal computation and policy processing.

The NetScaler appliance supports creation of variables of the following types:

- **Singleton variables.** Can have a single value of one of the following types: `ulong` and `text` (max-size). The `ulong` type is an unsigned 64-bit integer, the `text` type is a sequence of bytes, and max-size is the maximum number of bytes in the sequence.
- **Map variables.** Maps hold values associated with keys: each key-value pair is called a map entry. The key for each entry is unique within the map. Maps are specified as follows:

`map (key_type, value_type, max-values).`

where,

*key\_type* is the data type of the key. It is of type `text` (max-size).

*value\_type* is the data type of the values of the map. It can be of type `ulong` or `text` (max-size).

*max-values* is the maximum number of entries that the map can contain. It is of type `ulong`.

Values for these variables are set using assignments which must be invoked on policy actions.

Note: Variables are not yet supported in a high-availability setup or in a cluster.

## Variables Scope

A map variable or a singleton variable can have a global scope. Alternatively, the scope of a singleton variable can be limited to a single transaction.

- **Global Scope Variable** - A variable with global scope (the default) has only one instance, and that instance has the same value(s) across all cores of a NetScaler appliance and across all nodes of a cluster or HA configuration. Global variable values exist until they are explicitly deleted, until they expire, or until a standalone appliance is restarted or all nodes of a cluster or HA configuration are restarted.
- **Transaction Scope Variable** - A variable with transaction scope has a separate instance, with its own value, for each transaction processed by the NetScaler appliance. When the transaction processing is complete, the transaction variable value is deleted.

Note: Transaction scope variables are available in NetScaler release 10.5.e or later.



## Configuring and Using Variables

You must first create a variable and then assign a value or specify the operation that must be performed on the variable. After performing these operations, you can use the assignment as a policy action.

Note: Once configured, a variable's settings cannot be modified or reset. If the variable needs to be changed, the variable and all references to the variable (expressions and assignments) must be deleted. The variable can then be re-added with new settings, and the references (expressions and assignments) can be re-added.

### To configure variables by using the command line interface

1. Create a variable.

```
add ns variable <name> -type <string> [-scope global] [-ifFull ( undef | lru )] [-ifValueTooBig ( undef | truncate )] [-ifNoValue ( undef | init )] [-init <string>] [-expires <positive_integer>] [-comment <string>]
```

Note: Refer to the man page "man add ns variable" for description of the command parameters.

**Example 1:** Create a ulong variable named "my\_counter" and initialize it to 1.

```
add ns variable my_counter -type ulong -init 1
```

**Example 2:** Create a map named "user\_privilege\_map". The map will contain keys of maximum length 15 characters and text values of maximum length 10 characters, with a maximum of 10000 entries.

```
add ns variable user_privilege_map -type map(text(15),text(10),10000)
```

Note: If the map contains 10000 unexpired entries, assignments for new keys reuse one of the least recently used entries. By default, an expression trying to get a value for a non-existent key will initialize an empty text value.

2. Assign the value or specify the operation to be performed on the variable. This is done by creating an assignment.

```
add ns assignment <name> -variable <expression> [-set <expression> | -add <expression> | -sub <expression> | -append <expression> | -clear] [-comment <string>]
```

Note: A variable is referenced by using the variable selector (\$). Therefore, **\$variable1** is used to refer to text or ulong variables. Similarly, **\$variable2[key-expression]** is used to refer to map variables.

**Example 1:** Define an assignment named "inc\_my\_counter" that automatically adds 1 to the "my\_counter" variable.

```
add ns assignment inc_my_counter -variable $my_counter -add 1
```

**Example 2:** Define an assignment named "set\_user\_privilege" that adds to the "user\_privilege\_map" variable an entry for the client's IP address with the value returned by the "get\_user\_privilege" HTTP callout.

```
add ns assignment set_user_privilege -variable $user_privilege_map[client.ip.src.typecs
```

Note: If an entry for that key already exists, the value will be replaced. Otherwise a new entry for the key and value will be added. Based on the previous declaration for user\_privilege\_map, if the map already has 10000 entries, one of the least recently used entries will be reused for the new key and value.

3. Invoke the variable assignment in a policy.

There are two functions that can operate on map variables.

- o **\$name.valueExists(key-expression)**. Returns true if there is a value in the map selected by the key-expression. Otherwise returns false. This function will update the expiration and LRU information if the map entry exists, but will not create a new map entry if the value does not exist.
- o **\$name.valueCount**. Returns the number of values currently held by the variable. This is the number of entries in a map. For a singleton variable, this is 0 if the variable is uninitialized or 1 otherwise.

**Example:** Invoke the assignment named "set\_user\_privilege" with a compression policy.

```
> add cmp policy set_user_privilege_pol -rule $user_privilege_map.valueExists(client.ip
```

### To configure variables by using the configuration utility

1. Navigate to AppExpert > NS Variables, to create a variable.
2. Navigate to AppExpert > NS Assignments, to assign value(s) to the variable.
3. Navigate to the appropriate feature area where you want to configure the assignment as an action.

## Use Case: Caching User Privileges

In this use case, user privileges ("GOLD", "SILVER", and so on) must be retrieved from an external web service.

**To achieve this use case, perform the following operations:**

1. Create an HTTP callout to fetch the user privileges from the external web service.

```
> add policy httpcallout get_user_privilege
> set policy httpcallout get_user_privilege -ipaddress 10.217.193.84 -port 80 -returntype
```

2. Store the privileges in a variable.

```
> add ns variable user_privilege_map -type map(text(15),text(10),10000) -expires 1200
> add ns assignment set_user_privilege -variable $user_privilege_map[client.ip.src] -se
```

3. Create a policy to check if there is already a cached entry for the client's IP address; if not, it calls the HTTP callout to set a map entry for the client.

```
> add cmp policy set_user_privilege_pol -rule $user_privilege_map.valueExists(client.ip
```

4. Create a policy that compresses if the cached privilege entry for the client is "GOLD".

```
> add cmp policy compress_if_gold_privilege_pol -rule '$user_privilege_map[client.ip.sr
```

5. Bind the compression policies globally.

```
> bind cmp global set_user_privilege_pol -priority 10 NEXT
> bind cmp global compress_if_gold_privilege_pol -priority 20 END
```

## Use Case: Limiting the Number of Sessions

In this use case, the requirement is to limit the number of active backend sessions. In the deployment, each session login has login in the URL and each session logout has logout in the URL. On successful login, the backend sets a sessionid cookie with a unique 10 character value.

**To achieve this use case, perform the following operations:**

1. Create a map variable that can store each active session. The key of the map is the sessionid. The expiry time for the variable is set to 600 seconds (10 minutes).

```
> add ns variable session_map -type map(text(10),ulong,100) -expires 600
```

2. Create the following assignments for the map variable:

- o Create an entry for the sessionid and set that value to 1 (this value is not actually used).

```
> add ns assignment add_session -variable '$session_map[http.req.cookie.val
```

- o Deallocate the entry for a session ID, which implicitly decrements the value count for session\_map.

```
> add ns assignment delete_session -variable '$session_map[http.req.cookie.
```

3. Create responder policies for the following:

- o To check if a map entry exists for that sessionid in the HTTP request. The add\_session assignment is executed if the map entry does not exist.

```
> add responder policy add_session_pol '$session_map.valueExists(http.req.c
```

Note: The valueExists() function in the add\_session\_pol policy counts as a reference to the session's map entry, so each request resets the expiration timeout for its session. If no requests for a session are received after 10 minutes, the session's entry will be deallocated.

- o To check when the session is logged out. The delete\_session assignment is executed.

```
> add responder policy delete_session_pol 'http.req.url.contains("logout")'
```

- o To check for login requests and if the number of active sessions exceed 100. If these conditions are satisfied, in order to limit the number of sessions, the user is redirected to a page that indicates that the server is busy.

```
> add responder action redirect_too_busy redirect "/too_busy.html"
```

```
> add responder policy check_login_pol 'http.req.url.contains("login") && $
```

4. Bind the responder policies globally.

```
> bind responder global add_session_pol 10 next
> bind responder global delete_session_pol 20 next
> bind responder global check_login_pol 30
```

## Policies and Expressions

The following topics provide the conceptual and reference information that you require for configuring advanced policies on the Citrix® NetScaler® appliance.

You can also download a list of all the expressions supported on the NetScaler appliance and the hierarchical order in which they can be invoked. The reference is in a zip file which you can download from:

- For NetScaler 10.5: <http://support.citrix.com/article/CTX141344>
- For NetScaler 10.1: <http://support.citrix.com/article/CTX137705>

Introduction to Policies and Expressions	Describes the purpose of expressions, policies, and actions, and how different NetScaler applications make use of them.
Configuring Advanced Policies	Describes the structure of advanced policies and how to configure them individually and as policy banks.
Configuring Advanced Expressions: Getting Started	Describes expression syntax and semantics, and briefly introduces how to configure expressions and policies.
Advanced Expressions: Evaluating Text	Describes expressions that you configure when you want to operate on text (for example, the body of an HTTP POST request or the contents of a user certificate).
Advanced Expressions: Working with Dates, Times, and Numbers	Describes expressions that you configure when you want to operate on any type of numeric data (for example, the length of a URL, a client's IP address, or the date and time that an HTTP request was sent).
Advanced Expressions: Parsing HTTP, TCP, and UDP Data	Describes expressions for parsing IP and IPv6 addresses, MAC addresses, and data that is specific to HTTP and TCP traffic.
Advanced Expressions: Parsing SSL Certificates	Describes how to configure expressions for SSL traffic and client certificates, for example, how to retrieve the expiration date of a certificate or the certificate issuer.
Advanced Expressions: IP and MAC Addresses, Throughput, VLAN IDs	Describes expressions that you can use to work with any other client- or server-related data not discussed in other chapters.
Typecasting Data	Describes expressions for transforming data of one type to another.
Regular Expressions	Describes how to pass regular expressions as arguments to operators in advanced expressions.
Configuring Classic Policies and Expressions	Provides details on how to configure the simpler policies and expressions known as classic policies and classic expressions.
Expressions Reference	A reference for classic and advanced expression arguments.
Summary Examples of Advanced Expressions and Policies	Examples of classic and advanced expressions and policies, in both quick reference and tutorial format, that you can customize for your own use.
Tutorial Examples of Advanced Policies for Rewrite	Examples of advanced policies for use in the Rewrite feature.

Tutorial Examples of Classic Policies	Examples of classic policies for NetScaler features such as application firewall and SSL.
Migration of Apache mod_rewrite Rules to Advanced Policies	Examples of functions that were written using the Apache HTTP Server mod_rewrite engine, with examples of these functions after translation into Rewrite and Responder policies on the NetScaler.

## Introduction to Policies and Expressions

For many NetScaler features, policies control how a feature evaluates data, which ultimately determines what the feature does with the data. A policy uses a logical expression, also called a rule, to evaluate requests, responses, or other data, and applies one or more actions determined by the outcome of the evaluation. Alternatively, a policy can apply a profile, which defines a complex action.

Some NetScaler features use default syntax policies, which provide greater capabilities than do the older, classic, policies. If you migrated to a newer release of the NetScaler software and have configured classic policies for features that now use default syntax policies, you might have to manually migrate policies to the default syntax.

This document contains the following details:

- [Classic and Default Syntax Policies](#)
- [Classic and Default Syntax Expressions](#)
- [Converting Classic Expressions to the Newer Default Expression Syntax](#)
- [Before You Proceed](#)

## Classic and Default Syntax Policies

Classic policies evaluate basic characteristics of traffic and other data. For example, classic policies can identify whether an HTTP request or response contains a particular type of header or URL.

Default syntax policies can perform the same type of evaluations as classic policies. In addition, default syntax policies enable you to analyze more data (for example, the body of an HTTP request) and to configure more operations in the policy rule (for example, transforming data in the body of a request into an HTTP header).

In addition to assigning a policy an action or profile, you bind the policy to a particular point in the processing associated with the NetScaler features. The bind point is one factor that determines when the policy will be evaluated.

This document includes the following details:

- [Benefits of Using Default Syntax Policies](#)
- [Basic Components of a Classic or Default Syntax Policy](#)
- [How Different NetScaler Features Use Policies](#)
- [About Actions and Profiles](#)
- [About Policy Bindings](#)
- [About Evaluation Order of Policies](#)
- [Order of Evaluation Based on Traffic Flow](#)

## Benefits of Using Default Syntax Policies

Default syntax policies use a powerful expression language that is built on a class-object model, and they offer several options that enhance your ability to configure the behavior of various NetScaler features. With default syntax policies, you can do the following:

- Perform fine-grained analyses of network traffic from layers 2 through 7.
- Evaluate any part of the header or body of an HTTP or HTTPS request or response.
- Bind policies to the multiple bind points that the default syntax policy infrastructure supports at the default, override, and virtual server levels.
- Use goto expressions to transfer control to other policies and bind points, as determined by the result of expression evaluation.
- Use special tools such as pattern sets, policy labels, rate limit identifiers, and HTTP callouts, which enable you to configure policies effectively for complex use cases.

Additionally, the configuration utility extends robust graphical user interface support for default syntax policies and expressions and enables users who have limited knowledge of networking protocols to configure policies quickly and easily. The configuration utility also includes a policy evaluation feature for default syntax policies. You can use this feature to evaluate a default syntax policy and test its behavior before you commit it, thus reducing the risk of configuration errors.

## Basic Components of a Classic or Default Syntax Policy

Updated: 2013-09-02

Following are a few characteristics of both classic and default syntax policies:

Name.

Each policy has a unique name.

Rule.

The rule is a logical expression that enables the NetScaler feature to evaluate a piece of traffic or another object.

For example, a rule can enable the NetScaler to determine whether an HTTP request originated from a particular IP address, or whether a Cache-Control header in an HTTP request has the value "No-Cache."

Default syntax policies can use all of the expressions that are available in a classic policy, with the exception of classic expressions for the SSL VPN client. In addition, default syntax policies enable you to configure more complex expressions.

Bindings.

To ensure that the NetScaler can invoke a policy when it is needed, you associate the policy, or bind it, to one or more bind points.

You can bind a policy globally or to a virtual server. For more information, see "[About Policy Bindings](#)."

An associated action.

An action is a separate entity from a policy. Policy evaluation ultimately results in the NetScaler performing an action.

For example, a policy in the integrated cache can identify HTTP requests for .gif or .jpeg files. An action that you associate with this policy determines that the responses to these types of requests are served from the cache.

For some features, you configure actions as part of a more complex set of instructions known as a profile. For more information, see "[Order of Evaluation Based on Traffic Flow](#)."

## How Different NetScaler Features Use Policies

Updated: 2013-09-30

The NetScaler supports a variety of features that rely on policies for operation. The following table summarizes how the NetScaler features use policies.

Table 1. NetScaler Feature, Policy Type, and Policy Usage

Feature Name	Policy Type	How You Use Policies in the Feature
System	Classic	<p>For the Authentication function, policies contain authentication schemes for different authentication methods.</p> <p>For example, you can configure LDAP and certificate-based authentication schemes.</p> <p>You also configure policies in the Auditing function.</p>
DNS	Default	To determine how to perform DNS resolution for requests.
SSL	Classic and Default	<p>To determine when to apply an encryption function and add certificate information to clear text.</p> <p>To provide end-to-end security, after a message is decrypted, the SSL feature re-encrypts clear text and uses SSL to communicate with Web servers.</p>
Compression	Classic and Default	To determine what type of traffic is compressed.
Integrated Caching	Default	To determine whether HTTP responses are cacheable.
Responder	Default	To configure the behavior of the Responder function.
Protection Features	Classic	To configure the behavior of the Filter, SureConnect, and Priority Queuing functions.
Content Switching	Classic and Default	<p>To determine what server or group of servers is responsible for serving responses, based on characteristics of an incoming request.</p> <p>Request characteristics include device type, language, cookies, HTTP method, content type, and associated cache server.</p>
AAA - Traffic Management	<p>Classic</p> <p>Exceptions:</p> <ul style="list-style-type: none"><li>• Traffic policies support only default syntax policies</li></ul>	<p>To check for client-side security before users log in and establish a session.</p> <p>Traffic policies, which determine whether single sign-on (SSO) is required, use only the default syntax.</p> <p>Authorization policies authorize users and groups that access intranet resources through the appliance.</p>



	<ul style="list-style-type: none"> <li>Authorization policies support both classic and default syntax policies.</li> </ul>	
Cache Redirection	Classic	To determine whether responses are served from a cache or from an origin server.
Rewrite	Default	<p>To identify HTTP data that you want to modify before serving. The policies provide rules for modifying the data.</p> <p>For example, you can modify HTTP data to redirect a request to a new home page, or a new server, or a selected server based on the address of the incoming request, or you can modify the data to mask server information in a response for security purposes.</p> <p>The URL Transformer function identifies URLs in HTTP transactions and text files for the purpose of evaluating whether a URL should be transformed.</p>
Application Firewall	Classic and Default	To identify characteristics of traffic and data that should or should not be admitted through the firewall.
NetScaler Gateway, Clientless Access function	Default	To define rewrite rules for general Web access using the NetScaler Gateway.
NetScaler Gateway	Classic	To determine how the NetScaler Gateway performs authentication, authorization, auditing, and other functions.

## About Actions and Profiles

Updated: 2013-09-30

Policies do not themselves take action on data. Policies provide read-only logic for evaluating traffic. To enable a feature to perform an operation based on a policy evaluation, you configure actions or profiles and associate them with policies.

Note: Actions and profiles are specific to particular features. For information about assigning actions and profiles to features, see the documentation for the individual features.

### About Actions

Actions are steps that the NetScaler takes, depending on the evaluation of the expression in the policy. For example, if an expression in a policy matches a particular source IP address in a request, the action that is associated with this policy determines whether the connection is permitted.

The types of actions that the NetScaler can take are feature specific. For example, in Rewrite, actions can replace text in a request, change the destination URL for a request, and so on. In Integrated Caching, actions determine whether HTTP responses are served from the cache or an origin server.

In some NetScaler features actions are predefined, and in others they are configurable. In some cases, (for example, Rewrite), you configure the actions using the same types of expressions that you use to configure the associated policy rule.

### About Profiles

Some NetScaler features enable you to associate profiles, or both actions and profiles, with a policy. A profile is a collection of settings that enable the feature to perform a complex function. For example, in the application firewall, a profile for XML data can perform multiple screening operations, such as examining the data for illegal XML syntax or evidence of SQL injection.

## Use of Actions and Profiles in Particular Features

The following table summarizes the use of actions and profiles in different NetScaler features. The table is not exhaustive. For more information about specific uses of actions and profiles for a feature, see the documentation for the feature.

Table 2. Use of Actions and Profiles in Different NetScaler Features

Feature	Use of an Action	Use of a Profile
Application firewall	Synonymous with a profile	All application firewall features use profiles to define complex behaviors, including pattern-based learning.  You add these profiles to policies.
NetScaler Gateway	The following features of the NetScaler Gateway use actions: <ul style="list-style-type: none"> <li>o <b>Pre-Authentication.</b> Uses Allow and Deny actions. You add these actions to a profile.</li> <li>o <b>Authorization.</b> Uses Allow and Deny actions. You add these actions to a policy.</li> <li>o <b>TCP Compression.</b> Uses various actions. You add these actions to a policy.</li> </ul>	The following features use a profile: <ul style="list-style-type: none"> <li>o Pre-Authentication</li> <li>o Session</li> <li>o Traffic</li> <li>o Clientless Access</li> </ul> After configuring the profiles, you add them to policies.
Rewrite	You configure URL rewrite actions and add them to a policy.	Not used.
Integrated Caching	You configure caching and invalidation actions within a policy	Not used.
AAA - Traffic Management	You select an authentication type, set an authorization action of ALLOW or DENY, or set auditing to SYSLOG or NSLOG.	You can configure session profiles with a default timeout and authorization action.
Protection Features	You configure actions within policies for the following functions: <ul style="list-style-type: none"> <li>o Filter</li> <li>o Compression</li> <li>o Responder</li> <li>o SureConnect</li> </ul>	Not used.
SSL	You configure actions within SSL policies	Not used.
System	The action is implied. For the Authentication function, it is either Allow or Deny. For Auditing, it is Auditing On or Auditing Off.	Not used.
DNS	The action is implied. It is either Drop Packets or the location of a DNS server.	Not used.
SSL Offload	The action is implied. It is based on a policy that you associate with an SSL virtual server or a service.	Not used.

Compression	Determine the type of compression to apply to the data	Not used.
Content Switching	The action is implied. If a request matches the policy, the request is directed to the virtual server associated with the policy.	Not used.
Cache Redirection	The action is implied. If a request matches the policy, the request is directed to the origin server.	Not used.

## About Policy Bindings

Updated: 2013-09-30

A policy is associated with, or bound to, an entity that enables the policy to be invoked. For example, you can bind a policy to request-time evaluation that applies to all virtual servers. A collection of policies that are bound to a particular bind point constitutes a policy bank.

Following is an overview of different types of bind points for a policy:

Request time global.

A policy can be available to all components in a feature at request time.

Response time global.

A policy can be available to all components in a feature at response time.

Request time, virtual server-specific.

A policy can be bound to request-time processing for a particular virtual server. For example, you can bind a request-time policy to a cache redirection virtual server to ensure that particular requests are forwarded to a load balancing virtual server for the cache, and other requests are sent to a load balancing virtual server for the origin.

Response time, virtual server-specific.

A policy can also be bound to response-time processing for a particular virtual server.

User-defined policy label.

For default syntax policies, you can configure custom groupings of policies (policy banks) by defining a policy label and collecting a set of related policies under the policy label.

Other bind points.

The availability of additional bind points depends on type of policy (classic or default syntax), and specifics of the relevant NetScaler feature. For example, classic policies that you configure for the NetScaler Gateway have user and group bind points.

For additional information about default syntax policy bindings, see ["Binding Policies That Use the Default Syntax"](#) and [Configuring a Policy Bank for a Virtual Server](#). For additional information about classic policy bindings, see ["Configuring a Classic Policy."](#)

## About Evaluation Order of Policies

For classic policies, policy groups and policies within a group are evaluated in a particular order, depending on the following:

- The bind point for the policy, for example, whether the policy is bound to request-time processing for a virtual server or global response-time processing. For example, at request time, the NetScaler evaluates all request-time classic policies before evaluating any virtual server-specific policies.
- The priority level for the policy. For each point in the evaluation process, a priority level that is assigned to a policy determines the order of evaluation relative to other policies that share the same bind point. For example, when the NetScaler evaluates a bank of request-time, virtual server-specific policies, it starts with the policy that is assigned to the lowest priority value. In classic policies, priority levels must be unique across all bind points.

For default syntax policies, as with classic policies, the NetScaler selects a grouping, or bank, of policies at a particular point in overall processing. Following is the order of evaluation of the basic groupings, or banks, of default syntax policies:

1. Request-time global override
2. Request-time, virtual server-specific (one bind point per virtual server)
3. Request-time global default
4. Response-time global override
5. Response-time virtual server-specific

## 6. Response-time global default

However, within any of the preceding banks of policies, the order of evaluation is more flexible than in classic policies. Within a policy bank, you can point to the next policy to be evaluated regardless of the priority level, and you can invoke policy banks that belong to other bind points and user-defined policy banks.

## Order of Evaluation Based on Traffic Flow

As traffic flows through the NetScaler and is processed by various features, each feature performs policy evaluation. Whenever a policy matches the traffic, the NetScaler stores the action and continues processing until the data is about to leave the NetScaler. At that point, the NetScaler typically applies all matching actions. Integrated Caching, which only applies a final Cache or NoCache action, is an exception.

Some policies affect the outcome of other policies. Following are examples:

- If a response is served from the integrated cache, some other NetScaler features do not process the response or the request that initiated it.
- If the Content Filtering feature prevents a response from being served, no subsequent features evaluate the response.

If the application firewall rejects an incoming request, no other features can process it.

## Classic and Default Syntax Expressions

One of the most fundamental components of a policy is its rule. A policy rule is a logical expression that enables the policy to analyze traffic. Most of the policy's functionality is derived from its expression.

An expression matches characteristics of traffic or other data with one or more parameters and values. For example, an expression can enable the NetScaler to accomplish the following:

- Determine whether a request contains a certificate.
- Determine the IP address of a client that sent a TCP request.
- Identify the data that an HTTP request contains (for example, a popular spreadsheet or word processing application).
- Calculate the length of an HTTP request.

This document includes the following details:

- [About Classic Expressions](#)
- [About Default Syntax Expressions](#)

### About Classic Expressions

Classic expressions enable you to evaluate basic characteristics of data. They have a structured syntax that performs string matching and other operations.

Following are a few simple examples of classic expressions:

- An HTTP response contains a particular type of Cache Control header.

```
res.http.header Cache-Control contains public
```

- An HTTP response contains image data.

```
res.http.header Content-Type contains image/
```

- An SSL request contains a certificate.

```
req.ssl.client.cert exists
```

### About Default Syntax Expressions

Updated: 2013-09-02

Any feature that uses default syntax policies also uses default syntax expressions. For information about which features use default syntax policies, see the table "[NetScaler Feature, Policy Type, and Policy Usage](#)."

Default syntax expressions have a few other uses. In addition to configuring default syntax expressions in policy rules, you configure default syntax expressions in the following situations:

Integrated Caching:

You use default syntax expressions to configure a selector for a content group in the integrated cache.

Load Balancing:

You use default syntax expressions to configure token extraction for a load balancing virtual server that uses the TOKEN method for load balancing.

Rewrite:

You use default syntax expressions to configure rewrite actions.

Rate-based policies:

You use default syntax expressions to configure limit selectors when configuring a policy to control the rate of traffic to various servers.

Following are a few simple examples of default syntax expressions:

- An HTTP request URL contains no more than 500 characters.

```
http.req.url.length <= 500
```

- An HTTP request contains a cookie that has fewer than 500 characters.

```
http.req.cookie.length < 500
```

- An HTTP request URL contains a particular text string.

```
http.req.url.contains(".html")
```

# Converting Classic Expressions to the Newer Default Expression Syntax

You can convert a classic expression to the default expression syntax by using the nspepi conversion tool. You can also use the tool to convert all the classic expressions in the NetScaler configuration to the default syntax (with the exception of NetScaler entities that currently support only classic expressions).

The conversion tool does not convert policies configured for the following features, because the features currently support only classic policies:

- Authentication, Pre-authentication
- SSL
- Cache redirection
- VPN (session, traffic, and tunnel traffic)
- Content filtering (The responder feature not only provides you with functionality that is equivalent to that provided by the content filtering feature but also surpasses the content filtering feature in the use cases that it supports. Additionally, responder supports the more powerful default syntax for policy expressions.)

The following NetScaler features support both classic and default syntax expressions and, therefore, support the conversion of classic expressions to default syntax expressions:

- Application firewall policies
- Authorization policies
- Named expressions
- Compression policies
- Content switching policies
- User-defined, rule-based tokens/persistency (the “rule parameter value that is specified for a load balancing virtual server)

This document includes the following details:

- [About the Conversion Process](#)
- [Converting Expressions](#)
- [Converting a NetScaler Configuration File](#)
- [Conversion Warnings](#)

## About the Conversion Process

Updated: 2013-09-02

When parsing a NetScaler configuration file, the conversion tool performs the following actions:

1. In commands that create classic named expressions, the conversion tool replaces the names of the classic expressions with default syntax expressions.
2. In commands that support only the classic syntax, if classic named expressions are used, the conversion tool replaces the names of the classic expressions with the actual classic expressions they represent. This action ensures that the names of expressions in classic-only features do not reference the default syntax expressions created from Step 1.
3. In commands associated with entities that support both the classic syntax and the default syntax, the conversion tool replaces all classic expressions in commands with default syntax expressions.

### Example

Consider the following sample configuration commands:

```
add policy expression ne_c1 "METHOD == GET"
add policy expression ne_c2 "ne_c1 || URL == /*.htm "
add filter policy pol1 -rule "ne_c2" -reqAction YES
add cmp policy pol2 -rule "REQ.HTTP.HEADER Accept CONTAINS \'text/html\'" -resAction COMPRES
add cmp policy pol3 -rule "ne_c1 || ne_c2" -resAction GZIP
```

In the commands that create the classic named expressions ne\_c1 and ne\_c2, the tool replaces the names of the expressions with actual default syntax expressions. This action, which corresponds to Step 1 described earlier, results in the following commands:

```
add policy expression ne_c1 "HTTP.REQ.METHOD.EQ(\"GET\")"

add policy expression ne_c2 "HTTP.REQ.URL.SUFFIX.EQ(\"htm\")"
```

The filter policy command supports only the classic syntax. Therefore, the conversion tool replaces the classic named expression `ne_c1` with the actual classic expression it represents. Note that the tool replaces `ne_c1` in the expression for `ne_c2`, and then replaces `ne_c2` in the filter policy with the classic expression. This action, which corresponds to Step 2 described earlier, results in the following command:

```
add filter policy pol1 -rule "METHOD == GET || URL == /*.htm" -reqAction YES
```

The compression feature supports both classic and default syntax expressions. Therefore, in the command that creates the compression policy `pol2`, the conversion tool replaces the expression with a default syntax expression. This action, which corresponds to Step 3 described earlier, results in the following command:

```
add cmp policy pol2 -rule "HTTP.REQ.HEADER(\"Accept\").AFTER_STR(\"text/html\").LENGTH.GT(0)" -resAction COMPRESS
```

The command that creates the compression policy `pol3` is unaffected by the conversion process because, after the conversion process is complete, `ne_c1` and `ne_c2` reference the default syntax expressions that result from Step 1.

Client security messages are not supported in the newer default policy format and, therefore, are lost. The `SYS.EVAL_CLASSIC_EXPR` function is replaced with a default policy expression. The following entities support the `SYS.EVAL_CLASSIC_EXPR` function:

- o DNS policies
- o Rate limit selectors
- o Cache selectors
- o Cache policies
- o Content switching policies
- o Rewrite policies
- o URL transformation policies
- o Responder policies
- o Application firewall policies
- o Authorization policies
- o Compression policies
- o CVPN access policies

After performing the conversion, the tool saves the changes in a new configuration file. The new configuration file is created in the directory in which the input file exists. The name of the new configuration file is the same as the name of the input configuration file except for the string `new_` used as a prefix. Conversion warnings are reported in a warning line at the end of the screen output. Additionally, a warning file is created in the directory in which the input configuration file resides. For more information about the warning file and the types of warnings that are reported, see ["Conversion Warnings."](#)

## Converting Expressions

Updated: 2013-09-02

You can use the `nspepi` tool to convert a single classic expression to the default syntax. The `nspepi` tool must be run from the shell prompt on the NetScaler appliance.

### To convert a classic expression to the default syntax by using the command line interface

At the shell prompt, type:

```
nspepi -e "<classic expression>"
```

#### Example

```
root@NS# nspepi -e "REQ.HTTP.URL == /*.htm"
"HTTP.REQ.URL.REGEX_MATCH(re#/(.*)\.htm#)"
```

## Converting a NetScaler Configuration File

Updated: 2013-09-02

You can use the `nspepi` tool to convert all the classic expressions in a NetScaler configuration file to the default syntax (except for those commands that do not support the default syntax). The `nspepi` tool must be run from the shell prompt on the NetScaler appliance.



## To convert all the classic expressions in a NetScaler configuration file to the default syntax by using the command line interface

At the shell prompt, type:

```
nspepi -f "<ns config file>" -v
```

### Example

```
root@NS# nspepi -f ns.conf
OUTPUT: New configuration file created: new_ns.conf
OUTPUT: New warning file created: warn_ns.conf
WARNINGS: Total number of warnings due to bind commands: 18
WARNINGS: Line numbers which has bind command issues: 305, 306, 706, 707, 708, 709, 710, 711
714, 715, 767, 768, 774, 775, 776, 777
root@NS#
```

## Conversion Warnings

When classic expressions that are included in CLI commands are upgraded to the default syntax, the number of characters in the expression might exceed the 1499-character limit. The commands that include expressions longer than 1499 characters fail when the configuration is being applied. You must manually update these commands.

In addition, multiple classic policies can be bound to a given bind point with priority 0 or with equal priority, but the default syntax policy infrastructure does not support a priority value of 0 or policies with the same priority at a given bind point. These commands fail when the configuration is being applied. The commands must be updated manually with the correct priority values.

The line numbers of lines that threw a warning during conversion are listed at the end of the output in a warning line. In addition, a warning file is created in the same directory as the one in which the old and new configuration files reside. The name of the warning file is the same as the name of the input configuration file except that the string `warn_` is added as a prefix.

## Before You Proceed

Before configuring expressions and policies, be sure you understand the relevant NetScaler feature and the structure of your data, as follows:

- Read the documentation on the relevant feature.
- Look at the data stream for the type of data that you want to configure.

You may want to run a trace on the type of traffic or content that you want to configure. This will give you an idea of the parameters and values, and operations on these parameters and values, that you need to specify in an expression.

Note: The NetScaler supports either classic or default syntax policies within a feature. You cannot have both types in the same feature. Over the past few releases, some NetScaler features have migrated from using classic policies and expressions to default syntax policies and expressions. If a feature of interest to you has changed to the default syntax format, you may have to manually migrate the older information. Following are guidelines for deciding if you need to migrate your policies:

- If you configured classic policies in a version of the Integrated Caching feature prior to release 9.0 and then upgrade to version 9.0 or later, there is no impact. All legacy policies are migrated to the default syntax policy format.
- For other features, you need to manually migrate classic policies and expressions to the default syntax if the feature has migrated to the default syntax.

## Configuring Default Syntax Policies

You can create default syntax policies for various NetScaler features, including DNS, Rewrite, Responder, and Integrated Caching, and the clientless access function in the NetScaler Gateway. Policies control the behavior of these features.

When you create a policy, you assign it a name, a rule (an expression), feature-specific attributes, and an action that is taken when data matches the policy. After creating the policy, you determine when it is invoked by binding it globally or to either request-time or response-time processing for a virtual server.

Policies that share the same bind point are known as a *policy bank*. For example, all policies that are bound to a virtual server constitute the policy bank for the virtual server. When binding the policy, you assign it a priority level to specify when it is invoked relative to other policies in the bank. In addition to assigning a priority level, you can configure an arbitrary evaluation order for policies in a bank by specifying Goto expressions.

In addition to policy banks that are associated with a built-in bind point or a virtual server, you can configure *policy labels*. A policy label is a policy bank that is identified by an arbitrary name. You invoke a policy label, and the policies in it, from a global or virtual-server-specific policy bank. A policy label or a virtual-server policy bank can be invoked from multiple policy banks.

For some features, you can use the policy manager to configure and bind policies.

## Rules for Names in Identifiers Used in Policies

The names of identifiers in the named expression, HTTP callout, pattern set, and rate limiting features must begin with an ASCII alphabet or an underscore (\_). The remaining characters can be ASCII alphanumeric characters or underscores (\_).

The names of these identifiers must not begin with the following reserved words:

- The words `ALT`, `TRUE`, or `FALSE` or the `Q` or `S` one-character identifier.
- The special-syntax indicator `RE` (for regular expressions) or `XP` (for XPath expressions).
- Expression prefixes, which currently are the following:

- `CLIENT`
  - `EXTEND`
  - `HTTP`
  - `SERVER`
  - `SYS`
  - `TARGET`
  - `TEXT`
  - `URL`
  - `MYSQL`
  - `MSSQL`

Additionally, the names of these identifiers cannot be the same as the names of enumeration constants used in the policy infrastructure. For example, the name of an identifier cannot be `IGNORECASE`, `YEAR`, or `LATIN2_CZECH_CS` (a MySQL character set).

Note: The NetScaler appliance performs a case-insensitive comparison of identifiers with these words and enumeration constants. For example, names of the identifiers cannot begin with `TRUE`, `True`, or `true`.

## Creating or Modifying a Policy

All policies have some common elements. Creating a policy consists, at minimum, of naming the policy and configuring a rule. The policy configuration tools for the various features have areas of overlap, but also differences. For the details of configuring a policy for a particular feature, including associating an action with the policy, see the documentation for the feature.

To create a policy, begin by determining the purpose of the policy. For example, you may want to define a policy that identifies HTTP requests for image files, or client requests that contain an SSL certificate. In addition to knowing the type of information that you want the policy to work with, you need to know the format of the data that the policy is analyzing.

Next, determine whether the policy is globally applicable, or if it pertains to a particular virtual server. Also consider the effect that the order in which your policies are evaluated (which will be determined by how you bind the policies) will have on the policy that you are about to configure.

### To create a policy by using the command line interface

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

- `add responder|dns|cs|rewrite|cache policy <policyName> -rule <expression> [<feature-specific information>]`
- `show rewrite policy <name>`

#### Example 1:

```
add rewrite policy "pol_remove-ae" true "act_remove-ae"
Done
> show rewrite policy pol_remove-ae
Name: pol_remove-ae
Rule: true
RewriteAction: act_remove-ae
UndefAction: Use Global
Hits: 0
Undef Hits: 0
Bound to: GLOBAL RES_OVERRIDE
Priority: 90
GotoPriorityExpression: END
Done
>
```

#### Example 2:

```
add cache policy BranchReportsCachePolicy -rule q{http.req.url.query.value("actionoverride").contains("branchReports")}
Done
show cache policy BranchReportsCachePolicy
Name: BranchReportsCachePolicy
Rule: http.req.url.query.value("actionoverride").contains("branchReports")
CacheAction: CACHE
Stored in group: DEFAULT
UndefAction: Use Global
Hits: 0
Undef Hits: 0
Done
```

Note: At the command line, quote marks within a policy rule (the expression) must be escaped or delimited with the `q` delimiter. For more information, see ["Configuring Default Syntax Expressions in a Policy."](#)

### To create or modify a policy by using the configuration utility

1. In the navigation pane, expand the name of the feature for which you want to configure a policy, and then click Policies. For example, you can select Content Switching, Integrated Caching, DNS, Rewrite, or Responder.
2. In the details pane, click Add, or select an existing policy and click Open. A policy configuration dialog box appears.
3. Specify values for the following parameters. (An asterisk indicates a required parameter. For a term in parentheses, see the corresponding parameter in "Parameters for creating or modifying a policy.")

4. Click Create, and then click Close.

5. Click Save. A policy is added.

Note: After you create a policy, you can view the policy's details by clicking the policy entry in the configuration pane. Details that are highlighted and underlined are links to the corresponding entity (for example, a named expression).

## Policy Configuration Examples

These examples show how policies and their associated actions are entered at the command line interface. In the configuration utility, the expressions would appear in the Expression window of the feature-configuration dialog box for the integrated caching or rewrite feature.

Following is an example of creating a caching policy. Note that actions for caching policies are built in, so you do not need to configure them separately from the policy.

```
add cache policy BranchReportsCachePolicy -rule q{http.req.url.query.value("actionoverride"
```

Following is an example of a Rewrite policy and action:

```
add rewrite action myAction1 INSERT_HTTP_HEADER "myHeader" "valueForMyHeader"  
add rewrite policy myPolicy1 "http.req.url.contains(\"myURLstring\")" myAction1
```

Note: At the command line, quote marks within a policy rule (the expression) must be escaped or delimited with the q delimiter. For more information, see ["Configuring Default Syntax Expressions in a Policy."](#)

## Binding Policies That Use the Default Syntax

After defining a policy, you indicate when the policy is to be invoked by binding the policy to a bind point and specifying a priority level. You can bind a policy to only one bind point. A bind point can be global, that is, it can apply to all virtual servers that you have configured. Or, a bind point can be specific to a particular virtual server, which can be either a load balancing or a content switching virtual server. Not all bind points are available for all features.

The order in which policies are evaluated determines the order in which they are applied, and features typically evaluate the various policy banks in a particular order. Sometimes, however, other features can affect the order of evaluation. Within a policy bank, the order of evaluation depends on the values of parameters configured in the policies. Most features apply all of the actions associated with policies whose evaluation results in a match with the data that is being processed. The integrated caching feature is an exception.

## Feature-Specific Differences in Policy Bindings

You can bind policies to built-in, global bind points (or banks), to virtual servers, or to policy labels.

However, the NetScaler features differ in terms of the types of bindings that are available. The following table summarizes how you use policy bindings in various NetScaler features that use policies.

Table 1. Feature-Specific Bindings for Policies

Feature Name	Virtual Servers Configured in the Feature	Policies Configured in the Feature	Bind Points Configured for the Policies	Use of Policies in the Feature
DNS	none	DNS policies	Global	To determine how to perform DNS resolution for requests.
Content Switching Note: This feature can support either or classic policies or policies that use the default syntax, but not both.	Content Switching (CS)	Content Switching policies	<ul style="list-style-type: none"><li>Content switching or cache redirection virtual server</li><li>Policy label</li></ul>	<p>To determine what server or group of servers is responsible for serving responses, based on characteristics of an incoming request.</p> <p>Request characteristics include device type, language, cookies, HTTP method, content type, and associated cache server.</p>
Integrated Caching	none	Caching policies	<ul style="list-style-type: none"><li>Global override</li><li>Global default</li><li>Policy label</li><li>Load balancing, content switching, or SSL offload virtual server</li></ul>	To determine whether HTTP responses can be stored in, and served from, the NetScaler appliance's integrated cache.
Responder	none	Responder policies	<ul style="list-style-type: none"><li>Global override</li></ul>	To configure the behavior of the Responder function.



			<ul style="list-style-type: none"> <li>Global default</li> <li>Policy label</li> <li>Load balancing, content switching, or SSL offload virtual server</li> </ul>	
Rewrite	none	Rewrite policies	<ul style="list-style-type: none"> <li>Global override</li> <li>Global default</li> <li>Policy label</li> <li>Load balancing, content switching, or SSL offload virtual server</li> </ul>	<p>To identify HTTP data that you want to modify before serving. The policies provide rules for modifying the data.</p> <p>For example, you can modify HTTP data to redirect a request to a selected server based on the address of the incoming request, or to mask server information in a response for security purposes.</p>
URL Transform function in the Rewrite feature	none	Transformation policies	<ul style="list-style-type: none"> <li>Global override</li> <li>Global default</li> <li>Policy label</li> </ul>	To identify URLs in HTTP transactions and text files for the purpose of evaluating whether a URL should be altered.
NetScaler Gateway (clientless VPN functions only)	VPN server	Clientless Access policies	<ul style="list-style-type: none"> <li>VPN Global</li> <li>VPN server</li> </ul>	To determine how the NetScaler Gateway performs authentication, authorization, auditing, and other functions, and to define rewrite rules for general Web access using the NetScaler Gateway.

## Bind Points and Order of Evaluation

For a policy to take effect, you must ensure that the policy is invoked at some point during processing. To do so, you associate the policy with a bind point. The collection of policies that is bound to a bind point is known as a policy bank.

Following are the bind points that the NetScaler evaluates, listed in the typical order of evaluation within a policy bank

1. **Request-time override.** When a request flows through a feature, the NetScaler first evaluates request-time override policies for the feature.
2. **Request-time Load Balancing virtual server.** If policy evaluation cannot be completed after all the request-time override policies have been evaluated, the NetScaler processes request-time policies for load balancing virtual servers.
3. **Request-time Content Switching virtual server.** If policy evaluation cannot be completed after all the request-time policies for load balancing virtual servers have been evaluated, the NetScaler processes request-time policies for content switching virtual servers.
4. **Request-time default.** If policy evaluation cannot be completed after all request-time, virtual server-specific policies have been evaluated, the NetScaler processes request-time default policies.
5. **Response-time override.** At response time, the NetScaler starts with policies that are bound to the response-time override bind point.

6. **Response-time Load Balancing virtual server.** If policy evaluation cannot be completed after all response-time override policies have been evaluated, the NetScaler process the response-time policies for load balancing virtual servers.
7. **Response-time Content Switching virtual server.** If policy evaluation cannot be completed after all policies have been evaluated for load balancing virtual servers, the NetScaler process the response-time policies for content switching virtual servers.
8. **Response-time default.** If policy evaluation cannot be completed after all response-time, virtual-server-specific policies have been evaluated, the NetScaler processes response-time default policies.

## Policy Evaluation across Features

In addition to attending to evaluation of policies within a feature, if you have bound policies to a content switching virtual server, note that these policies are evaluated before other policies. Binding a policy to a content switching vserver produces a different result in NetScaler versions 9.0.x and later than in 8.x versions. In NetScaler 9.0 and later versions, evaluation occurs as follows:

- Content switching policies are evaluated before other policies. If a content switching policy evaluates to TRUE, the target load balancing vserver is selected.
- If all content switching policies evaluate to FALSE, the default load balancing vserver under the content switching VIP is selected.

After a target load balancing vserver is selected by the content switching process, policies are evaluated in the following order:

1. Policies that are bound to the global override bind point.
2. Policies that are bound to the default load balancing vserver.
3. Policies that are bound to the target content switching vserver.
4. Policies that are bound to the global default bind point.

To be sure that the policies are evaluated in the intended order, follow these guidelines:

- Make sure that the default load balancing vserver is not directly reachable from the outside; for example, the vserver IP address can be 0.0.0.0.
- To prevent exposing internal data on the load balancing default vserver, configure a policy to respond with a 503 Service Unavailable status and bind it to the default load balancing vserver.

## Entries in a Policy Bank

Each entry in a policy bank has, at minimum, a policy and a priority level. You can also configure entries that change the priority-based evaluation order, and you can configure entries that invoke external policy banks.

The following table summarizes each entry in a policy bank.

Table 2. Format of Each Entry in a Policy Bank

Policy Name	Priority	Goto Expression	Invocation Type	Policy Bank to Be Invoked
The policy name, or a dummy policy named NOPOLICY. The NOPOLICY entry controls evaluation flow without processing a rule.	An integer.	Optional.  Identifies the next policy in the bank to evaluate, or ends any further evaluation	Optional.  Indicates that an external policy bank will be invoked.  This field restricts the choices to a global policy label or a virtual server.	Optional.  Used with Invocation Type. This is the label for a policy bank or a virtual server name.  The NetScaler returns to the current bank after processing the external bank.

If the policy evaluates to TRUE, the NetScaler stores the action that is associated with the policy. If the policy evaluates to FALSE, the NetScaler evaluates the next policy. If the policy is neither TRUE nor FALSE, the NetScaler uses the associated Undef (undefined) action.

## Evaluation Order within a Policy Bank

Within a policy bank, the evaluation order depends on the following items:

A priority.

The most minimal amount of information about evaluation order is a numeric priority level. The lower the number, the higher the priority.

A Goto expression.

If supplied, the Goto expression indicates the next policy to be evaluated, typically within the same policy bank.. Goto expressions can only proceed forward in a bank. To prevent looping, a policy bank configuration is not valid if a Goto statement points backwards in the bank.

Invocation of other policy banks.

Any entry can invoke an external policy bank. The NetScaler provides a built-in entity named NOPOLICY that does not have a rule. You can add a NOPOLICY entry in a policy bank when you want to invoke another policy bank, but do not want to process any other rules prior to the invocation. You can have multiple NOPOLICY entries in multiple policy banks.

Values for a Goto expression are as follows:

NEXT.

This keyword selects the policy with the next higher priority level in the current policy bank.

An integer.

If you supply an integer, it must match the priority level of another policy in the current policy bank.

END.

This keyword stops evaluation after processing the current policy, and no additional policies in this bank are processed.

Blank.

If the Goto expression is empty, it is the same as specifying END.

A numeric expression.

This is a default syntax expression that resolves to a priority number for another policy in the current bank.

USE\_INVOCATION\_RESULT.

This phrase can be used only if you are invoking an external policy bank. Entering this phrase causes the NetScaler to perform one of the following actions:

- If the final Goto in the invoked policy bank has a value of END or is empty, the invocation result is END, and evaluation stops.
- If the final Goto expression in the invoked policy bank is anything other than END, the NetScaler performs a NEXT.

The following table illustrates a policy bank that uses Goto statements and policy bank invocations.

Table 3. Example of a Policy Bank That Uses Gotos and External Bank Invocations

Policy Name	Priority	Goto	Invocation	Policy Bank to Be Invoked
ClientCertificatePolicy (rule: does the request contain a client certificate?)	100	300	None	None
SubnetPolicy (rule: is the client from a private subnet?)	200	NEXT	None	None
NOPOLICY	300	USE INVOCATION RESULT	Request vserver	My_Request_VServer
NOPOLICY	350	USE INVOCATION RESULT	Policy Label	My_Policy_Label

WorkingHoursPolicy (rule: is it working hours?)	400	END	None	None
---	-----	-----	------	------

## How Policy Evaluation Ends

Evaluation of a policy bank ends when one of the following takes place:

- A policy evaluates to TRUE and its Goto statement value is END.

No further policies or policy banks in this feature are evaluated.

- An external policy bank is invoked, its evaluation returns an END, and the Goto statement uses a value of USE\_INVOCATION\_RESULT or END.

Evaluation continues with the next policy bank for this feature. For example, if the current bank is the request-time override bank, the NetScaler next evaluates request-time policy banks for the virtual servers.

- The NetScaler has walked through all the policy banks in this feature, but has not encountered an END.

If this is the last entry to be evaluated in this policy bank, the NetScaler proceeds to the next feature.

## How Features Use Actions after Policy Evaluation

After evaluating all relevant policies for a particular data point (for example, an HTTP request), the NetScaler stores all the actions that are associated with any policy that matched the data.

For most features, all the actions from matching policies are applied to a traffic packet as it leaves the NetScaler. The Integrated Caching feature only applies one action: CACHE or NOCACHE. This action is associated with the policy with the lowest priority value in the “highest priority” policy bank (for example, request-time override policies are applied before virtual server-specific policies).

## Binding a Policy Globally

The following binding procedures are typical. However, refer to the documentation for the feature of interest to you for complete instructions.

### To bind an Integrated Caching policy globally by using the command line interface

At the command prompt, type the following commands to bind an Integrated Caching policy and verify the configuration:

- `bind cache global <policy> -priority <positiveInteger> [-type REQ_OVERRIDE | REQ_DEFAULT | RES_OVERRIDE | RES_DEFAULT]`
- `show cache global`

#### Example

```
bind cache global _nonPostReq -priority 100 -type req_default
Done
> show cache global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 2

2)      Global bindpoint: RES_DEFAULT
        Number of bound policies: 1
Done
```

The type argument is optional to maintain backward compatibility. 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.

### To bind a Rewrite policy globally by using the command line interface

At the command prompt, type the following commands to bind a Rewrite policy and verify the configuration:

- `bind rewrite global <policyName> <priority> [-type REQ_OVERRIDE | REQ_DEFAULT | RES_OVERRIDE | RES_DEFAULT]`
- `show rewrite global`

#### Example

```
bind rewrite global pol_remove-pdf 100
Done
> show rewrite global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

2)      Global bindpoint: REQ_OVERRIDE
        Number of bound policies: 1
Done
```

The type argument is optional for globally bound policies, to maintain backward compatibility. 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.

### To bind a compression policy globally by using the command line interface

At the command prompt, type the following commands to bind a compression policy and verify the configuration:

- `bind cmp global <policyName> -priority <positiveInteger> [-type REQ_OVERRIDE | REQ_DEFAULT | RES_OVERRIDE | RES_DEFAULT]`
- `show cmp global`

### Example

```
> bind cmp global cmp_pol_1 -priority 100
Done
> show cmp policy cmp_pol_1
Name: cmp_pol_1
Rule: HTTP.REQ.URL.SUFFIX.EQ("BMP")
Response Action: COMPRESS
Hits: 0

Policy is bound to following entities
1) GLOBAL REQ_DEFAULT
Priority: 100
GotoPriorityExpression: END
Done
>
```

## To bind a Responder policy globally by using the command line interface

At the command prompt, type the following commands to bind a Responder policy and verify the configuration:

- `bind responder global <policyName> <priority> [-type OVERRIDE | DEFAULT ]`
- `show responder global`

### Example

```
bind responder global pol404Error1 200
Done
> show responder global
1) Global bindpoint: REQ_DEFAULT
   Number of bound policies: 1

Done
```

## To bind a DNS policy globally by using the command line interface

At the command prompt, type the following commands to bind a DNS policy and verify the configuration:

- `bind dns global <policyName> <priority>`
- `show dns global`

### Example

```
> bind dns global pol_ddos_drop1 150
Done
> show dns global
Policy name : pol_ddos_drop
Priority : 100
Goto expression : END
Policy name : pol_ddos_drop1
Priority : 150
Done
>
```

## To bind an Integrated Caching, Responder, Rewrite, or Compression policy globally by using the configuration utility

1. In the navigation pane, click the name of the feature for which you want to bind the policy.
2. In the details pane, click <Feature Name> policy manager.
3. In the Policy Manager dialog box, select the bind point to which you want to bind the policy (for example, for Integrated Caching, Rewrite, or Compression, you could select Request and Default Global). The Responder does not differentiate between request-time and response-time policies.
4. Click Insert Policy and, from the Policy Name pop-up menu, select the policy name. A priority is assigned automatically to the policy, but you can click the cell in the Priority column and drag it anywhere within the dialog box i

you want the policy to be evaluated after other policies in this bank. The priority is automatically reset. Note that priority values within a policy bank must be unique.

5. Click Apply Changes.
6. Click Close. A message in the status bar indicates that the policy is bound successfully.

## **To bind a DNS policy globally by using the configuration utility**

1. Navigate to Traffic Management > DNS > Policies.
2. In the details pane, click Global Bindings.
3. In the global bindings dialog box, click Insert Policy, and select the policy that you want to bind globally.
4. Click in the Priority field and enter the priority level.
5. Click OK. A message in the status bar indicates that the policy is bound successfully.

## Binding a Policy to a Virtual Server

A globally bound policy applies to all load balancing and content switching virtual servers.

Note that when binding a policy to a virtual server, you must identify it as a request-time or a response-time policy.

### To bind a policy to a load balancing or content switching virtual server by using the command line interface

At the command prompt, type the following commands to bind a policy to a load balancing or content switching virtual server and verify the configuration:

- o `bind lb|cs vserver <virtualServerName> -policyName <policyName> -priority <positiveInteger> -type REQUEST|RESPONSE`
- o `show lb vserver <name>`

#### Example

```
> bind lb vserver lbvip -policyName ns_cmp_msapp -priority 50
Done
> show lb vserver lbvip
    lbvip (8.7.6.6:80) - HTTP          Type: ADDRESS
    State: DOWN
    Last state change was at Wed Jul 15 05:54:24 2009 (+226 ms)
    Time since last state change: 28 days, 01:57:26.350
    Effective State: DOWN
    Client Idle Timeout: 180 sec
    Down state flush: ENABLED
    Disable Primary Vserver On Down : DISABLED
    Port Rewrite : DISABLED
    No. of Bound Services :    0 (Total)          0 (Active)
    Configured Method: LEASTCONNECTION
    Mode: IP
    Persistence: NONE
    Vserver IP and Port insertion: OFF
    Push: DISABLED   Push VServer:
    Push Multi Clients: NO
    Push Label Rule: none

1)      Policy : ns_cmp_msapp Priority:50
2)      Policy : cf-pol Priority:1          Inherited
Done
```

### To bind a policy to an SSL offload virtual server by using the command line interface

At the command prompt, type the following commands to bind a policy to an SSL offload virtual server and verify the configuration:

```
bind ssl vserver <vServerName>@ -policyName <policyName> -priority <positiveInteger>
```

### To bind a policy to a virtual server by using the configuration utility

1. In the navigation pane, expand Traffic Management > Load Balancing, Traffic Management > Content Switching, Traffic Management > SSL Offload, Security > AAA- Application Traffic, or NetScaler Gateway, and then click Virtual Servers.
2. In the details pane, double-click the virtual server to which you want to bind the policy, and then click Open.
3. On the Policies tab, click the icon for the type of policy that you want to bind (the choices are feature-specific), and then click the name of the policy. Note that for some features, you can bind both classic policies and policies that use the default syntax to the virtual server.
4. If you are binding a policy to a Content Switching virtual server, in the Target field select a load balancing virtual server to which traffic that matches the policy is sent.
5. Click OK. A message in the status bar indicates that the policy is bound successfully.



## Displaying Policy Bindings

You can display policy bindings to verify that they are correct.

### To display policy bindings by using the command line interface

At the command prompt, type the following commands to display policy bindings and verify the configuration:  
show rewrite policy <name>

#### Example

```
> show rewrite policy pol_remove-pdf
  Name: pol_remove-pdf
  Rule: http.req.url.contains(".pdf")
  RewriteAction: act_remove-ae
  UndefAction: Use Global
  Hits: 0
  Undef Hits: 0
  Bound to: GLOBAL REQ_DEFAULT
  Priority: 100
  GotoPriorityExpression: END
Done
>
```

### To display global policy bindings for Integrated Caching, Rewrite, or Responder by using the configuration utility

1. In the navigation pane, expand the feature that contains the policy that you want to view, and then click Policies.
2. In the details pane, click the policy. Bound policies have a check mark next to them.
3. At the bottom of the page, under Details, next to Bound to, view the entity to which the policy is bound.

### To display global policy bindings for DNS or Clientless Access in the NetScaler Gateway by using the configuration utility

1. Navigate to Traffic Management > DNS > Policies.
2. In the details pane, click Global Bindings.

### To display global policy bindings for Content Switching by using the configuration utility

1. Navigate to Traffic Management > Content Switching > Policies.
2. In detailed pane, select policy.
3. In the details pane, click Show Bindings.

## Unbinding a Policy

If you want to re-assign a policy or delete it, you must first remove its binding.

### To unbind an integrated caching, rewrite, or compression default syntax policy globally by using the command line interface

At the command prompt, type the following commands to unbind an integrated caching, rewrite, or compression default syntax policy globally and verify the configuration:

- `unbind cache|rewrite|cmp global <policyName> [-type req_override|req_default|res_override|res_default] [-priority <positiveInteger>]`
- `show cache|rewrite|cmp global`

#### Example

```
> unbind cache global_nonPostReq
Done
> show cache global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

2)      Global bindpoint: RES_DEFAULT
        Number of bound policies: 1

Done
```

The priority is required only for the “dummy” policy named NOPOLICY.

### To unbind a responder policy globally by using the command line interface

At the command prompt, type the following commands to unbind a responder policy globally and verify the configuration:

- `unbind responder global <policyName> [-type override|default] [-priority <positiveInteger>]`
- `show responder global`

#### Example

```
> unbind responder global pol404Error
Done
> show responder global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

Done
```

The priority is required only for the “dummy” policy named NOPOLICY.

### To unbind a DNS policy globally by using the command line interface

At the command prompt, type the following commands to unbind a DNS policy globally and verify the configuration:

- `unbind responder global <policyName>`
- `unbind responder global`

#### Example

```
unbind dns global dfgdfg
Done
show dns global
Policy name : dfgdfggfhg
```

```
Priority : 100
Goto expression : END
Done
```

## To unbind a default syntax policy from a virtual server by using the command line interface

At the command prompt, type the following commands to unbind a default syntax policy from a virtual server and verify the configuration:

- `unbind cs vserver <name> -policyName <policyName> [-priority <positiveInteger>] [-type REQUEST|RESPONSE]`
- `show lb vserver <name>`

### Example

```
unbind cs vserver vs-cont-switch -policyName poll
Done
> show cs vserver vs-cont-switch
vs-cont-switch (10.102.29.10:80) - HTTP Type: CONTENT
State: UP
Last state change was at Wed Aug 19 08:56:55 2009 (+18 ms)
Time since last state change: 0 days, 02:47:55.750
Client Idle Timeout: 180 sec
Down state flush: ENABLED
Disable Primary Vserver On Down : DISABLED
Port Rewrite : DISABLED
State Update: DISABLED
Default:          Content Precedence: RULE
Vserver IP and Port insertion: OFF
Case Sensitivity: ON
Push: DISABLED   Push VServer:
Push Label Rule: none
Done
```

The priority is required only for the “dummy” policy named NOPOLICY.

## To unbind an integrated caching, responder, rewrite, or compression default syntax policy globally by using the configuration utility

1. In the navigation pane, click the feature with the policy that you want to unbind (for example, Integrated Caching).
2. In the details pane, click <Feature Name> policy manager.
3. In the Policy Manager dialog box, select the bind point with the policy that you want to unbind, for example, Default Global.
4. Click the policy name that you want to unbind, and then click Unbind Policy.
5. Click Apply Changes.
6. Click Close. A message in the status bar indicates that the policy is unbound successfully.

## To unbind a DNS policy globally by using the configuration utility

1. Navigate to Traffic Management > DNS > Policies.
2. In the details pane, click Global Bindings.
3. In the Global Bindings dialog box, select policy and click unbind policy.
4. Click OK. A message in the status bar indicates that the policy is unbound successfully.

## To unbind a default syntax policy from a load balancing or content switching virtual server by using the configuration utility

1. Navigate to Traffic Management, and expand Load Balancing or Content Switching, and then click Virtual Servers.
2. In the details pane, double-click the virtual server from which you want to unbind the policy.
3. On the Policies tab, in the Active column, clear the check box next to the policy that you want to unbind.
4. Click OK. A message in the status bar indicates that the policy is unbound successfully.

## Creating Policy Labels

In addition to the built-in bind points where you set up policy banks, you can also configure user-defined policy labels and associate policies with them.

Within a policy label, you bind policies and specify the order of evaluation of each policy relative to others in the bank of policies for the policy label. The NetScaler also permits you to define an arbitrary evaluation order as follows:

- You can use `goto` expressions to point to the next entry in the bank to be evaluated after the current one.
- You can use an entry in a policy bank to invoke another bank.

This document includes the following details:

- [Creating Policy Labels](#)
- [Binding a Policy to a Policy Label](#)

## Creating Policy Labels

Updated: 2013-11-14

Each feature determines the type of policy that you can bind to a policy label, the type of load balancing virtual server that you can bind the label to, and the type of content switching virtual server from which the label can be invoked. For example, a TCP policy label can only be bound to a TCP load balancing virtual server. You cannot bind HTTP policies to a policy label of this type. And you can invoke a TCP policy label only from a TCP content switching virtual server.

After configuring a new policy label, you can invoke it from one or more banks for the built-in bind points.

### To create a caching policy label by using the command line interface

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

- `add cache policylabel <labelName> -evaluates req|res`
- `show cache policylabel<labelName>`

#### Example

```
> add cache policylabel lbl-cache-pol -evaluates req
Done

> show cache policylabel lbl-cache-pol
Label Name: lbl-cache-pol
Evaluates: REQ
Number of bound policies: 0
Number of times invoked: 0
Done
>
```

### To create a Content Switching policy label by using the command line interface

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

- `add cs policylabel <labelName> http|tcp|rtsp|ssl`
- `show cs policylabel <labelName>`

#### Example

```
> add cs policylabel lbl-cs-pol http
Done
> show cs policylabel lbl-cs-pol
Label Name: lbl-cs-pol
Label Type: HTTP
Number of bound policies: 0
Number of times invoked: 0
Done
```

## To create a Rewrite policy label by using the command line interface

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

- add rewrite policylabel <labelName> http\_req|http\_res|url|text|clientless\_vpn\_req|clientless\_vpn\_res
- show rewrite policylabel <labelName>

### Example

```
> add rewrite policylabel lbl-rewrt-pol http_req
Done

> show rewrite policylabel lbl-rewrt-pol
Label Name: lbl-rewrt-pol
Transform Name: http_req
Number of bound policies: 0
Number of times invoked: 0
Done
```

## To create a Responder policy label by using the command line interface

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

- add responder policylabel <labelName>
- show responder policylabel <labelName>

### Example

```
> add responder policylabel lbl-respndr-pol
Done

> show responder policylabel lbl-respndr-pol
Label Name: lbl-respndr-pol
Number of bound policies: 0
Number of times invoked: 0
Done
```

Note: Invoke this policy label from a policy bank. For more information, see ["Binding a Policy to a Policy Label."](#)

## To create a policy label by using the configuration utility

1. In the navigation pane, expand the feature for which you want to create a policy label, and then click Policy Labels. The choices are Integrated Caching, Rewrite, Content Switching, or Responder.
2. In the details pane, click Add.
3. In the Name box, enter a unique name for this policy label.
4. Enter feature-specific information for the policy label. For example, for Integrated Caching, in the Evaluates drop-down menu, you would select REQ if you want this policy label to contain request-time policies, or select RES if you want this policy label to contain response-time policies. For Rewrite, you would select a Transform name.
5. Click Create.
6. Configure one of the built-in policy banks to invoke this policy label. For more information, see ["Binding a Policy to a Policy Label."](#) A message in the status bar indicates that the policy label is created successfully.

## Binding a Policy to a Policy Label

As with policy banks that are bound to the built-in bind points, each entry in a policy label is a policy that is bound to the policy label. As with policies that are bound globally or to a vserver, each policy that is bound to the policy label can also invoke a policy bank or a policy label that is evaluated after the current entry has been processed. The following table summarizes the entries in a policy label.

### Name

The name of a policy, or, to invoke another policy bank without evaluating a policy, the "æœdummy" policy name NOPOLICY.

You can specify NOPOLICY more than once in a policy bank, but you can specify a named policy only once.

### Priority

An integer. This setting can work with the Goto expression.

### Goto Expression

Determines the next policy to evaluate in this bank. You can provide one of the following values:

#### 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:

The priority number of the next policy to be evaluated.

#### Numeric expression:

An expression that produces the priority number of the next policy to be evaluated.

The Goto can only proceed forward in a policy bank.

If you omit the Goto expression, it is the same as specifying END.

### Invocation Type

Designates a policy bank type. The value can be one of the following:

#### 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

The name of a virtual server or a policy label, depending on the value that you specified for the Invocation Type.

## Configuring a Policy Label or Virtual Server Policy Bank

After you have created policies, and created policy banks by binding the policies, you can perform additional configuration of policies within a label or policy bank. For example, before you configure invocation of an external policy bank, you might want to wait until you have configured that policy bank.

This document includes the following details:

- [Configuring a Policy Label](#)
- [Configuring a Policy Bank for a Virtual Server](#)

## Configuring a Policy Label

Updated: 2013-11-14

A policy label consists of a set of policies and invocations of other policy labels and virtual server-specific policy banks. An Invoke parameter enables you to invoke a policy label or a virtual server-specific policy bank from any other policy bank. A special-purpose NoPolicy entry enables you to invoke an external bank without processing an expression (a rule). The NoPolicy entry is a “dummy” policy that does not contain a rule.

For configuring policy labels from the NetScaler command line, note the following elaborations of the command syntax:

- gotoPriorityExpression is configured as described in ["Entries in a Policy Bank."](#)
- The type argument is required. This is unlike binding a conventional policy, where this argument is optional.
- You can invoke the bank of policies that are bound to a virtual server by using the same method as you use for invoking a policy label.

### To configure a policy label by using the command line interface

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

- `bind cache|rewrite|responder policylabel <policylabelName> -policyName <policyName> -priority <priority> [-gotoPriorityExpression <gotopriorityExpression>] [-invoke reqvserver|resvserver|policylabel <policyLabelName>|<vserverName>]`
- `show cache|rewrite|responder policylabel <policylabelName>`

#### Example

```
bind cache policylabel _reqBuiltinDefaults -policyName _nonGetReq -priority 100
Done
show cache policylabel _reqBuiltinDefaults
Label Name: _reqBuiltinDefaults
Evaluates: REQ
Number of bound policies: 3
Number of times invoked: 0
1) Policy Name: _nonGetReq
Priority: 100
GotoPriorityExpression: END
2) Policy Name: _advancedConditionalReq
Priority: 200
GotoPriorityExpression: END
3) Policy Name: _personalizedReq
Priority: 300
GotoPriorityExpression: END
Done
```

### To invoke a policy label from a Rewrite policy bank with a NOPOLICY entry by using the command line interface

At the command prompt, type the following commands to invoke a policy label from a Rewrite policy bank with a NOPOLICY entry and verify the configuration:

- o bind rewrite global <policyName> <priority> <gotoPriorityExpression> -type REQ\_OVERRIDE|REQ\_DEFAULT|RES\_OVERRIDE|RES\_DEFAULT -invoke reqvserver|resvserver|policylabel <policyLabelName>|<vserverName>
- o show rewrite global

### Example

```
> bind rewrite global NOPOLICY 100 -type REQ_DEFAULT -invoke policylabel lbl-rev
Done
> show rewrite global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

2)      Global bindpoint: REQ_OVERRIDE
        Number of bound policies: 1

Done
```

## To invoke a policy label from an Integrated Caching policy bank by using the command line interface

At the command prompt, type the following commands to invoke a policy label from an Integrated Caching policy bank and verify the configuration:

- o bind cache global NOPOLICY -priority <priority> -gotoPriorityExpression <gotopriorityExpression> -type REQ\_OVERRIDE|REQ\_DEFAULT|RES\_OVERRIDE|RES\_DEFAULT -invoke reqvserver|resvserver|policylabel <policyLabelName>|<vserverName>
- o show cache global

### Example

```
bind cache global NOPOLICY -priority 100 -gotoPriorityExpression END -type REQ_
Done
> show cache global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 2

2)      Global bindpoint: RES_DEFAULT
        Number of bound policies: 1

Done
```

## To invoke a policy label from a Responder policy bank by using the command line interface

At the command prompt, type the following commands to invoke a policy label from a Responder policy bank and verify the configuration:

- o bind responder global NOPOLICY <priority> <gotopriorityExpression> -type OVERRIDE|DEFAULT -invoke vserver|policylabel <policyLabelName>|<vserverName>
- o show responder global

### Example

```
> bind responder global NOPOLICY 100 NEXT -type DEFAULT -invoke policylabel lbl-
Done
> show responder global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 2

Done
```

## To configure a policy label by using the configuration utility

1. In the navigation pane, expand the feature for which you want to configure a policy label, and then click Policy Labels. The choices are Integrated Caching, Rewrite, or Responder.



2. In the details pane, double-click the label that you want to configure.
3. If you are adding a new policy to this policy label, click Insert Policy, and in the Policy Name field, select New Policy. For more information about adding a policy, see ["Creating or Modifying a Policy."](#) Note that if you are invoking a policy bank, and do not want a rule to be evaluated prior to the invocation, click Insert Policy, and in the Policy Name field select NOPOLICY.
4. For each entry in this policy label, configure the following:

Policy Name:

This is already determined by the Policy Name, new policy, or NOPOLICY entry that you inserted in this bank.

Priority:

A numeric value that determines either an absolute order of evaluation within the bank, or is used in conjunction with a Goto expression.

Expression:

The policy rule. Policy expressions are described in detail in the following chapters. For an introduction, see ["Configuring Default Syntax Expressions: Getting Started."](#)

Action:

The action to be taken if this policy evaluates to TRUE.

Goto Expression:

Optional. Used to augment the Priority level to determine the next policy or policy bank to evaluate. For more information on possible values for a Goto expression, see the table ["Entries in a Policy Bank."](#)

Invoke:

Optional. Invokes another policy bank.

5. Click Ok. A message in the status bar indicates that the policy label is configured successfully.

## Configuring a Policy Bank for a Virtual Server

Updated: 2013-09-02

You can configure a bank of policies for a virtual server. The policy bank can contain individual policies, and each entry in the policy bank can optionally invoke a policy label or a bank of policies that you configured for another virtual server. If you invoke a policy label or policy bank, you can do so without triggering an expression (a rule) by selecting a NOPOLICY entry instead of a policy name.

### To add policies to a virtual server policy bank by using the command line interface

At the command prompt, type the following commands to add policies to a virtual server policy bank and verify the configuration:

- `bind lb|cs vserver <virtualServerName> <serviceType> [-policyName <policyName>] [-priority <positiveInteger>] [-gotoPriorityExpression <expression>] [-type REQUEST|RESPONSE]`
- `show lb|cs vserver <virtualServerName>`

### Example

```
add lb vserver vs-cont-sw TCP
Done
show lb vserver vs-cont-sw
vs-cont-sw (0.0.0.0:0) - TCP      Type: ADDRESS
State: DOWN
Last state change was at Wed Aug 19 10:04:02 2009 (+279 ms)
Time since last state change: 0 days, 00:02:14.420
Effective State: DOWN
Client Idle Timeout: 9000 sec
Down state flush: ENABLED
Disable Primary Vserver On Down : DISABLED
No. of Bound Services :    0 (Total)        0 (Active)
Configured Method: LEASTCONNECTION
Mode: IP
Persistence: NONE
Connection Failover: DISABLED
Done
```

### To invoke a policy label from a virtual server policy bank with a NOPOLICY entry by using the command line interface

At the command prompt, type the following commands to invoke a policy label from a virtual server policy bank with a NOPOLICY entry and verify the configuration:

- o `bind lb|cs vserver <virtualServerName> -policyName NOPOLICY_REWRITE|NOPOLICY_CACHE|NOPOLICY_RESPONDER -priority <integer> -type REQUEST|RESPONSE -gotoPriorityExpression <gotopriorityExpression> -invoke reqVserver|resVserver|policyLabel <vserverName>|<labelName>`
- o `show lb vserver`

### Example

```
> bind lb vserver vs-cont-sw -policyname NOPOLICY-REWRITE -priority 200 -type REQUEST
Done
```

## To configure a virtual server policy bank by using the configuration utility

1. In the left navigation pane, expand Traffic Management > Load Balancing, Traffic Management > Content Switching, Traffic Management > SSL Offload, Security > AAA - Application Traffic, or NetScaler Gateway, as appropriate, and then click Virtual Servers.
2. In the details pane, select the virtual server that you want to configure, and then click Open.
3. In the Configure Virtual Server dialog box click the Policies tab.
4. To create a new policy in this bank, click the icon for the type of policy or policy label that you want to add to the virtual server's bank of policies, click Insert Policy. Note that if you want to invoke a policy label without evaluating a policy rule, select the NOPOLICY "dummy" policy.
5. To configure an existing entry in this policy bank, enter the following:

Priority:

A numeric value that determines either an absolute order of evaluation within the bank or is used in conjunction with a Goto expression.

Expression:

The policy rule. Policy expressions are described in detail in the following chapters. For an introduction, see "[Configuring Default Syntax Expressions: Getting Started](#)."

Action: on:

The action to be taken if this policy evaluates to TRUE.

Goto Expression:

Optional. Determines the next policy or policy bank to evaluate. For more information on possible values for a Goto expression, see "[Entries in a Policy Bank](#)."

Invoke:

Optional. To invoke another policy bank, select the name of the policy label or virtual server policy bank that you want to invoke.

6. When you are done, click OK. A message in the status bar indicates that the policy is configured successfully.

## Invoking or Removing a Policy Label or Virtual Server Policy Bank

Unlike a policy, which can only be bound once, you can use a policy label or a virtual server's policy bank any number of times by invoking it. Invocation can be performed from two places:

- From the binding for a named policy in a policy bank.
- From the binding for a NOPOLICY "dummy" entry in a policy bank.

Typically, the policy label must be of the same type as the policy from which it is invoked. For example, you would invoke a responder policy label from a responder policy.

Note: When binding or unbinding a global NOPOLICY entry in a policy bank at the command line, you specify a priority to distinguish one NOPOLICY entry from another.

## To invoke a rewrite or integrated caching policy label by using the command line interface

At the command prompt, type the one of the following commands to invoke a rewrite or integrated caching policy label and verify the configuration:

- **bind cache global** <policy> -priority <positive\_integer> [-gotoPriorityExpression <expression>] -type **REQ\_OVERRIDE|REQ\_DEFAULT|RES\_OVERRIDE|RES\_DEFAULT** -invoke reqvserver|resvserver|policylabel <label\_name>
- **bind rewrite global** <policy> -priority <positive\_integer> [-gotoPriorityExpression <expression>] -type **REQ\_OVERRIDE|REQ\_DEFAULT|RES\_OVERRIDE|RES\_DEFAULT** -invoke reqvserver|resvserver|policylabel <label\_name>
- **show cache global|show rewrite global**

### Example

```
> bind cache global _nonPostReq2 -priority 100 -type req_override -invoke
policylabel lbl-cache-pol
Done
> show cache global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 2

2)      Global bindpoint: RES_DEFAULT
        Number of bound policies: 1

3)      Global bindpoint: REQ_OVERRIDE
        Number of bound policies: 1

Done
```

## To invoke a responder policy label by using the command line interface

At the command prompt, type the following commands to invoke a responder policy label and verify the configuration:

- **bind responder global** <policy\_Name> <priority\_as\_positive\_integer> [-gotoPriorityExpression] -type **REQ\_OVERRIDE|REQ\_DEFAULT|OVERRIDE|DEFAULT** -invoke vserver|policylabel <label\_name>
- **show responder global**

### Example

```
> bind responder global pol404Error1 300 -invoke policylabel lbl-respndr-pol
Done
> show responder global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 2

Done
>
```

## To invoke a Virtual Server Policy Bank by using the command line interface

At the command prompt, type the following commands to invoke a Virtual Server Policy Bank and verify the configuration:

- o **bind lb vsrver** <vsrver\_name> **-policyName** <policy\_Name> **-priority** <positive\_integer> [**-gotoPriorityExpression** <expression>] **-type** REQUEST|RESPONSE **-invoke** reqvsrver|resvsrver|policylabel <policy\_Label\_Name>
- o **bind lb vsrver** <vsrver\_name>

### Example

```
> bind lb vsrver lbvip -policyName ns_cmp_msapp -priority 100
Done

> show lb vsrver lbvip
lbvip (8.7.6.6:80) - HTTP          Type: ADDRESS
State: DOWN
Last state change was at Wed Jul 15 05:54:24 2009 (+166 ms)
Time since last state change: 28 days, 06:37:49.250
Effective State: DOWN
Client Idle Timeout: 180 sec
Down state flush: ENABLED
Disable Primary Vserver On Down : DISABLED
Port Rewrite : DISABLED
No. of Bound Services : 0 (Total)          0 (Active)
Configured Method: LEASTCONNECTION
Mode: IP
Persistence: NONE
Vserver IP and Port insertion: OFF
Push: DISABLED  Push VServer:
Push Multi Clients: NO
Push Label Rule: none

1)      CSPolicy: pol-cont-sw   CSVserver: vs-cont-sw   Priority: 100   Hits: 0

1)      Policy : pol-ssl Priority:0
2)      Policy : ns_cmp_msapp Priority:100
3)      Policy : cf-pol Priority:1      Inherited
Done
>
```

## To remove a rewrite or integrated caching policy label by using the command line interface

At the command prompt, type one of the following commands to remove a rewrite or integrated caching policy label and verify the configuration:

- o **unbind rewrite global** <policyName> **-priority** <positiveInteger> **-type** REQ\_OVERRIDE|REQ\_DEFAULT|RES\_OVERRIDE|RES\_DEFAULT
- o **unbind cache global** <policyName> **-priority** <positiveInteger> **-type** REQ\_OVERRIDE|REQ\_DEFAULT|RES\_OVERRIDE|RES\_DEFAULT
- o **show rewrite global|show cache global**

### Example

```
> unbind rewrite global NOPOLICY -priority 100 -type REQ_OVERRIDE
Done
> show rewrite global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

Done
```

## To remove a responder policy label by using the command line interface

At the command prompt, type the following commands to remove a responder policy label and verify the configuration:

- `unbind responder global <policyName> -priority <positiveInteger> -type OVERRIDE|DEFAULT`
- `show responder global`

### Example

```
> unbind responder global NOPOLICY -priority 100 -type REQ_DEFAULT
Done
> show responder global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

Done
```

## To remove a Virtual Server policy label by using the command line interface

At the command prompt, type one of the following commands to remove a Virtual Server policy label and verify the configuration:

- `unbind lb vserver <virtualServerName> -policyName NOPOLICY-REWRITE|NOPOLICY-RESPONDER|NOPOLICY-CACHE -type REQUEST|RESPONSE -priority <positiveInteger>`
- `unbind cs vserver <virtualServerName> -policyName NOPOLICY-REWRITE|NOPOLICY-RESPONDER|NOPOLICY-CACHE -type REQUEST|RESPONSE -priority <positiveInteger>`
- `show lb vserver|show cs vserver`

### Example

```
> unbind lb vserver lbvip -policyName ns_cmp_msapp -priority 200
Done
> show lb vserver lbvip
    lbvip (8.7.6.6:80) - HTTP          Type: ADDRESS
    State: DOWN
    Last state change was at Wed Jul 15 05:54:24 2009 (+161 ms)
    Time since last state change: 28 days, 06:47:54.600
    Effective State: DOWN
    Client Idle Timeout: 180 sec
    Down state flush: ENABLED
    Disable Primary Vserver On Down : DISABLED
    Port Rewrite : DISABLED
    No. of Bound Services :    0 (Total)          0 (Active)
    Configured Method: LEASTCONNECTION
    Mode: IP
    Persistence: NONE
    Vserver IP and Port insertion: OFF
    Push: DISABLED  Push VServer:
    Push Multi Clients: NO
    Push Label Rule: none

1)      CSPolicy: pol-cont-sw   CSVserver: vs-cont-sw   Priority: 100   Hits: 0

1)      Policy : pol-ssl Priority:0
2)      Policy : cf-pol Priority:1      Inherited

Done
```

## To invoke a policy label or virtual server policy bank by using the configuration utility

1. Bind a policy, as described in "Binding a Policy Globally", "Binding a Policy to a Virtual Server", or "Binding a Policy to a Policy Label." Alternatively, you can enter a NOPOLICY "dummy" entry instead of a policy name. You do this if you do not want to evaluate a policy before evaluating the policy bank.

2. In the Invoke field, select the name of the policy label or virtual server policy bank that you want to evaluate if traffic matches the bound policy. A message in the status bar indicates that the policy label or virtual server policy bank is invoked successfully.

## **To remove a policy label invocation by using the configuration utility**

1. Open the policy and clear the Invoke field. Unbinding the policy also removes the invocation of the label. A message in the status bar indicates that the policy label is removed successfully.

## Configuring and Binding Policies with the Policy Manager

Some applications provide a specialized Policy Manager in the NetScaler configuration utility to simplify configuring policy banks. It also lets you find and delete policies and actions that are not being used.

The Policy Manager is currently available for the Rewrite, Integrated Caching, Responder, and Compression features. The following are keyboard equivalents for the procedures in this section:

- For editing a cell in the Policy Manager, you can tab to the cell and click `F2` or press the `SPACE` bar on the keyboard.
- To select an entry in a drop-down menu, you can tab to the entry, press the `space` bar to view the drop-down menu, use the `UP` and `DOWN` `ARROW` keys to navigate to the entry that you want, and press the `space` bar again to select the entry.
- To cancel a selection in a drop-down menu, press the `Escape` key.
- To insert a policy, tab to the row above the insertion point and press `Control + Insert`, or click `Insert Policy`.
- To remove a policy, tab to the row that contains the policy and press `Delete`.

Note: Note that when you delete the policy, the NetScaler searches the Goto Expression values of other policies in the bank. If any of these Goto Expression values match the priority level of the deleted policy, they are removed.

## To configure policy bindings by using the Policy Manager

1. In the navigation pane, click the feature for which you want to configure policies. The choices are Responder, Integrated Caching, Rewrite or Compression.
2. In the details pane, click Policy Manager.
3. If you are configuring classic policy bindings for compression, in the Compression Policy Manager dialog box, click **Switch to Classic Syntax**. The dialog box switches to the classic syntax view and displays the Switch to Default Syntax button. At any time before you complete configuring policy bindings, if you want to configure bindings for policies that use the default syntax, click the Switch to Default Syntax button.
4. For features other than Responder, to specify the bind point, click Request or Response, and then click one of the request-time or response-time bind points. The options are Override Global, LB Virtual Server, CS Virtual Server, Default Global, or Policy Label. If you are configuring the Responder, the Request and Response flow types are not available.
5. To bind a policy to this bind point, click Insert Policy, and select a previously configured policy, a NOPOLICY label, or the New policy option. Depending on the option that you select, you have the following choices:
  - **New policy:** Create the policy as described in "Creating or Modifying a Policy," and then configure the priority level, GoTo expression, and policy invocation as described in the table, "Format of Each Entry in a Policy Bank."
  - **Existing policy, NOPOLICY, or NOPOLICY<feature name>:** Configure the priority level, GoTo expression, and policy invocation as described in the table, "Format of Each Entry in a Policy Bank." The **NOPOLICY** or **NOPOLICY<feature name>** options are available only for policies that use default syntax expressions.
6. Repeat the preceding steps to add entries to this policy bank.
7. To modify the priority level for an entry, you can do any of the following:
  - Double-click the Priority field for an entry and edit the value.
  - Click and drag a policy to another row in the table.
  - Click Regenerate Priorities.

In all three cases, priority levels of all other policies are modified as needed to accommodate the new value. Goto Expressions with integer values are also updated automatically. For example, if you change a priority value of 10 to 100, all policies with a Goto Expression value of 10 are updated to the value 100.

8. To change the policy, action, or policy bank invocation for an row in the table, click the down arrow to the right of the entry and do one of the following:
  - To change the policy, select another policy name or select New Policy and follow the steps in "Creating or Modifying a Policy."
  - To change the Goto Expression, select Next, End, USE\_INVOCATION\_RESULT, or select more and enter an expression whose result returns the priority level of another entry in this policy bank.
  - To modify an invocation, select an existing policy bank, or click New Policy Label and follow the steps in "Binding a Policy to a Policy Label."
9. To unbind a policy or a policy label invocation from this bank, click any field in the row that contains the policy or policy label, and then click Unbind Policy.
10. When you are done, click Apply Changes. A message in the status bar indicates that the policy is bound successfully.

## To remove unused policies by using the Policy Manager

1. In the navigation pane, click the feature for which you want to configure the policy bank. The choices are Responder, Integrated Caching, or Rewrite.
2. In the details pane, click <Feature Name> policy manager.
3. In the <Feature Name> Policy Manager dialog box, click Cleanup Configuration.
4. In the Cleanup Configuration dialog box, select the items that you want to delete, and then click Remove.
5. In the Remove dialog box, click Yes.
6. Click Close. A message in the status bar indicates that the policy is removed successfully.



## Configuring Default Syntax Expressions: Getting Started

Default syntax policies evaluate data on the basis of information that you supply in default syntax expressions. A default syntax expression analyzes data elements (for example, HTTP headers, source IP addresses, the NetScaler system time, and POST body data). In addition to configuring a default syntax expression in a policy, in some NetScaler features you configure default syntax expressions outside of the context of a policy.

To create a default syntax expression, you select a prefix that identifies a piece of data that you want to analyze, and then you specify an operation to perform on the data. For example, an operation can match a piece of data with a text string that you specify, or it can transform a text string into an HTTP header. Other operations match a returned string with a set of strings or a string pattern. You configure compound expressions by specifying Boolean and arithmetic operators, and by using parentheses to control the order of evaluation.

Default syntax expressions can also contain classic expressions. You can assign a name to a frequently used expression to avoid having to build the expression repeatedly.

Policies and a few other entities include rules that the NetScaler uses to evaluate a packet in the traffic flowing through it, to extract data from the NetScaler system itself, to send a request (a “callout”) to an external application, or to analyze another piece of data. A rule takes the form of a logical expression that is compared against traffic and ultimately returns values of TRUE or FALSE.

The elements of the rule can themselves return TRUE or FALSE, string, or numeric values.

Before configuring a default syntax expression, you need to understand the characteristics of the data that the policy or other entity is to evaluate. For example, when working with the Integrated Caching feature, a policy determines what data can be stored in the cache. With Integrated Caching, you need to know the URLs, headers, and other data in the HTTP requests and responses that the NetScaler receives. With this knowledge, you can configure policies that match the actual data and enable the NetScaler to manage caching for HTTP traffic. This information helps you determine the type of expression that you need to configure in the policy.

## Basic Elements of a Default Syntax Expression

A default syntax expression consists of, at a minimum, a prefix (or a single element used in place of a prefix). Most expressions also specify an operation to be performed on the data that the prefix identifies. You format an expression of up to 1,499 characters as follows:

```
<prefix>.<operation> [<compound-operator> <prefix>.<operation> . . .]
```

where

**<prefix>**

is an anchor point for starting an expression.

The prefix is a period-delimited key that identifies a unit of data. For example, the following prefix examines HTTP requests for the presence of a header named Content-Type:

```
http.req.header( "Content-Type" )
```

Prefixes can also be used on their own to return the value of the object that the prefix identifies.

**<operation>**

identifies an evaluation that is to be performed on the data identified by the prefix.

For example, consider the following expression:

```
http.req.header( "Content-Type" ).eq( "text/html" )
```

In this expression, the following is the operator component:

```
eq( "text/html" )
```

This operator causes the NetScaler to evaluate any HTTP requests that contain a Content-Type header, and in particular, to determine if the value of this header is equal to the string "text/html". For more information, see "Operations."

**<compound-operator>**

is a Boolean or arithmetic operator that forms a compound expression from multiple prefix or prefix.operation elements.

For example, consider the following expression:

```
http.req.header( "Content-Type" ).eq( "text/html" ) && http.req.url.contains( ".html" )
```

This document includes the following details:

- [Prefixes](#)
- [Single-Element Expressions](#)
- [Operations](#)
- [Basic Operations on Expression Prefixes](#)

## Prefixes

Updated: 2013-09-30

An expression prefix represents a discrete piece of data. For example, an expression prefix can represent an HTTP URL, an HTTP Cookie header, or a string in the body of an HTTP POST request. An expression prefix can identify and return a wide variety of data types, including the following:

- A client IP address in a TCP/IP packet
- NetScaler system time
- An external callout over HTTP
- A TCP or UDP record type

In most cases, an expression prefix begins with one of the following keywords:

**CLIENT:**

Identifies a characteristic of the client that is either sending a request or receiving a response, as in the following examples:

- The prefix `client.ip.dst` designates the destination IP address in the request or response.

- o The prefix `client.ip.src` designates the source IP address.

#### HTTP:

Identifies an element in an HTTP request or a response, as in the following examples:

- o The prefix `http.req.body(integer)` designates the body of the HTTP request as a multiline text object, up to the character position designated in `integer`.
- o The prefix `http.req.header("header_name")` designates an HTTP header, as specified in `header_name`.
- o The prefix `http.req.url` designates an HTTP URL in URL-encoded format.

#### SERVER:

Identifies an element in the server that is either processing a request or sending a response.

#### SYS:

Identifies a characteristic of the NetScaler that is processing the traffic.

Note: Note that DNS policies support only SYS, CLIENT, and SERVER objects.

In addition, in the NetScaler Gateway, the Clientless VPN function can use the following types of prefixes:

#### TEXT:

Identifies any text element in a request or a response.

#### TARGET:

Identifies the target of a connection.

#### URL:

Identifies an element in the URL portion of an HTTP request or response.

As a general rule of thumb, any expression prefix can be a self-contained expression. For example, the following prefix is a complete expression that returns the contents of the HTTP header specified in the string argument (enclosed in quotation marks):

```
http.res.header("myheader")
```

Or you can combine prefixes with simple operations to determine TRUE and FALSE values. For example, the following returns a value of TRUE or FALSE:

```
http.res.header("myheader").exists
```

You can also use complex operations on individual prefixes and multiple prefixes within an expression, as in the following example:

```
http.req.url.length + http.req.cookie.length <= 500
```

Which expression prefixes you can specify depends on the NetScaler feature. The following table describes the expression prefixes that are of interest on a per-feature basis

Table 1. Permitted Types of Expression Prefixes in Various NetScaler Features

Feature	Types of Expression Prefix Used in the Feature
DNS	SYS, CLIENT, SERVER
Responder in Protection Features	HTTP, SYS, CLIENT
Content Switching	HTTP, SYS, CLIENT
Rewrite	HTTP, SYS, CLIENT, SERVER, URL, TEXT, TARGET, VPN
Integrated Caching	HTTP, SYS, CLIENT, SERVER
NetScaler Gateway, Clientless Access	HTTP, SYS, CLIENT, SERVER, URL, TEXT, TARGET, VPN

Note: For details on the permitted expression prefixes in a feature, see the documentation for that feature.

## Single-Element Expressions

The simplest type of default syntax expression contains a single element. This element can be one of the following:

- `true`. A default syntax expression can consist simply of the value `true`. This type of expression always returns a value of `TRUE`. It is useful for chaining policy actions and triggering Goto expressions.
- `false`. A default syntax expression can consist simply of the value `false`. This type of expression always returns a value of `FALSE`.
- A prefix for a compound expression. For example, the prefix `HTTP.REQ.HOSTNAME` is a complete expression that returns a host name and `HTTP.REQ.URL` is a complete expression that returns a URL. The prefix could also be used in conjunction with operations and additional prefixes to form a compound expression.

## Operations

In most expressions, you also specify an operation on the data that the prefix identifies. For example, suppose that you specify the following prefix:

```
http.req.url
```

This prefix extracts URLs in HTTP requests. This expression prefix does not require any operators to be used in an expression. However, when you configure an expression that processes HTTP request URLs, you can specify operations that analyze particular characteristics of the URL. Following are a few possibilities:

- Search for a particular host name in the URL.
- Search for a particular path in the URL.
- Evaluate the length of the URL.
- Search for a string in the URL that indicates a time stamp and convert it to GMT.

The following is an example of a prefix that identifies an HTTP header named `Server` and an operation that searches for the string `IIS` in the header value:

```
http.res.header("Server").contains("IIS")
```

Following is an example of a prefix that identifies host names and an operation that searches for the string `www.mycompany.com` as the value of the name:

```
http.req.hostname.eq("www.mycompany.com")
```

## Basic Operations on Expression Prefixes

The following table describes a few of the basic operations that can be performed on expression prefixes.

Table 2. Basic Operations for Expressions

Operation	Determines Whether or Not
<code>CONTAINS(&lt;string&gt;)</code>	The object matches <code>&lt;string&gt;</code> . Following is an example:  <code>http.req.header("Cache-Control").contains("no-cache")</code>
<code>EXISTS</code>	A particular item is present in an object. Following is an example:  <code>http.res.header("MyHdr").exists</code>
<code>EQ(&lt;text&gt;)</code>	A particular non-numeric value is present in an object. Following is an example:  <code>http.req.method.eq(post)</code>
<code>EQ(&lt;integer&gt;)</code>	A particular numeric value is present in an object. Following is an example:  <code>client.ip.dst.eq(10.100.10.100)</code>
<code>LT(&lt;integer&gt;)</code>	An object's value is less than a particular value. Following is an example:

	<code>http.req.content_length.lt(5000)</code>
<code>GT(&lt;integer&gt;)</code>	<p>An object's value is greater than a particular value. Following is an example:</p> <p><code>http.req.content_length.gt(5)</code></p>

The following table summarizes a few of the available types of operations.

Table 3. Basic Types of Operations

Operation Type	Description
Text operations	<p>Match individual strings and sets of strings with any portion of a target. The target can be an entire string, the start of a string, or any portion of text in between the start and the end of the string.</p> <p>For example, you can extract the string "XYZ" from "XYZSomeText". Or, you can compare an HTTP header value with an array of different strings.</p> <p>You can also transform text into another type of data. Following are examples:</p> <ul style="list-style-type: none"> <li>• Transform a string into an integer value</li> <li>• Create a list from the query strings in a URL</li> <li>• Transform a string into a time value</li> </ul>
Numeric operations	<p>Numeric operations include applying arithmetic operators, evaluating content length, the number of items in a list, dates, times, and IP addresses.</p>

## Compound Default Syntax Expressions

You can configure a default syntax expression that contains Boolean or arithmetic operators and multiple atomic operations. The following compound expression contains a boolean AND:

```
http.req.hostname.eq("mycompany.com") && http.req.method.eq(post)
```

The following expression adds the value of two targets, and compares the result to a third value:

```
http.req.url.length + http.req.cookie.length <= 500
```

A compound expression can contain any number of logical and arithmetic operators. The following expression evaluates the length of an HTTP request on the basis of its URL and cookie, evaluates text in the header, and performs a Boolean AND on these two results:

```
http.req.url.length + http.req.cookie.length <= 500 && http.req.header.contains("some text")
```

You can use parentheses to control the order of evaluation in a compound expression.

This document includes the following details:

- [Booleans in Compound Expressions](#)
- [Parentheses in Compound Expressions](#)
- [Compound Operations for Strings](#)
- [Compound Operations for Numbers](#)

## Booleans in Compound Expressions

You configure compound expressions with the following operators:

- &&.**  
This operator is a logical AND. For the expression to evaluate to TRUE, all components that are joined by the And must evaluate to TRUE. Following is an example:  

```
http.req.url.hostname.eq("myHost") && http.req.header("myHeader").exists
```
- ||.**  
This operator is a logical OR. If any component of the expression that is joined by the OR evaluates to TRUE, the entire expression is TRUE.
- !.**  
Performs a logical NOT on the expression.

In some cases, the NetScaler configuration utility offers AND, NOT, and OR operators in the Add Expression dialog box. However, these are of limited use. Citrix recommends that you use the operators &&, ||, and ! to configure compound expressions that use Boolean logic.

## Parentheses in Compound Expressions

You can use parentheses to control the order of evaluation of an expression. The following is an example:

```
http.req.url.contains("myCompany.com") || (http.req.url.hostname.eq("myHost") && http.req.header("myHeader").exists)
```

The following is another example:

```
(http.req.header("Content-Type").exists && http.req.header("Content-Type").eq("text/html")) || (http.req.header("Transfer-Encoding").exists || http.req.header("Content-Length").exists)
```

## Compound Operations for Strings

The following table describes operators that you can use to configure compound operations on string data.

Table 1. String-Based Operations for Compound Default Syntax Expressions

All string operations
-----------------------

### Operations that produce a string value

str + str	Concatenates the value of the expression on the left of the operator with the value on the right. Following is an example:  <code>http.req.hostname + http.req.url.protocol</code>
str + num	Concatenates the value of the expression on the left of the operator with a numeric value on the right. Following is an example:  <code>http.req.hostname + http.req.url.content_length</code>
num + str	Concatenates the numeric value of the expression on the left side of the operator with a string value on the right. Following is an example:  <code>http.req.url.content_length + http.req.url.hostname</code>
str + ip	Concatenates the string value of the expression on the left side of the operator with an IP address value on the right. Following is an example:  <code>http.req.hostname + 10.00.000.00</code>
ip + str	Concatenates the IP address value of the expression on the left of the operator with a string value on the right. Following is an example:  <code>client.ip.dst + http.req.url.hostname</code>
str1 ALT str2	Uses the string1 or string2 value that is derived from the expression on either side of the operator, as long as neither of these expressions is a compound expressions. Following is an example:  <code>http.req.hostname alt client.ip.src</code>

### Operations on strings that produce a result of TRUE or FALSE

str == str	Evaluates whether the strings on either side of the operator are the same. Following is an example:  <code>http.req.header("myheader") == http.res.header("myheader")</code>
str <= str	Evaluates whether the string on the left side of the operator is the same as the string on the right, or precedes it alphabetically.
str >= str	Evaluates whether the string on the left side of the operator is the same as the string on the right, or follows it alphabetically.
str < str	Evaluates whether the string on the left side of the operator precedes the string on the right alphabetically.
str > str	Evaluates whether the string on the left side of the operator follows the string on the right alphabetically.
str !! = str	Evaluates whether the strings on either side of the operator are different.

### Logical operations on strings

bool && bool	<p>This operator is a logical AND. When evaluating the components of the compound expression, all components that are joined by the AND must evaluate to TRUE. Following is an example:</p> <pre>http.req.method.eq(GET) &amp;&amp; http.req.url.query.contains("viewReport &amp;&amp; my_pagelabel")</pre>
bool    bool	<p>This operator is a logical OR. When evaluating the components of the compound expression, if any component of the expression that is joined by the OR evaluates to TRUE, the entire expression is TRUE. Following is an example:</p> <pre>http.req.url.contains(".js")    http.res.header("Content-Type").contains("javascript")</pre>
! bool	<p>Performs a logical NOT on the expression.</p>

## Compound Operations for Numbers

Updated: 2013-09-02

You can configure compound numeric expressions. For example, the following expression returns a numeric value that is the sum of an HTTP header length and a URL length:

```
http.req.header.length + http.req.url.length
```

The following tables describes operators that you can use to configure compound expressions for numeric data.

Table 2. Arithmetic Operations on Numbers

Operator	Description
num + num	<p>Add the value of the expression on the left of the operator to the value of the expression on the right. Following is an example:</p> <pre>http.req.content_length + http.req.url.length</pre>
num - num	<p>Subtract the value of the expression on the right of the operator from the value of the expression on the left.</p>
num * num	<p>Multiply the value of the expression on the left of the operator with the value of the expression on the right. Following is an example:</p> <pre>client.interface.rxthroughput * 9</pre>
num / num	<p>Divide the value of the expression on the left of the operator by the value of the expression on the right.</p>
num % num	<p>Calculate the modulo, or the numeric remainder on a division of the value of the expression on the left of the operator by the value of the expression on the right.</p> <p>For example, the values "15 mod 4" equals 3, and "12 mod 4" equals 0.</p>
~number	<p>Returns a number after applying a bitwise logical negation of the number. The following example assumes that numeric.expression returns 12 (binary 1100):</p> <pre>~numeric.expression.</pre> <p>The result of applying the ~ operator is -11 (a binary 1110011, 32 bits total with all ones to the left).</p> <p>Note that all returned values of less than 32 bits before applying the operator implicitly have zeros to the left to make them 32 bits wide.</p>



number ^ number	<p>Compares two bit patterns of equal length and performs an XOR operation on each pair of corresponding bits in each number argument, returning 1 if the bits are different, and 0 if they are the same.</p> <p>Returns a number after applying a bitwise XOR to the integer argument and the current number value. If the values in the bitwise comparison are the same, the returned value is a 0. The following example assumes that numeric.expression1 returns 12 (binary 1100) and numeric.expression2 returns 10 (binary 1010):</p> <pre>numeric.expression1 ^ numeric.expression2</pre> <p>The result of applying the ^ operator to the entire expression is 6 (binary 0110).</p> <p>Note that all returned values of less than 32 bits before applying the operator implicitly have zeros to the left to make them 32 bits wide.</p>
number   number	<p>Returns a number after applying a bitwise OR to the number values. If either value in the bitwise comparison is a 1, the returned value is a 1. The following example assumes that numeric.expression1 returns 12 (binary 1100) and numeric.expression2 returns 10 (binary 1010):</p> <pre>numeric.expression1   numeric.expression2</pre> <p>The result of applying the   operator to the entire expression is 14 (binary 1110).</p> <p>Note that all returned values of less than 32 bits before applying the operator implicitly have zeros to the left to make them 32 bits wide.</p>
number & number	<p>Compares two bit patterns of equal length and performs a bitwise AND operation on each pair of corresponding bits, returning 1 if both of the bits contains a value of 1, and 0 if either bits are 0.</p> <p>The following example assumes that numeric.expression1 returns 12 (binary 1100) and numeric.expression2 returns 10 (binary 1010):</p> <pre>numeric.expression1 &amp; numeric.expression2</pre> <p>The whole expression evaluates to 8 (binary 1000).</p> <p>Note that all returned values of less than 32 bits before applying the operator implicitly have zeros to the left to make them 32 bits wide.</p>
num << num	<p>Returns a number after a bitwise left shift of the number value by the right-side number argument number of bits.</p> <p>Note that the number of bits shifted is integer modulo 32. The following example assumes that numeric.expression1 returns 12 (binary 1100) and numeric.expression2 returns 3:</p> <pre>numeric.expression1 &lt;&lt; numeric.expression2</pre> <p>The result of applying the LSHIFT operator is 96 (a binary 1100000).</p> <p>Note that all returned values of less than 32 bits before applying the operator implicitly have zeros to the left to make them 32 bits wide.</p>
num >> num	<p>Returns a number after a bitwise right shift of the number value by the integer argument number of bits.</p> <p>Note that the number of bits shifted is integer modulo 32. The following example assumes that numeric.expression1 returns 12 (binary 1100) and numeric.expression2 returns 3:</p> <pre>numeric.expression1 &gt;&gt; numeric.expression2</pre> <p>The result of applying the RSHIFT operator is 1 (a binary 0001).</p>

Note that all returned values of less than 32 bits before applying the operator implicitly have zeros to the left to make them 32 bits wide.

Table 3. Numeric Operators That Produce a Result of TRUE or FALSE

Operator	Description
num == num	Determine if the value of the expression on the left of the operator is equal to the value of the expression on the right.
num != num	Determine if the value of the expression on the left of the operator is not equal to the value of the expression on the right.
num > num	Determine if the value of the expression on the left of the operator is greater than the value of the expression on the right.
num < num	Determine if the value of the expression on the left of the operator is less than the value of the expression on the right.
num >= num	Determine if the value of the expression on the left of the operator is greater than or equal to the value of the expression on the right.
num <= num	Determine if the value of the expression on the left of the operator is less than or equal to the value of the expression on the right.

## Functions for Data Types in the Policy Infrastructure

The NetScaler policy infrastructure supports the following numeric data types:

- Integer (32 bits)
- Unsigned long (64 bits)
- Double (64 bits)

Simple expressions can return all of these data types. Therefore, you can create compound expressions that use arithmetic operators and logical operators to evaluate or return values of these data types. Additionally, you can use all of these values in policy expressions. Literal constants of type unsigned long can be specified by appending the string `u1` to the number. Literal constants of type double contain a period (`.`), an exponent, or both.

### Arithmetic Operators, Logical Operators, and Type Promotion

In compound expressions, the following standard arithmetic and logical operators can be used for the double and unsigned long data types:

- `+`, `-`, `*`, and `/`
- `%`, `~`, `^`, `&`, `|`, `<<`, and `>>` (do not apply to double)
- `==`, `!=`, `>`, `<`, `>=`, and `<=`

All of these operators have the same meaning as in the C programming language.

In all cases of mixed operations between operands of type integer, unsigned long, and double, type promotion is performed so that the operation can be performed on operands of the same type. A type of lower precedence is automatically promoted to the type of the operand with the highest precedence involved in the operation. The order of precedence (higher to lower) is as follows:

- Double
- Unsigned long
- Integer

Therefore, an operation that returns a numeric result returns a result of the highest type involved in the operation.

For example, if the operands are of type integer and unsigned long, the integer operand is automatically converted to type unsigned long. This type conversion is performed even in simple expressions in which the type of data identified by the expression prefix does not match the type of data that is passed as the argument to the function. To illustrate such an example, in the operation `HTTP.REQ.CONTENT_LENGTH.DIV(3ul)`, the integer returned by the prefix `HTTP.REQ.CONTENT_LENGTH` is automatically converted to unsigned long (the type of the data passed as the argument to the `DIV()` function), and an unsigned long division is performed. Similarly, the argument can be promoted in an expression. For example, `HTTP.REQ.HEADER("myHeader").TYPECAST_DOUBLE_AT.DIV(5)` promotes the integer 5 to type double and performs double-precision division.

The following table describes the arithmetic and Boolean functions that can be used with the integer, unsigned long, and double data types. For information about expressions for casting data of one type to data of another type, see "[Typecasting Data](#)."

## Specifying the Character Set in Expressions

The policy infrastructure on the Citrix® NetScaler® appliance supports the ASCII and UTF-8 character sets. The default character set is ASCII. If the traffic for which you are configuring an expression consists of only ASCII characters, you need not specify the character set in the expression. However, you must specify the character set in every simple expression that is meant for UTF-8 traffic. To specify the UTF-8 character set in a simple expression, you must include the `SET_CHAR_SET(<charset>)` function, with `<charset>` specified as `UTF_8`, as shown in the following examples:

```
HTTP.REQ.BODY(10).SET_CHAR_SET(UTF_8).CONTAINS("ÃŸ")
```

```
HTTP.RES.BODY(100).SET_CHAR_SET(UTF_8).BEFORE_STR("BÃ¼cher").AFTER_STR("WÃ¼rterbuch")
```

In an expression, the `SET_CHAR_SET()` function must be introduced at the point in the expression after which data processing must be carried out in the specified character set. For example, in the expression `HTTP.REQ.BODY(1000).AFTER_REGEX(re/following example/).BEFORE_REGEX(re/In the preceding example/).CONTAINS_ANY("Greek_ alphabet")`, if the strings stored in the pattern set "Greek\_alphabet" are in UTF-8, you must include the `SET_CHAR_SET(UTF_8)` function immediately before the `CONTAINS_ANY("<string>")` function, as follows:

```
HTTP.REQ.BODY(1000).AFTER_REGEX(re/following example/).BEFORE_REGEX(re/In the preceding example/).SET_CHAR_SET(UTF_8).CONTAINS_ANY("Greek_ alphabet")
```

The `SET_CHAR_SET()` function sets the character set for all further processing (that is, for all subsequent functions) in the expression unless it is overridden later in the expression by another `SET_CHAR_SET()` function that changes the character set. Therefore, if all the functions in a given simple expression are intended for UTF-8, you can include the `SET_CHAR_SET(UTF_8)` function immediately after functions that identify text (for example, the `HEADER("<name>")` or `BODY(<int>)` functions). In the second example that follows the first paragraph above, if the ASCII arguments passed to the `AFTER_REGEX()` and `BEFORE_REGEX()` functions are changed to UTF-8 strings, you can include the `SET_CHAR_SET(UTF_8)` function immediately after the `BODY(1000)` function, as follows:

```
HTTP.REQ.BODY(1000).SET_CHAR_SET(UTF_8).AFTER_REGEX(re/BÃ¼cher/).BEFORE_REGEX(re/WÃ¼rterbuch/).CONTAINS_ANY("Greek_alphabet")
```

The UTF-8 character set is a superset of the ASCII character set, so expressions configured for the ASCII character set continue to work as expected if you change the character set to UTF-8.

## Compound Expressions with Different Character Sets

In a compound expression, if one subset of expressions is configured to work with data in the ASCII character set and the rest of the expressions are configured to work with data in the UTF-8 character set, the character set specified for each individual expression is considered when the expressions are evaluated individually. However, when processing the compound expression, just before processing the operators, the appliance promotes the character set of the returned ASCII values to UTF-8. For example, in the following compound expression, the first simple expression evaluates data in the ASCII character set while the second simple expression evaluates data in the UTF-8 character set:

```
HTTP.REQ.HEADER("MyHeader") == HTTP.REQ.BODY(10).SET_CHAR_SET(UTF_8)
```

However, when processing the compound expression, just before evaluating the "is equal to" Boolean operator, the NetScaler appliance promotes the character set of the value returned by `HTTP.REQ.HEADER("MyHeader")` to UTF-8.

The first simple expression in the following example evaluates data in the ASCII character set. However, when the NetScaler appliance processes the compound expression, just before concatenating the results of the two simple expressions, the appliance promotes the character set of the value returned by `HTTP.REQ.BODY(10)` to UTF-8.

```
HTTP.REQ.BODY(10) + HTTP.REQ.HEADER("MyHeader").SET_CHAR_SET(UTF_8)
```

Consequently, the compound expression returns data in the UTF-8 character set.

## Specifying the Character Set Based on the Character Set of Traffic

You can set the character set to UTF-8 on the basis of traffic characteristics. If you are not sure whether the character set of the traffic being evaluated is UTF-8, you can configure a compound expression in which the first expression checks for UTF-8 traffic and subsequent expressions set the character set to UTF-8. Following is an example of a compound expression that first checks the value of "charset" in the request's Content-Type header for "UTF-8" before checking whether the first 1000 bytes in the request contain the UTF-8 string `BÃ¼cher`:

```
HTTP.REQ.HEADER("Content-Type").SET_TEXT_MODE(IGNORECASE).TYPECAST_NVLIST_T('=', ' ; ', ' ' ).VALUE("charset").EQ("UTF-8") && HTTP.REQ.BODY(1000).SET_CHAR_SET(UTF_8).CONTAINS("BÃ¼cher")
```

If you are sure that the character set of the traffic being evaluated is UTF-8, the second expression in the example is sufficient.

## Character and String Literals in Expressions

During expression evaluation, even if the current character set is ASCII, character literals and string literals, which are enclosed in single quotation marks (') and quotation marks (""), respectively, are considered to be literals in the UTF-8 character set. In a given expression, if a function is operating on character or string literals in the ASCII character set and you include a non-ASCII character in the literal, an error is returned.

## Values in Hexadecimal and Octal Formats

When configuring an expression, you can enter values in octal and hexadecimal formats. However, each hexadecimal or octal byte is considered a UTF-8 byte. Invalid UTF-8 bytes result in errors regardless of whether the value is entered manually or pasted from the clipboard. For example, "\xce\x20" is an invalid UTF-8 character because "c8" cannot be followed by "20" (each byte in a multi-byte UTF-8 string must have the high bit set). Another example of an invalid UTF-8 character is "\xce \xa9," since the hexadecimal characters are separated by a white-space character.

## Functions That Return UTF-8 Strings

Only the <text>.XPATH and <text>.XPATH\_JSON functions always return UTF-8 strings. The following MySQL routines determine at runtime which character set to return, depending on the data in the protocol:

- MYSQL\_CLIENT\_T.USER
- MYSQL\_CLIENT\_T.DATABASE
- MYSQL\_REQ\_QUERY\_T.COMMAND
- MYSQL\_REQ\_QUERY\_T.TEXT
- MYSQL\_REQ\_QUERY\_T.TEXT(<unsigned int>)
- MYSQL\_RES\_ERROR\_T.SQLSTATE
- MYSQL\_RES\_ERROR\_T.MESSAGE
- MYSQL\_RES\_FIELD\_T.CATALOG
- MYSQL\_RES\_FIELD\_T.DB
- MYSQL\_RES\_FIELD\_T.TABLE
- MYSQL\_RES\_FIELD\_T.ORIGINAL\_TABLE
- MYSQL\_RES\_FIELD\_T.NAME
- MYSQL\_RES\_FIELD\_T.ORIGINAL\_NAME
- MYSQL\_RES\_OK\_T.MESSAGE
- MYSQL\_RES\_ROW\_T.TEXT\_ELEM(<unsigned int>)

## Terminal Connection Settings for UTF-8

When you set up a connection to the NetScaler appliance by using a terminal connection (by using PuTTY, for example), you must set the character set for transmission of data to UTF-8.

## Classic Expressions in Default Syntax Expressions

Classic expressions describe basic characteristics of traffic. In some cases, you may want to use a classic expression in a default syntax expression. You can do so with the default syntax expression configuration tool. This can be helpful when manually migrating the older classic expressions to the default syntax.

Note that when you upgrade the NetScaler to version 9.0 or higher, Integrated Caching policies are automatically upgraded to default syntax policies, and the expressions in these policies are upgraded to the default syntax.

The following is the syntax for all default syntax expressions that use a classic expression:

```
SYS.EVAL_CLASSIC_EXPR("expression")
```

Following are examples of the `SYS.EVAL_CLASSIC_EXPR("expression")` expression:

```
sys.eval_classic_expr("req.ssl.client.cipher.bits > 1000")
sys.eval_classic_expr("url contains abc")
sys.eval_classic_expr("req.ip.sourceip == 10.102.1.61 -netmask 255.255.255.255")
sys.eval_classic_expr("time >= *:30:00GMT")
sys.eval_classic_expr("e1 || e2")
sys.eval_classic_expr("req.http.urlllen > 50")
sys.eval_classic_expr("dayofweek == wedGMT")
```

## Configuring Default Syntax Expressions in a Policy

You can configure a default syntax expression of up to 1,499 characters in a policy. The user interface for default syntax expressions depends to some extent on the feature for which you are configuring the expression, and on whether you are configuring an expression for a policy or for another use.

When configuring expressions on the command line, you delimit the expression by using quotation marks (â€œ. .â€• or ' . '). Within an expression, you escape additional quotation marks by using a back-slash (\). For example, the following are standard methods for escaping quotation marks in an expression:

```
" \"abc\" "
```

```
â€˜~\"abc\"â€™
```

You must also use a backslash to escape question marks and other backslashes on the command line. For example, the expression `http.req.url.contains(â€œ?â€•)` requires a backslash so that the question mark is parsed. Note that the backslash character will not appear on the command line after you type the question mark. On the other hand, if you escape a backslash (for example, in the expression `'http.req.url.contains( "\\http" )'`), the escape characters are echoed on the command line.

To make an entry more readable, you can escape the quotation marks for an entire expression. At the start of the expression you enter the escape sequence `â€œqâ€• plus one of the following special characters: / { < | ~ $ ^ + = & % @ ` ? .`

You enter only the special character at the end of the expression, as follows:

```
q@http.req.url.contains("sometext") && http.req.cookie.exists@
```

```
q~http.req.url.contains("sometext") && http.req.cookie.exists~
```

Note that an expression that uses the { delimiter is closed with }.

For some features (for example, Integrated Caching and Responder), the policy configuration dialog box provides a secondary dialog box for configuring expressions. This dialog enables you to choose from drop-down lists that show the available choices at each point during expression configuration. You cannot use arithmetic operators when using these configuration dialogs, but most other default syntax expression features are available. To use arithmetic operators, write your expressions in free-form format.

## To configure a default syntax rule by using the command line interface

At the command prompt, type the following commands to configure a default syntax rule and verify the configuration:

1. `add cache|dns|rewrite|cs policy policyName -rule expression featureSpecificParameters -action`
  2. `show cache|dns|rewrite|cs policy policyName`
- Following is an example of configuring a caching policy:

### Example

```
> add cache policy pol-cache -rule http.req.content_length.le(5) -action INVALID
Done

> show cache policy pol-cache
    Name: pol-cache
    Rule: http.req.content_length.le(5)
    CacheAction: INVALID
    Invalidate groups: DEFAULT
    UndefAction: Use Global
    Hits: 0
    Undef Hits: 0

Done
```

## To configure a default syntax policy expression by using the configuration utility

1. In the navigation pane, click the name of the feature where you want to configure a policy, for example, you can select Integrated Caching, Responder, DNS, Rewrite, or Content Switching, and then click Policies.
2. Click Add.

3. For most features, click in the Expression field. For Content Switching, click Configure.
4. Click the Prefix icon (the house) and select the first expression prefix from the drop-down list. For example, in Responder, the options are HTTP, SYS, and CLIENT. The next set of applicable options appear in a drop-down list.
5. Double-click the next option to select it, and then type a period (.). Again, a set of applicable options appears in another drop-down list.
6. Continue selecting options until an entry field (signalled by parentheses) appears. When you see an entry field, enter an appropriate value in the parentheses. For example, if you select GT(int) (greater-than, integer format), you specify an integer in the parentheses. Text strings are delimited by quotation marks. Following is an example:

```
HTTP.REQ.BODY(1000).BETWEEN("this","that")
```

7. To insert an operator between two parts of a compound expression, click the Operators icon (the sigma), and select the operator type. Following is an example of a configured expression with a Boolean OR (signalled by double vertical bars, ||):

```
HTTP.REQ.URL.EQ("www.mycompany.com") || HTTP.REQ.BODY(1000).BETWEEN("this","that")
```

8. To insert a named expression, click the down arrow next to the Add icon (the plus sign) and select a named expression.
9. To configure an expression using drop-down menus, and to insert built-in expressions, click the Add icon (the plus sign). The Add Expression dialog box works in a similar way to the main dialog box, but it provides drop-down lists for selecting options, and it provides text fields for data entry instead of parentheses. This dialog box also provides a Frequently Used Expressions drop-down list that inserts commonly used expressions. When you are done adding the expression, click OK.
10. When finished, click Create. A message in the status bar indicates that the policy expression is configured successfully.

## To test a default syntax expression by using the configuration utility

1. In the navigation pane, click the name of the feature for which you want to configure a policy (for example, you can select Integrated Caching, Responder, DNS, Rewrite, or Content Switching), and then click Policies.
2. Select a policy and click Open.
3. To test the expression, click the Evaluate icon (the check mark).
4. In the expression evaluator dialog box, select the Flow Type that matches the expression.
5. In the HTTP Request Data or HTTP Response Data field, paste the HTTP request or response that you want to parse with the expression, and click Evaluate. Note that you must supply a complete HTTP request or response, and the header and body should be separated by blank line. Some programs that trap HTTP headers do not also trap the response. If you are copying and pasting only the header, insert a blank line at the end of the header to form a complete HTTP request or response.
6. Click Close to close this dialog box.



## Configuring Named Default Syntax Expressions

Instead of retyping the same expression multiple times in multiple policies, you can configure a named expression and refer to the name any time you want to use the expression in a policy. For example, you could create the following named expressions:

```
ThisExpression:
    http.req.body(100).contains("this")
ThatExpression:
    http.req.body(100).contains("that")
```

You can then use these named expressions in a policy expression. For example, the following is a legal expression based on the preceding examples:

```
ThisExpression || ThatExpression
```

You can use the name of a default syntax expression as the prefix to a function. The named expression can be either a simple expression or a compound expression. The function must be one that can operate on the type of data that is returned by the named expression.

### Example 1: Simple Named Expression as a Prefix

The following simple named expression, which identifies a text string, can be used as a prefix to the `AFTER_STR` ("`<string>`") function, which works with text data:

```
HTTP.REQ.BODY(1000)
```

If the name of the expression is `top1KB`, you can use `top1KB.AFTER_STR("username")` instead of `HTTP.REQ.BODY(1000).AFTER_STR("username")`.

### Example 2: Compound Named Expression as a Prefix

You can create a compound named expression called `basic_header_value` to concatenate the user name in a request, a colon (:), and the user's password, as follows:

```
add policy expression basic_header_value "HTTP.REQ.USER.NAME + \":\" + HTTP.REQ.USER.PASSWD"
```

You can then use the name of the expression in a rewrite action, as shown in the following example:

```
add rewrite action insert_b64encoded_authorization insert_http_header authorization
'"Basic " + basic_header_value.b64encode' -bypassSafetyCheck YES
```

In the example, in the expression that is used to construct the value of the custom header, the B64 encoding algorithm is applied to the string returned by the compound named expression.

You can also use a named expression (either by itself or as a prefix to a function) to create the text expression for the replacement target in a rewrite.

## To configure a named default syntax expression by using the command line interface

At the command prompt, type the following commands to configure a named expression and verify the configuration:

- `add policy expression <name><value>`
- `show policy expression <name>`

### Example

```
> add policy expression myExp "http.req.body(100).contains(\"the other\")"
Done

> show policy expression myExp
1)      Name: myExp  Expr: "http.req.body(100).contains("the other")"  Hits: 0
Done
```

The expression can be up to 1,499 characters.

## To configure a named expression by using the configuration utility

1. In the navigation pane, expand AppExpert, and then click Expressions.
2. Click Advanced Expressions.
3. Click Add.
4. Enter a name and a description for the expression.
5. Configure the expression by using the process described in ["To configure a default syntax policy expression by using the configuration utility."](#) A message in the status bar indicates that the policy expression is configured successfully.

## Configuring Default Syntax Expressions Outside the Context of a Policy

A number of functions, including the following, can require a default syntax expression that is not part of a policy:

### Integrated Caching selectors:

You define multiple non-compound expressions (selectlets) in the definition of the selector. Each selectlet is in an implicit logical AND relationship with the others.

### Load Balancing:

You configure an expression for the TOKEN method of load balancing for a load balancing virtual server.

### Rewrite actions:

Expressions define the location of the rewrite action and the type of rewriting to be performed, depending on the type of rewrite action that you are configuring. For example, a DELETE action only uses a target expression. A REPLACE action uses a target expression and an expression to configure the replacement text.

### Rate-based policies:

You use default syntax expressions to configure Limit Selectors. You can use these selectors when configuring policies to throttle the rate of traffic to various servers. You define up to five non-compound expressions (selectlets) in the definition of the selector. Each selectlet is in an implicit logical AND with the others.

## To configure a default syntax expression outside a policy by using the command line interface (cache selector example)

At the command prompt, type the following commands to configure a default syntax expression outside a policy and verify the configuration:

- o add cache selector <selectorName> <rule>
- o show cache selector <selectorName>

### Example

```
> add cache selector mainpageSelector "http.req.cookie.value(\"ABC_def\")"
"http.req.url.query.value(\"_ghi\")" selector "mainpageSelector" added
Done
> show cache selector mainpageSelector
Name: mainpageSelector
Expressions:
    1) http.req.cookie.value("ABC_def")
    2) http.req.url.query.value("_ghi")
Done
```

Following is an equivalent command that uses the more readable q delimiter, as described in "Configuring Default Syntax Expressions in a Policy":

```
> add cache selector mainpageSelector2 q~http.req.cookie.value("ABC_def")~
q~http.req.url.query.value("_ghi")~selector "mainpageSelector2" added
Done
> show cache selector mainpageSelector2
Name: mainpageSelector2
Expressions:
    1) http.req.cookie.value("ABC_def")
    2) http.req.url.query.value("_ghi")
Done
```

## Default Syntax Expressions: Evaluating Text

You can configure a policy with a default syntax expression that evaluates text in a request or response. Default syntax text expressions can range from simple expressions that perform string matching in HTTP headers to complex expressions that encode and decode text. You can configure text expressions to be case sensitive or case insensitive and to use or ignore spaces. You can also configure complex text expressions by combining text expressions with Boolean operators

You can use expression prefixes and operators for evaluating HTTP requests, HTTP responses, and VPN and Clientless VPN data. However, text expression prefixes are not restricted to evaluating these elements of your traffic. For information about additional default syntax text expression prefixes and operators, see the following topics:

- ["Pattern Sets"](#)
- ["Regular Expressions"](#)
- ["Typecasting Data"](#)
- ["Default Syntax Expressions: Parsing HTTP, TCP, and UDP Data"](#)
- ["Default Syntax Expressions: Parsing SSL Certificates"](#)
- ["Expressions for SSL Certificate Dates"](#)

## About Text Expressions

You can configure various expressions for working with text that flows through the NetScaler appliance. Following are some examples of how you can parse text by using a default syntax expression:

- o Determine that a particular HTTP header exists.

For example, you may want to identify HTTP requests that contains a particular Accept-Language header for the purpose of directing the request to a particular server.

- o Determine that a particular HTTP URL contains a particular string.

For example, you may want to block requests for particular URLs. Note that the string can occur at the beginning, middle, or end of another string.

- o Identify a POST request that is directed to a particular application.

For example, you may want to identify all POST requests that are directed to a database application for the purpose of refreshing cached application data.

Note that there are specialized tools for viewing the data stream for HTTP requests and responses. For example, from the following URL, you can download a Firefox Web browser plug-in that displays HTTP request and response headers:

["https://addons.mozilla.org/en-US/firefox/addon/3829"](https://addons.mozilla.org/en-US/firefox/addon/3829)

The following plug-in displays headers, query strings, POST data, and other information:

["https://addons.mozilla.org/en-US/firefox/addon/6647"](https://addons.mozilla.org/en-US/firefox/addon/6647)

After you download these plug-ins, they are accessible from the Firefox Tools menu.

## About Operations on Text

A text-based expression consists of at least one prefix to identify an element of data and usually (although not always) an operation on that prefix. Text-based operations can apply to any part of a request or a response. Basic operations on text include various types of string matches.

For example, the following expression compares a header value with a string:

```
http.req.header("myHeader").contains("some-text")
```

Following expressions are examples of matching a file type in a request:

```
http.req.url.suffix.contains("jpeg")
```

```
http.req.url.suffix.eq("jpeg")
```

In the preceding examples, the contains operator permits a partial match and the eq operator looks for an exact match.

Other operations are available to format the string before evaluating it. For example, you can use text operations to strip out quotes and white spaces, to convert the string to all lowercase, or to concatenate strings.

Note: Complex operations are available to perform matching based on patterns or to convert one type of text format to another type.

For more information, see the following topics:

- o ["Pattern Sets and Data Sets."](#)
- o ["Regular Expressions."](#)
- o ["Typecasting Data."](#)

## Compounding and Precedence in Text Expressions

You can apply various operators to combine text prefixes or expressions. For example, the following expression concatenates the returned values of each prefix:

```
http.req.hostname + http.req.url
```

Following is an example of a compound text expression that uses a logical AND. Both components of this expression must be TRUE for a request to match the expression:

```
http.req.method.eq(post) && http.req.body(1024).startswith("destination=")
```

Note: For more information on operators for compounding, see ["Compound Default Syntax Expressions."](#)

## Categories of Text Expressions

The primary categories of text expressions that you can configure are:

- Information in HTTP headers, HTTP URLs, and the POST body in HTTP requests.

For more information, see ["Expression Prefixes for Text in HTTP Requests and Responses."](#)

- Information regarding a VPN or a clientless VPN.

For more information, see ["Expression Prefixes for VPNs and Clientless VPNs."](#)

- TCP payload information.

For more information about TCP payload expressions, see ["Default Syntax Expressions: Parsing HTTP, TCP, and UDP Data."](#)

- Text in a Secure Sockets Layer (SSL) certificate.

For information about text expressions for SSL and SSL certificate data, see ["Default Syntax Expressions: Parsing SSL Certificates"](#) and ["Expressions for SSL Certificate Dates."](#)

Note: Parsing a document body, such as the body of a POST request, can affect performance. You may want to test the performance impact of policies that evaluate a document body.

## Guidelines for Text Expressions

From a performance standpoint, it typically is best to use protocol-aware functions in an expression. For example, the following expression makes use of a protocol-aware function:

```
HTTP.REQ.URL.QUERY
```

The previous expression performs better than the following equivalent expression, which is based on string parsing:

```
HTTP.REQ.URL.AFTER_STR(" ? ")
```

In the first case, the expression looks specifically at the URL query. In the second case, the expression scans the data for the first occurrence of a question mark.

There is also a performance benefit from structured parsing of text, as in the following expression:

```
HTTP.REQ.HEADER("Example").TYPECAST_LIST_T(' , ').GET(1)
```

(For more information on typecasting, see ["Typecasting Data."](#)) The typecasting expression, which collects comma-delimited data and structures it into a list, typically would perform better than the following unstructured equivalent:

```
HTTP.REQ.HEADER("Example").AFTER_STR(", ").BEFORE_STR(", ")
```

Finally, unstructured text expressions typically have better performance than regular expressions. For example, the following is an unstructured text expression:

```
HTTP.REQ.HEADER("Example").AFTER_STR("more")
```

The previous expression would generally provide better performance than the following equivalent, which uses a regular expression:

```
HTTP.REQ.HEADER("Example").AFTER_REGEX(re/more/)
```

For more information on regular expressions, see ["Regular Expressions."](#)

## Expression Prefixes for Text in HTTP Requests and Responses

An HTTP request or response typically contains text, such as in the form of headers, header values, URLs, and POST body text. You can configure expressions to operate on one or more of these text-based items in an HTTP request or response.

The following table describes the expression prefixes that you can configure to extract text from different parts of an HTTP request or response.

Table 1. HTTP Expression Prefixes That Return Text

Prefix	Description
<code>HTTP.REQ.BODY(&lt;integer&gt;)</code>	<p>Returns the body of an HTTP request as a multi-<code>&lt;integer&gt;</code>.</p> <p>There is no maximum value for the body argument; large values can affect performance.</p> <p>Note: Although it is possible to specify this prefix,</p>
<code>HTTP.REQ.HOSTNAME</code>	<p>Returns the HTTP host name in the first line of text in the last occurrence of the HOST header.</p> <p>Note that there are two similar prefixes that return host names:</p> <ul style="list-style-type: none"> <li>• <code>http.req.url.hostname</code> only returns the host name in the URL.</li> <li>• <code>http.req.header("Host")</code> only returns the host name you must typecast this string, as <code>("host").typecast_http_hostname</code>.</li> </ul> <p>For more information on typecasting, see "Type</p>
<code>HTTP.REQ.HOSTNAME.DOMAIN</code>	<p>Returns the domain name part of the host name. For example, in <code>myhost.com:8080</code>, the domain is <code>myhost.com</code>.</p> <p>Returns incorrect results if the host name has an IP address. See "Default Syntax Expressions: IP and MAC Address</p> <p>All text operations that you specify after this prefix</p>
<code>HTTP.REQ.HOSTNAME.SERVER</code>	<p>Returns the server name part of the host name. For example, in <code>www.myhost.com</code>, the server is <code>www</code>.</p> <p>All text operations that you specify after this prefix</p>
<code>HTTP.REQ.METHOD</code>	<p>Returns the value of the METHOD in an HTTP request. For example, <code>http.req.method.equals("GET")</code> is case sensitive.</p>
<code>HTTP.REQ.URL</code>	<p>Returns the HTTP URL.</p>
<code>HTTP.REQ.URL.HOSTNAME</code>	<p>Returns the host name in the HTTP URL.</p> <p>Do not use this prefix in bidirectional policies.</p> <p>Note that there are two similar prefixes that return host names:</p> <ul style="list-style-type: none"> <li>• <code>HTTP.REQ.HOSTNAME</code> returns the host name in the last occurrence of the Host</li> </ul>

	<ul style="list-style-type: none"> <li>○ <code>HTTP.REQ.HEADER("Host")</code> only returns the name you must typecast this string, as <code>("host").typecast_http_hostname</code></li> </ul> <p>For more information on typecasting, see "Type</p>
<code>HTTP.REQ.URL.HOSTNAME.DOMAIN</code>	<p>Returns the domain name part of the host name <code>myhost.com:8080</code>, the domain is <code>myhost.com</code>.</p> <p>This operation returns incorrect results if the host addresses, see "Default Syntax Expressions: IP</p> <p>All text operations that you specify after this prefix: <code>SET_TEXT_MODE</code> operator.</p>
<code>HTTP.REQ.URL.HOSTNAME.SERVER</code>	<p>Returns the server name part of the host name. <code>myhost.com:8080</code>, the server is <code>www.myhost.com</code></p> <p>All text operations that you specify after this prefix:</p>
<code>HTTP.REQ.URL.HOSTNAME.PORT</code>	<p>Returns the port in the host name. The string follows the port value. For example, if the host name is <code>www.mycompany.com</code>: the port is <code>:</code>. If the host location is just after <code>".com"</code>.</p> <p>If the numerical value in the port is missing, it as</p>
<code>HTTP.REQ.URL.PATH</code>	<p>Returns a slash- (/) separated list from the path</p> <p>For example, if the URL is <code>http://www.myhost.com/a/b/c/mypage.html</code>.</p> <p>The expression <code>http.req.url.path.get(1)</code> GET operation, see "Expressions for Extracting</p>
<code>HTTP.REQ.URL.PATH_AND_QUERY</code>	<p>Returns the portion of the URL that follows the host</p> <p>For example, if the URL is <code>http://www.myhost.com/a=1</code>.</p>
<code>HTTP.REQ.URL.PROTOCOL</code>	<p>Returns the protocol in the URL.</p> <p>This prefix cannot be used in bidirectional policies</p> <p><code>http.req.hostname + http.req.url.pr</code></p>
<code>HTTP.REQ.URL.QUERY</code>	<p>Returns a name-value list, using the delimiters <code>&amp;</code></p> <p>Following is an example:</p> <p><code>http.req.url.query.contains("viewRe</code></p>
<code>HTTP.REQ.URL.QUERY.VALUE</code>	<p>Returns the value from the name-value pair in the query from the query component in the URL.</p> <p>Following is an example:</p> <p><code>http.req.url.query.value("action")</code></p> <p>The first component that matches the name is sensitive to NOIGNORECASE text modes. The URLENCODING</p>



HTTP.REQ.URL.SUFFIX	<p>Returns the file name suffix in a URL.</p> <p>For example, if the path in the URL is /a/b/c/myexample: example:</p> <pre>http.req.url.suffix.contains("jpeg")</pre>
HTTP.REQ.USER	Returns the AAA user associated with the current request.
HTTP.REQ.USER.EXTERNAL_GROUPS	<p>Returns a list of the external groups to which a user belongs.</p> <p>For example, HTTP.REQ.USER.EXTERNAL_GROUPS returns a list of the external groups to which the user belongs.</p>
HTTP.REQ.USER.EXTERNAL_GROUPS.IGNORE_EMPTY_ELEMENTS	<p> Ignores the empty elements in the list of external groups.</p> <p>If the element delimiter in the list is a comma (",") and the list is:</p> <pre>a=10,,b=11, ,c=89</pre> <p>But the element following "b=11" is not considered.</p> <p>For example, consider the following header in a request:</p> <pre>Cust_Header : 123,,24, ,15</pre> <p>Then the following expression returns a value of 123:</p> <pre>HTTP.REQ.HEADER("Cust_Header").TYPEOF</pre> <p>The following expression returns a value of 5:</p> <pre>HTTP.REQ.HEADER("Cust_Header").TYPEOF</pre>
HTTP.REQ.USER.EXTERNAL_GROUPS ( sep )	<p>Returns a list of all the external groups to which a user belongs, separated by the delimiter.</p> <p>For example, the following expression gives a list of the external groups separated by a colon (":"):</p> <pre>HTTP.REQ.USER.EXTERNAL_GROUPS ( ':' )</pre> <p>Parameters:</p> <p>sep - delimiter</p>
HTTP.REQ.USER.GROUPS	<p>Returns a list of the internal and external groups to which a user belongs, separated by a comma (",").</p> <p>In this list, internal groups are listed first, followed by external groups.</p>
HTTP.REQ.USER.GROUPS.IGNORE_EMPTY_ELEMENTS	<p> Ignores the empty elements in the list of groups.</p> <p>If the element delimiter in the list is a comma (",") and the list is:</p> <pre>a=10,,b=11, ,c=89</pre> <p>But the element that follows "b=11" is not considered.</p> <p>For example, consider the following header in a request:</p> <pre>Cust_Header : 123,,24, ,15</pre> <p>The following expression returns a value of 4:</p> <pre>HTTP.REQ.HEADER("Cust_Header").TYPEOF</pre>

	<p>The following expression returns a value of 5:</p> <pre>HTTP.REQ.HEADER( "Cust_Header" ).TYPE</pre>
<code>HTTP.REQ.USER.GROUPS( sep )</code>	<p>Returns a list of groups to which the user belongs as the argument.</p> <p>For example, the following expression returns a</p> <pre>HTTP.REQ.USER.GROUPS( ' : ' )</pre> <p>In this list, internal groups are listed first, followed by external groups.</p> <p>Parameters:</p> <p>sep - delimiter</p>
<code>HTTP.REQ.USER.INTERNAL_GROUPS</code>	<p>Returns a list of internal groups to which the user belongs.</p> <p>For example, the following expression returns a list of internal groups.</p> <pre>HTTP.REQ.USER.INTERNAL_GROUPS</pre>
<code>HTTP.REQ.USER.INTERNAL_GROUPS. IGNORE_EMPTY_ELEMENTS</code>	<p> Ignores the empty elements in the list of internal groups.</p> <p>If the element delimiter in the list is a comma (','), the following expression returns a value of 4:</p> <pre>a=10,,b=11, ,c=89</pre> <p>But the element following "b=11" is not considered.</p> <p>For example, consider the following header in a request:</p> <pre>Cust_Header : 123,,24, ,15</pre> <p>The following expression returns a value of 4:</p> <pre>HTTP.REQ.HEADER( "Cust_Header" ).TYPE</pre> <p>The following expression returns a value of 5:</p> <pre>HTTP.REQ.HEADER( "Cust_Header" ).TYPE</pre>
<code>HTTP.REQ.USER.INTERNAL_GROUPS( sep )</code>	<p>Returns a list of the internal groups to which the user belongs.</p> <p>For example, the following expression returns a list of internal groups.</p> <pre>HTTP.REQ.USER.INTERNAL_GROUPS( ' : ' )</pre> <p>Parameters:</p> <p>sep - delimiter</p>
<code>HTTP.REQ.USER.IS_MEMBER_OF( group_name )</code>	<p>Returns a boolean TRUE if the user who is named group_name is a member of the group.</p> <p>Following is an example:</p> <pre>http.req.user.is_member_of( "mygroup"</pre> <p>Parameter:</p> <p>group_name: The name of the group.</p>

HTTP.REQ.USER.NAME	<p>Returns the name of the user in the request.</p> <p>Following is an example:</p> <pre>http.req.username.contains("rohit")</pre>
HTTP.REQ.USER.PASSWD	<p>Returns the password of the user.</p>
HTTP.REQ.VERSION	<p>Returns the HTTP version listed in the request.</p> <p>Following is an example:</p> <pre>http.req.version "\"HTTP/1.0\""</pre>
HTTP.RES.BODY(<integer>)	<p>Returns a portion of the HTTP response body. 1 &lt;integer&gt; argument.</p> <p>If there are fewer characters in the body than an example:</p> <pre>http.res.body(100).suffix('L',1)</pre>
HTTP.RES.STATUS_MSG	<p>Returns the HTTP response status message.</p>
HTTP.RES.VERSION	<p>Returns the HTTP version listed in the response</p>
HTTP.REQ.URL.HOSTNAME.EQ(<hostname>)	<p>Returns a Boolean TRUE value if the host name insensitive and if textmode is URLENCODED, the host name is www.mycompany.com., the following</p> <pre>http.req.url.hostname.eq("www.mycom</pre>
HTTP.REQ.IS_NTLM_OR_NEGOTIATE	<p>Returns a Boolean TRUE if the request is a part</p>
HTTP.REQ.URL.CVPN_ENCODE	<p>Converts the URL to the clientless VPN format.</p>
HTTP.REQ.URL.PATH.IGNORE_EMPTY_ELEMENTS	<p>Ignores the empty elements in the list. For example, a=10;b=11;,c=89 has an empty element following a=10:</p> <pre>a=10;b=11;,c=89</pre> <p>The element following b=11 is not considered as a header element.</p> <p>As another example, consider the following header:</p> <pre>Cust_Header : 123,,24, ,15</pre> <p>The following expression returns a value of 4:</p> <pre>http.req.header("Cust_Header").type</pre> <p>The following expression returns a value of 5:</p> <pre>http.req.header("Cust_Header").type</pre>
HTTP.REQ.URL.QUERY.IGNORE_EMPTY_ELEMENTS	<p>This method ignores the empty elements in a name-value list. For example, a=10;;b=11;;c=89 following list has an empty element following a=10:</p> <pre>a=10;;b=11;;c=89</pre> <p>The element following b=11 is not considered as a header element.</p>

For example, consider the following header:

```
Cust_Header : a=1;;b=2; ;c=3
```

The following expression returns a value of 4:

```
http.req.header( "Cust_Header" ).type  
count
```

The following expression returns a value of 5:

```
http.req.header( "Cust_Header" ).type
```

## Expression Prefixes for VPNs and Clientless VPNs

The default syntax expression engine provides prefixes that are specific to parsing VPN or Clientless VPN data. This data includes the following:

- Host names, domains, and URLs in VPN traffic.
- Protocols in the VPN traffic.
- Queries in the VPN traffic.

These text elements are often URLs and components of URLs. In addition to applying the text-based operations on these elements, you can parse these elements by using operations that are specific to parsing URLs. For more information, see ["Expressions for Extracting Segments of URLs."](#)

The following table describes the expression prefixes for this type of data.

Table 1. VPN and Clientless VPN Expression Prefixes That Return Text

VPN and Clientless VPN Expression	Description
<code>VPN.BASEURL.CVPN_DECODE</code>	Extracts the original URL from a clientless VPN URL.
<code>VPN.BASEURL.CVPN_ENCODE</code>	Converts a URL to clientless VPN format.
<code>VPN.BASEURL.HOSTNAME</code>	Extracts the HTTP host name from the host name.  This prefix cannot be used in bidirectional policies.
<code>VPN.BASEURL.HOSTNAME.DOMAIN</code>	Extracts the domain name from the host name.  For example, if the host name is <code>www.mycompany.mycompany.com</code> .  This prefix returns incorrect results if the host name contains IP addresses, see <a href="#">"Default Syntax Expressions: IP addresses"</a> .  All text operations after this prefix are case insensitive.
<code>VPN.BASEURL.HOSTNAME.EQ (&lt;hostname&gt;)</code>	Returns a Boolean TRUE if the host name matches the specified host name.  For example, if the host name is <code>www.mycompany.com</code> , the following expression returns TRUE: <code>vpn.baseurl.hostname.eq("www.mycompany.com")</code>  If the text mode is URLENCODED, the host name is URL encoded. For more information, see <a href="#">"Operations for HTTP, HTML, and XML Encoding and Decoding"</a> .
<code>VPN.BASEURL.HOSTNAME.SERVER</code>	Evaluates the server portion of the host name.  For example, if the host name is <code>www.mycompany.mycompany.com</code> , the following expression returns <code>www</code> : <code>vpn.baseurl.hostname.server</code>  All text operations after this prefix are case insensitive.
<code>VPN.BASEURL.PATH</code>	Extracts a slash- (/) separated list from the path component of the URL.  For example, if the URL is <code>http://www.mycompany.com/a/b/c/mypage.html?a=1&amp;b=2</code> , the following expression returns <code>a/b/c</code> : <code>http.req.url.path.get(1)</code>  For more information on the GET operation, see <a href="#">"GET Operation"</a> .

VPN.BASEURL.PATH.IGNORE_EMPTY_ELEMENTS	<p>This prefix ignores the elements in a list. For example, after <code>a=10</code>:</p> <p><code>a=10,,b=11, ,c=89</code></p> <p>The element following <code>b=11</code> contains a space, and</p> <p>Consider the following HTTP header:</p> <p><code>Cust_Header : 123,,24, ,15</code></p> <p>The following expression returns a count of 4 when</p> <pre>http.req.header("Cust_Header").typecase</pre> <p>The following expression returns a count of 5 when</p> <pre>http.req.header("Cust_Header").typecase</pre>
VPN.BASEURL.PATH_AND_QUERY	<p>Evaluates the text in the URL that follows the host</p> <p>For example, if the URL is <code>http://www.mycompany.com/a/b/c/mypage.html?a=1</code>.</p>
VPN.BASEURL.PROTOCOL	<p>Evaluates the protocol in the URL.</p> <p>Do not use this prefix in bidirectional policies.</p>
VPN.BASEURL.QUERY	<p>Extracts a name-value list, using the <code>a=10</code> and</p>
VPN.BASEURL.QUERY.IGNORE_EMPTY_ELEMENTS	<p>This method ignores the empty elements in a name-value list. For example, after <code>a=10</code>:</p> <p><code>a=10;;b=11; ;c=89</code></p> <p>The element following <code>b=11</code> contains a space and</p> <p>Consider the following HTTP header:</p> <p><code>Cust_Header : a=1;;b=2; ;c=3</code></p> <p>The following expression produces a count of 4 after</p> <pre>http.req.header("Cust_Header").typecase.count</pre> <p>The following expression produces a count of 5 after</p> <pre>http.req.header("Cust_Header").typecase</pre>
VPN.BASEURL.SUFFIX	<p>Evaluates the file name suffix in a URL.</p> <p>For example, if the path is <code>/a/b/c/my.page.html</code>, this</p>
VPN.CLIENTLESS_BASEURL	<p>Evaluates the clientless VPN base URL.</p>
VPN.CLIENTLESS_BASEURL.CVPN_DECODE	<p>Extracts the original URL from the clientless VPN format.</p>
VPN.CLIENTLESS_BASEURL.CVPN_ENCODE	<p>Converts a URL to the clientless VPN format.</p>
VPN.CLIENTLESS_BASEURL.HOSTNAME	<p>Evaluates the host name in the URL.</p>

	Do not use this prefix in bidirectional policies.
<code>VPN.CLIENTLESS_BASEURL.HOSTNAME.DOMAIN</code>	<p>Evaluates the domain name part of the host name.</p> <p>For example, if the host name is <code>www.mycompany.com</code>.</p> <p>This operation returns incorrect results if the host r addresses, see "<a href="#">Default Syntax Expressions: IP ar</a></p> <p>All text operations after this prefix are case insensi</p>
<code>VPN.CLIENTLESS_BASEURL.HOSTNAME.EQ(&lt;hostname&gt;)</code>	<p>Returns a Boolean TRUE if the host name matche</p> <p>For example, if the host name is <code>www.mycompany</code></p> <pre>vpn.clientless_baseurl.hostname.eq("'</pre> <p>The comparison is case insensitive. If the textmod comparison. For more information, see "<a href="#">Operation: Characters</a>."</p>
<code>VPN.CLIENTLESS_BASEURL.HOSTNAME.SERVER</code>	<p>Evaluates the server part of a host name.</p> <p>For example, if the host name is <code>www.mycompany.mycompany.com</code>.</p> <p>All text operations after this prefix are case insensi</p>
<code>VPN.CLIENTLESS_BASEURL.PATH</code>	<p>Evaluates a slash- (/) separated list in the URL pat</p> <p>For example, this prefix selects <code>/a/b/c/mypage.htm</code></p> <p><code>http://www.mycompany.com/a/b/c/mypage.html?a=</code></p> <p>The following expression selects <code>â€œaâ€•</code> from th</p> <pre>http.req.url.path.get(1)</pre> <p>For more information on the GET operation, see "E</p>
<code>VPN.CLIENTLESS_BASEURL.PATH.IGNORE_EMPTY_ELEMENTS</code>	<p>Ignores empty elements in a list. For example, if th element following <code>â€œa=10â€•</code>:</p> <pre>a=10,b=11, ,c=89</pre> <p>The element following <code>b=11</code> contains a space and</p> <p>Consider the following HTTP header:</p> <pre>Cust_Header : 123,,24, ,15</pre> <p>The following expression returns a value of 4 after</p> <pre>http.req.header("Cust_Header").typeca</pre> <p>The following expression returns a value of 5 after</p> <pre>http.req.header("Cust_Header").typeca</pre>
<code>VPN.CLIENTLESS_BASEURL.PATH_AND_QUERY</code>	<p>Evaluates the text following the host name in a UR</p> <p>For example, this prefix selects <code>/a/b/c/mypage.htm</code></p> <p><code>http://www.mycompany.com/a/b/c/mypage.html?a=</code></p>

<code>VPN.CLIENTLESS_BASEURL.PROTOCOL</code>	<p>Evaluates the protocol in the URL.</p> <p>Do not use this prefix in bidirectional policies.</p>
<code>VPN.CLIENTLESS_BASEURL.QUERY</code>	Extracts a name-value list that uses the delimiters
<code>VPN.CLIENTLESS_BASEURL.QUERY.IGNORE_EMPTY_ELEMENTS</code>	<p>Ignores empty elements in a name-value list. For example, if the query is <code>a=10&amp;b=11;c=89</code>:</p> <p><code>a=10;;b=11; ;c=89</code></p> <p>The element following <code>b=11</code> contains a space and a semicolon.</p> <p>As another example, consider the following HTTP header:</p> <p><code>Cust_Header : a=1;;b=2; ;c=3</code></p> <p>The following expression returns a value of 4 after ignoring empty elements:</p> <pre>http.req.header("Cust_Header").typecast(count)</pre> <p>The following expression returns a value of 5 after ignoring empty elements:</p> <pre>http.req.header("Cust_Header").typecast(count)</pre>
<code>VPN.CLIENTLESS_BASEURL.SUFFIX</code>	Evaluates the file suffix in a URL. For example, if the URL is <code>http://www.example.com/html</code> .
<code>VPN.CLIENTLESS_HOSTURL</code>	Selects the clientless VPN host URL.
<code>VPN.CLIENTLESS_HOSTURL.CVPN_DECODE</code>	Selects the original URL from the clientless VPN format.
<code>VPN.CLIENTLESS_HOSTURL.CVPN_ENCODE</code>	Converts a URL to clientless VPN format.
<code>VPN.CLIENTLESS_HOSTURL.HOSTNAME</code>	<p>Extracts the host name in the URL.</p> <p>Do not use this prefix in bidirectional policies.</p>
<code>VPN.CLIENTLESS_HOSTURL.HOSTNAME.DOMAIN</code>	<p>Extracts the domain name from the host name. For example, if the host name is <code>mycompany.com:8080</code>, the domain is <code>mycompany.com</code>.</p> <p>This operation returns incorrect results if the host name contains IP addresses, see <a href="#">"Default Syntax Expressions: IP Addresses"</a>.</p> <p>All text operations after this prefix are case insensitive.</p>
<code>VPN.CLIENTLESS_HOSTURL.HOSTNAME.EQ(&lt;hostname&gt;)</code>	<p>Results in Boolean TRUE if the host name matches the specified hostname, case insensitive.</p> <p>For example, if the host name is <code>www.mycompany.com</code>, the expression <code>vpn.clientless_hosturl.hostname.eq("www.mycompany.com")</code> returns TRUE.</p> <pre>vpn.clientless_hosturl.hostname.eq("www.mycompany.com")</pre> <p>If the text mode is URLENCODED, the host name is encoded. For more information, see <a href="#">Operations for HTTP, HTML, and XML Encoding and Decoding</a>.</p>
<code>VPN.CLIENTLESS_HOSTURL.HOSTNAME.SERVER</code>	Evaluates the server part of the host name.



	<p>For example, if the host name is www.mycompany.com.</p> <p>The comparison is case insensitive, and all text op</p>
VPN.CLIENTLESS_HOSTURL.PATH	<p>Evaluates a slash- (/) separated list on the path co</p> <p>For example, consider the following URL:</p> <p>http://www.mycompany.com/a/b/c/mypage.html?a=</p> <p>This prefix selects /a/b/c/mypage.html from the pre</p>
VPN.CLIENTLESS_HOSTURL.PATH.IGNORE_EMPTY_ELEMENTS	<p>This method ignores the empty elements in a list. f contains an empty element after the entry a=1</p> <p>a=10,b=11, ,c=89</p> <p>The element following b=11 contains a space and</p> <p>Consider the following header:</p> <p>Cust_Header : 123,,24, ,15</p> <p>The following expression returns a value of 4 for th</p> <p>http.req.header( "Cust_Header" ).typeca</p> <p>The following expression returns a value of 5 for th</p> <p>http.req.header( "Cust_Header" ).typeca</p>
VPN.CLIENTLESS_HOSTURL.PATH_AND_QUERY	<p>Evaluates the portion of the URL that follows the h</p> <p>For example, consider the following URL:</p> <p>http://www.mycompany.com/a/b/c/mypage.html?a=</p> <p>This prefix returns /a/b/c/mypage.html?a=1 from th</p>
VPN.CLIENTLESS_HOSTURL.PROTOCOL	<p>Evaluates the protocol in the URL.</p> <p>Do not use this prefix in bidirectional policies.</p>
VPN.CLIENTLESS_HOSTURL.QUERY	<p>Extracts a name-value list, using the a=a and</p>
VPN.CLIENTLESS_HOSTURL.QUERY.IGNORE_EMPTY_ELEMENTS	<p>Ignores empty elements in a name-value list. For e contains an empty element after a=10a=:</p> <p>a=10;;b=11; ;c=89</p> <p>In the preceding example, the element following b=</p> <p>Consider the following header:</p> <p>Cust_Header : a=1;;b=2; ;c=3</p> <p>The following expression returns a value of 4 after</p> <p>http.req.header( "Cust_Header" ).typeca count</p> <p>The following expression returns a value of 5 after</p> <p>http.req.header( "Cust_Header" ).typeca</p>

VPN.CLIENTLESS_HOSTURL.SUFFIX	<p>Extracts a file name suffix in a URL.</p> <p>For example, if the path is /a/b/c/my.page.html, this</p>
VPN.HOST.DOMAIN	<p>Extracts the domain name part of the host name. For mycompany.com:8080, the domain is mycompany</p> <p>This prefix returns incorrect results if the host name addresses, see <a href="#">"Default Syntax Expressions: IP addresses"</a></p> <p>All text operations after this prefix case insensitive.</p>
VPN.HOST.EQ( <hostname> )	<p>Returns a Boolean TRUE value if the host name matches</p> <p>For example, if the host name is www.mycompany.com, TRUE:</p> <pre>vpn.host.eq( "www.mycompany.com" )</pre> <p>If the text mode is URLENCODED the host name is <a href="#">URL Encoded</a></p>
VPN.HOST.SERVER	<p>Extracts the server name part of the host name. For mycompany.com:8080, the server is www.mycompany.com</p> <p>All text operations after this prefix are case insensitive.</p>

## Basic Operations on Text

Basic operations on text include operations for string matching, calculating the length of a string, and controlling case sensitivity. You can include white space in a string that is passed as an argument to an expression, but the string cannot exceed 255 characters.

## String Comparison Functions

The following table lists basic string matching operations in which the functions return a Boolean TRUE or FALSE.

Table 1. String Comparison Functions

Function	Description
<code>&lt;text&gt;.CONTAINS(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the target contains <string>.  Following is an example:  <code>http.req.url.contains( ".jpeg" )</code>
<code>&lt;text&gt;.EQ(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the target is an exact match with <string>.  For example, the following expression returns a Boolean TRUE for a URL with a host name of <code>myhostabc</code> :  <code>http.req.url.hostname.eq( "myhostabc" )</code>
<code>&lt;text&gt;.STARTSWITH(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the target begins with <string>.  For example, the following expression returns a Boolean TRUE for a URL with a host name of <code>myhostabc</code> :  <code>http.req.url.hostname.startswith( "myhost" )</code>
<code>&lt;text&gt;.ENDSWITH(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the target ends with <string>.  For example, the following expression returns a Boolean TRUE for a URL with a host name of <code>myhostabc</code> :  <code>http.req.url.hostname.endswith( "abc" )</code>
<code>&lt;text&gt;.NE(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the prefix is not equal to the string argument.  If the prefix returns a non-string value, the function argument is compared to the string representation of the value returned by the prefix. You can use the functions with <code>SET_TEXT_MODE(IGNORECASE)</code> or <code>SET_TEXT_MODE(NOIGNORECASE)</code> , and with both ASCII and UTF-8 character sets.
<code>&lt;text&gt;.GT(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the prefix is alphabetically greater than the string argument.  If the prefix returns a non-string value, the function argument is compared to the string representation of the value returned by the prefix. You can use the functions with <code>SET_TEXT_MODE(IGNORECASE)</code> or <code>SET_TEXT_MODE(NOIGNORECASE)</code> , and with both ASCII and UTF-8 character sets.
<code>&lt;text&gt;.GE(&lt;string&gt;)</code>	Returns a Boolean TRUE value if the prefix is alphabetically greater than or equal to the string argument.

	<p>If the prefix returns a non-string value, the function argument is compared to the string representation of the value returned by the prefix. You can use the functions with SET_TEXT_MODE (IGNORECASE) or SET_TEXT_MODE(NOIGNORECASE), and with both ASCII and UTF-8 character sets.</p>
<code>&lt;text&gt;.LT(&lt;string&gt;)</code>	<p>Returns a Boolean TRUE value if the prefix is alphabetically lesser than the string argument.</p> <p>If the prefix returns a non-string value, the function argument is compared to the string representation of the value returned by the prefix. You can use the functions with SET_TEXT_MODE (IGNORECASE) or SET_TEXT_MODE(NOIGNORECASE), and with both ASCII and UTF-8 character sets.</p>
<code>&lt;text&gt;.LE(&lt;string&gt;)</code>	<p>Returns a Boolean TRUE value if the prefix is alphabetically lesser than or equal to the string argument.</p> <p>If the prefix returns a non-string value, the function argument is compared to the string representation of the value returned by the prefix. You can use the functions with SET_TEXT_MODE (IGNORECASE) or SET_TEXT_MODE(NOIGNORECASE), and with both ASCII and UTF-8 character sets.</p>

## Calculating the Length of a String

The `<text>.LENGTH` operation returns a numeric value that is equal to the number of characters (not bytes) in a string:

```
<text>.LENGTH
```

For example, you may want to identify request URLs that exceed a particular length. Following is an expression that implements this example:

```
HTTP.REQ.URL.LENGTH < 500
```

After taking a count of the characters or elements in a string, you can apply numeric operations to them. For more information, see ["Default Syntax Expressions: Working with Dates, Times, and Numbers."](#)

## Considering, Ignoring, and Changing Text Case

The following functions operate on the case (upper-case or lower-case) of the characters in the string.

Table 2. Functions for Considering, Ignoring, and Changing Text Case

Function	Description
<code>&lt;text&gt;.SET_TEXT_MODE (IGNORECASE   NOIGNORECASE)</code>	This function turns case sensitivity on or off for all text operations.
<code>&lt;text&gt;.TO_LOWER</code>	<p>Converts the target to lowercase for a text block of up to 2 kilobyte (KB). Returns UNDEF if the target exceeds 2 KB.</p> <p>For example, the string <code>â€œABCD:â€•</code> is converted to <code>â€œabcd:â€•</code>.</p>
<code>&lt;text&gt;.TO_UPPER</code>	<p>Converts the target to uppercase. Returns UNDEF if the target exceeds 2 KB.</p> <p>For example, the string <code>â€œabcd:â€•</code> is converted to <code>â€œABCD:â€•</code>.</p>

## Stripping Specific Characters from a String

You can use the `STRIP_CHARS(<string>)` function to remove specific characters from the text that is returned by a default syntax expression prefix (the input string). All instances of the characters that you specify in the argument are stripped from the input string. You can use any text method on the resulting string, including the methods used for matching the string with a pattern set.

For example, in the expression `CLIENT.UDP.DNS.DOMAIN.STRIP_CHARS("._-"),` the `STRIP_CHARS(<string>)` function strips all periods (.), hyphens (-), and underscores (\_) from the domain name returned by the prefix `CLIENT.UDP.DNS.DOMAIN`. If the domain name that is returned is "a.dom\_ai\_n-name", the function returns the string "adomainname".

In the following example, the resulting string is compared with a pattern set called "listofdomains":

```
CLIENT.UDP.DNS.DOMAIN.STRIP_CHARS("._-").CONTAINS_ANY("listofdomains")
```

Note: You cannot perform a rewrite on the string that is returned by the `STRIP_CHARS(<string>)` function.

The following functions strip matching characters from the beginning and end of a given string input.

Table 3. Functions for Stripping Characters From the Beginning or End of a String

Function	Description
<code>&lt;text&gt;.STRIP_START_CHARS(s)</code>	<p>Strips matching characters from the beginning of the input string until the first non-matching character is found and returns the remainder of the string. You must specify the characters that you want to strip as a single string within quotation marks.</p> <p>For example, if the name of a header is <code>TestLang</code> and <code>://_en_us:</code> is its value, <code>HTTP.RES.HEADER("TestLang").STRIP_START_CHARS(" :/_")</code> strips the specified characters from the beginning of the value of the header until the first non-matching character <code>e</code> is found and returns <code>en_us:</code> as a string.</p>
<code>&lt;text&gt;.STRIP_END_CHARS(s)</code>	<p>Strips matching characters from the end of the input string to the first non-matching character is found and returns the remainder of the string. You must specify the characters that you want to strip as a single string within quotation marks.</p> <p>For example, if the name of a header is <code>TestLang</code> and <code>://_en_us:</code> is its value, <code>HTTP.RES.HEADER("TestLang").STRIP_END_CHARS(" :/_")</code> strips the specified characters from the end of the value of the header until the first non-matching character <code>s</code> is found and returns <code>://_en_us</code> as a string.</p>

## Appending a String to Another String

You can use the `APPEND()` function to append the string representation of the argument to the string representation of the value returned by the preceding function. The preceding function can be one that returns a number, unsigned long, double, time value, IPv4 address, or IPv6 address. The argument can be a text string, number, unsigned long, double, time value, IPv4 address, or IPv6 address. The resulting string value is the same string value that is obtained by using the `+` operator.

## Complex Operations on Text

In addition to performing simple string matching, you can configure expressions that examine more complex aspects of text, including examining the length of a string and looking within a text block for patterns rather than specific strings.

Be aware of the following for any text-based operation:

- For any operation that takes a string argument, the string cannot exceed 255 characters.
- You can include white space when you specify a string in an expression.

This document includes the following details:

- Operations on the Length of a String
- Operations on a Portion of a String
- Operations for Comparing the Alphanumeric Order of Two Strings
- Extracting an Integer from a String of Bytes That Represent Text
- Converting Text to a Hash Value
- Encoding and Decoding Text by Applying the Base64 Encoding Algorithm
- Refining the Search in a Rewrite Action by Using the EXTEND Function
- Converting Text to Hexadecimal Format
- Encrypting and Decrypting Text

## Operations on the Length of a String

The following operations extract strings on the basis of a character count.

Table 1. String Operations Based on a Character Count

Character Count Operation	Description
<code>&lt;text&gt;.TRUNCATE(&lt;count&gt;)</code>	Returns a string after truncating the end of the target by the number of characters in <code>&lt;count&gt;</code> .  If the entire string is shorter than <code>&lt;count&gt;</code> , nothing is returned.
<code>&lt;text&gt;.TRUNCATE(&lt;character&gt;, &lt;count&gt;)</code>	Returns a string after truncating the text after <code>&lt;character&gt;</code> by the number of characters specified in <code>&lt;count&gt;</code> .
<code>&lt;text&gt;.PREFIX(&lt;character&gt;, &lt;count&gt;)</code>	Selects the longest prefix in the target that has at most <code>&lt;count&gt;</code> occurrences of <code>&lt;character&gt;</code> .
	<code>&lt;text&gt;.SUFFIX(&lt;character&gt;, &lt;count&gt;)</code> Selects the longest suffix in the target that has at most <code>&lt;count&gt;</code> occurrences of <code>&lt;character&gt;</code> .  For example, consider the following response body:  JLEwx  The following expression returns a value of <code>â€œJLEwxâ€œ</code> :  <code>http.res.body(100).suffix('L',1)</code>  The following expression returns <code>â€œLLEwxâ€œ</code> :  <code>http.res.body(100).suffix('L',2)</code>
<code>&lt;text&gt;.SUBSTR(&lt;starting_offset&gt;, &lt;length&gt;)</code>	Select a string with <code>&lt;length&gt;</code> number of characters from the target object. Begin extracting the string after the <code>&lt;starting_offset&gt;</code> . If the number of characters after the offset are fewer than the value of the <code>&lt;length&gt;</code> argument, select all the remaining characters.

```
<text>.SKIP(<character>,  
<count>)
```

Select a string from the target after skipping over the longest prefix that has at most <count> occurrences of <character>.

## Operations on a Portion of a String

You can extract a subset of a larger string by using one of the operations in the following table.

Table 2. Basic Operations on a Portion of a String

Basic Text Operation	Description
<code>&lt;text&gt;.BEFORE_STR(&lt;string&gt;)</code>	<p>Returns the text that precedes the first occurrence of &lt;string&gt;.</p> <p>If there is no match for &lt;string&gt;, the expression returns a text object of 0 length.</p> <p>Following is an example:</p> <pre>http.res.body(1024).after_str("start_string").before_str("e: ("https")</pre>
<code>&lt;text&gt;.AFTER_STR(&lt;string&gt;)</code>	<p>Returns the text that follows the first occurrence of &lt;string&gt;.</p> <p>If there is no match for &lt;string&gt;, the expression returns a text object of 0 length.</p> <p>Following is an example:</p> <pre>http.res.body(1024).after_str("start_string").before_str("e: ("https")</pre>
<code>&lt;text&gt;.BETWEEN(&lt;starting string&gt;, &lt;ending string&gt;)</code>	<p>Returns a Boolean TRUE value if the length of the text object is greater than or equal to the sum of the &lt;starting string&gt; and &lt;ending string&gt; argument lengths, and if a prefix of the target matches &lt;starting string&gt; and a suffix matches &lt;ending string&gt;.</p>
<code>&lt;text&gt;.PREFIX(&lt;prefix length&gt;)</code>	<p>Returns the starting string from a target block of text that contains the number of characters specified by the &lt;prefix length&gt; argument.</p> <p>If the &lt;prefix length&gt; argument exceeds the number of characters in the target, the expression returns a text object of 0 length.</p>
<code>&lt;text&gt;.SUFFIX(&lt;suffix length&gt;)</code>	<p>Returns the ending string from a target block of text that contains the number of characters specified by the &lt;suffix length&gt; argument. If the &lt;suffix length&gt; argument exceeds the number of characters in the target, the expression returns a text object of 0 length.</p>
<code>&lt;text&gt;.SUBSTR(&lt;string&gt;)</code>	<p>Select the first block of text in the target that matches the &lt;string&gt;.</p>
<code>&lt;text&gt;.SKIP(&lt;prefix length&gt;)</code>	<p>Selects the text in the target after skipping over a &lt;prefix length&gt; number of characters.</p> <p>If the entire target has fewer characters than &lt;prefix length&gt;, the entire target is returned.</p>
<code>&lt;text&gt;.STRIP_END_WS</code>	<p>Selects the text after removing white space from the end of the target.</p>
<code>&lt;text&gt;.STRIP_START_WS</code>	<p>Selects the text after removing white space from the beginning of the target.</p>
<code>&lt;text&gt;.UNQUOTE(&lt;character&gt;)</code>	<p>Selects the &lt;character&gt;, removes white space that immediately precedes and follows the &lt;character&gt;, and returns the remaining text. If the remaining text is quoted by &lt;character&gt;, this prefix also removes the quotes.</p> <p>For example, the operation UNQUOTE('"') changes the following text:</p> <pre>"abc xyz def "</pre> <p>To the following:</p> <pre>abc xyz def</pre>

## Operations for Comparing the Alphanumeric Order of Two Strings

The COMPARE operation examines the first nonmatching character of two different strings. This operation is based on lexicographic order, which is the method used when ordering terms in dictionaries.

This operation returns the arithmetic difference between the ASCII values of the first nonmatching characters in the compared strings. The following differences are examples:

- The difference between "abc" and "abd" is -1 (based on the third pair-wise character comparison).
- The difference between "@" and "abc" is -33.
- The difference between "1" and "abc" is -47.

Following is the syntax for the COMPARE operation.

```
<text>.COMPARE(<string>)
```

## Extracting an Integer from a String of Bytes That Represent Text

You can use the following functions to treat a string of bytes that represent text as a sequence of bytes, extract 8, 16, or 32 bits from the sequence, and then convert the extracted bits to an integer.

Table 3. Operations for Extracting an Integer from a String of Bytes That Represent Text

Function	Description
<code>&lt;text&gt;.GET_SIGNED8(&lt;n&gt;)</code>	Treats the string of bytes represented by text as a sequence of 8-bit signed integers and returns the integer at byte offset <i>n</i> . If the offset makes all or part of the value outside of the current text, an UNDEF condition is raised.
<code>&lt;text&gt;.GET_UNSIGNED8(&lt;n&gt;)</code>	Treats the string of bytes represented by text as a sequence of 8-bit unsigned integers and returns the integer at byte offset <i>n</i> . If the offset makes all or part of the value outside of the current text, an UNDEF condition is raised.
<code>&lt;text&gt;.GET_SIGNED16(&lt;n&gt;, &lt;endianness&gt;)</code>	<p>Treats the text string returned by the prefix as a string of bytes, extracts 16 bits starting at byte offset <i>n</i>, and converts the extracted bit sequence to a 16-bit signed integer. If the offset makes all or part of the value outside of the current text, an UNDEF condition is raised.</p> <p>The first parameter <i>n</i> is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second parameter, <i>endianness</i>, takes a mnemonic value of <code>LITTLE_ENDIAN</code> or <code>BIG_ENDIAN</code>.</p> <p>Note: In NetScaler 9.2, the parameter <i>n</i> was an index into an array of 16-bit items. In NetScaler 9.3, the parameter is a byte offset. Therefore, if you used this function in NetScaler 9.2, after you upgrade to NetScaler 9.3, you must change <i>n</i> to <i>2*n</i> to obtain the same results as you did earlier. For example, if the value of <i>n</i> before the upgrade was 4, you must change the value of <i>n</i> to 8. The parameter <i>endianness</i> also no longer takes the values that it did in NetScaler 9.2, which were 0 and 1. Instead, <i>endianness</i> accepts the mnemonic values mentioned earlier.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).GET_SIGNED16(8, BIG_ENDIAN)</pre>
<code>&lt;text&gt;.GET_UNSIGNED16(&lt;n&gt;, &lt;endianness&gt;)</code>	



	<p>Treats the text string returned by the prefix as a string of bytes, extracts 16 bits starting at byte offset <i>n</i>, and converts the extracted bit sequence to a 16-bit unsigned integer. If the offset makes all or part of the value outside of the current text, an <code>UNDEF</code> condition is raised.</p> <p>The first parameter <i>n</i> is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second parameter, <i>endianness</i>, takes a mnemonic value of <code>LITTLE_ENDIAN</code> or <code>BIG_ENDIAN</code>.</p> <p>Note: In NetScaler 9.2, the parameter <i>n</i> was an index into an array of 16-bit items. In NetScaler 9.3, the parameter is a byte offset. Therefore, if you used this function in NetScaler 9.2, after you upgrade to NetScaler 9.3, you must change <i>n</i> to <math>2 * n</math> to obtain the same results as you did earlier. For example, if the value of <i>n</i> before the upgrade was 4, you must change the value of <i>n</i> to 8. The parameter <i>endianness</i> also no longer takes the values that it did in NetScaler 9.2, which were 0 and 1. Instead, <i>endianness</i> accepts the mnemonic values mentioned earlier.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).GET_UNSIGNED16(8, LITTLE_ENDIAN)</pre>
<pre>&lt;text&gt;.GET_SIGNED32(&lt;n&gt;, &lt;endianness&gt;)</pre>	<p>Treats the text string returned by the prefix as a string of bytes, extracts 32 bits starting at byte offset <i>n</i>, and converts the extracted bit sequence to a 32-bit signed integer. If the offset makes all or part of the value outside of the current text, an <code>UNDEF</code> condition is raised.</p> <p>The first parameter <i>n</i> is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second parameter, <i>endianness</i>, takes a mnemonic value of <code>LITTLE_ENDIAN</code> or <code>BIG_ENDIAN</code>.</p> <p>Note: In NetScaler 9.2, the parameter <i>n</i> was an index into an array of 32-bit items. In NetScaler 9.3, the parameter is a byte offset. Therefore, if you used this function in NetScaler 9.2, after you upgrade to NetScaler 9.3, you must change <i>n</i> to <math>4 * n</math> to obtain the same results as you did earlier. For example, if the value of <i>n</i> before the upgrade was 4, you must change the value of <i>n</i> to 16. The parameter <i>endianness</i> also no longer takes the values that it did in NetScaler 9.2, which were 0 and 1. Instead, <i>endianness</i> accepts the mnemonic values mentioned earlier.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(1000).GET_SIGNED32(12, BIG_ENDIAN)</pre>
<pre>&lt;text&gt;.GET_UNSIGNED32(&lt;n&gt;, &lt;endianness&gt;)</pre>	<p>Treats the text string returned by the prefix as a string of bytes, extracts 32 bits starting at byte offset <i>n</i>, and returns the extracted bit sequence as part of a 64-bit unsigned long integer. If the offset makes all or part of the value outside of the current text, an <code>UNDEF</code> condition is raised.</p> <p>The first parameter <i>n</i> is the byte offset from the current position in the text string. Providing a byte offset enables the function to handle items that are not aligned on the boundaries that are required by indexes. The second parameter, <i>endianness</i>, takes a mnemonic value of <code>LITTLE_ENDIAN</code> or <code>BIG_ENDIAN</code>.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(1000).GET_UNSIGNED32(30, LITTLE_ENDIAN)</pre>

## Converting Text to a Hash Value

You can convert a text string to a hash value by using the `HASH` function. This function returns a 31-bit positive integer as a result of the operation. Following is the format of the expression:

`<text>.HASH`

This function ignores case and white spaces. For example, after the operation, the two strings `Ab c` and `a bc` would produce the same hash value.

## Encoding and Decoding Text by Applying the Base64 Encoding Algorithm

The following two functions encode and decode a text string by applying the Base64 encoding algorithm

Table 4. Functions for Encoding and Decoding a Text String by Using Base64 Encoding

Function	Description
<code>text.B64ENCODE</code>	Encodes the text string (designated by <code>text</code> ) by applying the Base64 encoding algorithm.
<code>text.B64DECODE</code>	Decodes the Base64-encoded string (designated by <code>text</code> ) by applying the Base64 decoding algorithm. The operation raises an <code>UNDEF</code> if <code>text</code> is not in B64-encoded format.

## Refining the Search in a Rewrite Action by Using the EXTEND Function

The `EXTEND` function is used in rewrite actions that specify patterns or pattern sets and target the bodies of HTTP packets. When a pattern match is found, the `EXTEND` function extends the scope of the search by a predefined number of bytes on both sides of the matching string. A regular expression can then be used to perform a rewrite on matches in this extended region. Rewrite actions that are configured with the `EXTEND` function perform rewrites faster than rewrite actions that evaluate entire HTTP bodies using only regular expressions.

The format of the `EXTEND` function is `EXTEND(m,n)`, where `m` and `n` are the number of bytes by which the scope of the search is extended before and after the matching pattern, respectively. When a match is found, the new search scope comprises `m` bytes that immediately precede the matching string, the string itself, and the `n` bytes that follow the string. A regular expression can then be used to perform a rewrite on a portion of this new string.

The `EXTEND` function can be used only if the rewrite action in which it is used fulfills the following requirements:

- The search is performed by using patterns or patterns sets (not regular expressions)
- The rewrite action evaluates only the bodies of HTTP packets.

Additionally, the `EXTEND` function can be used only with the following types of rewrite actions:

- `replace_all`
- `insert_after_all`
- `delete_all`
- `insert_before_all`

For example, you might want to delete all instances of `"http://exampleurl.com/"` and `"http://exampleurl.au/"` in the first 1000 bytes of the body. To do this, you can configure a rewrite action to search for all instances of the string `exampleurl`, extend the scope of the search on both sides of the string when a match is found, and then use a regular expression to perform the rewrite in the extended region. The following example extends the scope of the search by 20 bytes to the left and 50 bytes to the right of the matching string:

```
add rewrite action delurl_example delete_all 'HTTP.REQ.BODY(1000)' -pattern exampleurl -
refineSearch 'extend(20,50).regex_select(re#http://exampleurl.(com|au)#)'
```

## Converting Text to Hexadecimal Format

The following function converts text to hexadecimal format and extracts the resulting string:

`<text>.BLOB_TO_HEX(<string>)`

For example, this function converts the byte string `â€œabcâ€•` to `â€œ61:62:63â€•`.

## Encrypting and Decrypting Text

Updated: 2013-09-02

In default syntax expressions, you can use the `ENCRYPT` and `DECRYPT` functions to encrypt and decrypt text. Data encrypted by the `ENCRYPT` function on a given NetScaler appliance or high availability (HA) pair is intended for decryption by the `DECRYPT` function on the same NetScaler appliance or HA pair. The appliance supports the RC4, DES3, AES128, AES192, and AES256 encryption methods. The key value that is required for encryption is not user-specifiable. When an encryption method is set, the appliance automatically generates a random key value that is appropriate for the specified method. The default method is AES256 encryption, which is the most secure encryption method and the one that Citrix recommends.

You do not need to configure encryption unless you want to change the encryption method or you want the appliance to generate a new key value for the current encryption method.

Note: You can also encrypt and decrypt XML payloads. For information about the functions for encrypting and decrypting XML payloads, see ["Encrypting and Decrypting XML Payloads."](#)

## Configuring Encryption

Updated: 2013-09-02

During startup, the appliance runs the `set ns encryptionParams` command with, by default, the AES256 encryption method, and uses a randomly generated key value that is appropriate for AES256 encryption. The appliance also encrypts the key value and saves the command, with the encrypted key value, to the NetScaler configuration file. Consequently, the AES256 encryption method is enabled for the `ENCRYPT` and `DECRYPT` functions by default. The key value that is saved in the configuration file persists across reboots even though the appliance runs the command each time you restart it.

You can run the `set ns encryptionParams` command manually, or use the configuration utility, if you want to change the encryption method or if you want the appliance to generate a new key value for the current encryption method. To use the CLI to change the encryption method, set only the `method` parameter, as shown in **"Example 1: Changing the Encryption Method."** If you want the appliance to generate a new key value for the current encryption method, set the `method` parameter to the current encryption method and the `keyValue` parameter to an empty string (""), as shown in **"Example 2: Generating a New Key Value for the Current Encryption Method."** After you generate a new key value, you must save the configuration. If you do not save the configuration, the appliance uses the newly generated key value only until the next restart, after which it reverts to the key value in the saved configuration.

### To configure encryption by using the configuration utility

1. Navigate to System > Settings.
2. In the Settings area, click Change Encryption parameters.
3. In the Change Encryption Parameters dialog box, do one of the following:
  - o To change the encryption method, in the Method list, select the encryption method that you want.
  - o To generate a new key value for the current encryption method, click Generate a new key for the selected method.
4. Click OK.

## Using the ENCRYPT and DECRYPT Functions

You can use the `ENCRYPT` and `DECRYPT` functions with any expression prefix that returns text. For example, you can use the `ENCRYPT` and `DECRYPT` functions in rewrite policies for cookie encryption. In the following example, the rewrite actions encrypt a cookie named `MyCookie`, which is set by a back-end service, and decrypt the same cookie when it is returned by a client:

```
add rewrite action my-cookie-encrypt-action replace "HTTP.RES.SET_COOKIE.COOKIE(\"MyCookie\").VALUE(0)" "HTTP.RES.SET_COOKIE.COOKIE(\"MyCookie\").VALUE(0).ENCRYPT" -bypassSafetyCheck YES
```

```
add rewrite action my-cookie-decrypt-action replace "HTTP.REQ.COOKIE.VALUE(\"MyCookie\")" "HTTP.REQ.COOKIE.VALUE(\"MyCookie\").DECRYPT" -bypassSafetyCheck YES
```

After you configure policies for encryption and decryption, save the configuration to bring the policies into effect.

## Default Syntax Expressions: Working with Dates, Times, and Numbers

Most numeric data that the NetScaler appliance processes consists of dates and times. In addition to working with dates and times, the appliance processes other numeric data, such as the lengths of HTTP requests and responses. To process this data, you can configure default syntax expressions that process numbers.

A numeric expression consists of an expression prefix that returns a number and sometimes, but not always, an operator that can perform an operation on the number. Examples of expression prefixes that return numbers are `SYS.TIME.DAY`, `HTTP.REQ.CONTENT_LENGTH`, and `HTTP.RES.BODY.LENGTH`. Numeric operators can work with any prefix expression that returns data in numeric format. The `GT(<int>)` operator, for example, can be used with any prefix expression, such as `HTTP.REQ.CONTENT_LENGTH`, that returns an integer. Numeric expression prefixes and operators are also covered in ["Compound Operations for Numbers"](#) and ["Default Syntax Expressions: Parsing HTTP, TCP, and UDP Data."](#)

## Format of Dates and Times in an Expression

When configuring a default syntax expression in a policy that works with dates and times (for example, the NetScaler system time or a date in an SSL certificate), you specify a time format as follows:

```
GMT|LOCAL [<yyyy>] [<month>] [<d>] [<h>] [<m>] [<s>]
```

Where:

- <yyyy> is a four-digit year after GMT or LOCAL.
- <month> is a three-character abbreviation for the month, for example, Jan, Dec.
- <d> is a day of the week or an integer for the date.

You cannot specify the day as Monday, Tuesday, and so on. You specify either an integer for a specific day of the month, or you specify a date as the first, second, third weekday of the month, and so on.

Following are examples of specifying a day of the week:

Sun\_1 is the first Sunday of the month.

Sun\_3 is the third Sunday of the month.

Wed\_3 is the third Wednesday of the month.

30 is an example of an exact date in a month.

- <h> is the hour, for example, 10h.
- <s> is the number of seconds, for example, 30s.

The following example expression is true if the date is between 2008 Jan and 2009 Jan, based on GMT.

```
http.req.date.between(GMT 2008 Jan, GMT 2009 Jan)
```

The following example expression is true for March and all months that follow March in the calendar year, based on GMT:

```
sys.time.ge(GMT 2008 Mar)
```

When you specify a date and time, note that the format is case sensitive and must preserve the exact number of blank spaces between entries.

Note: In an expression that requires two time values, both must use GMT or both must use LOCAL. You cannot mix the two in an expression.

Note: Unlike when you use the SYS.TIME prefix in a default syntax expression, if you specify SYS.TIME in a rewrite action, the NetScaler returns a string in conventional date format (for example, Sun, 06 Nov 1994 08:49:37 GMT). For example, the following rewrite action replaces the http.res.date header with the NetScaler system time in a conventional date format:

```
add rewrite action sync_date replace http.res.date sys.time
```

## Expressions for the NetScaler System Time

The `SYS.TIME` expression prefix extracts the NetScaler system time. You can configure expressions that establish whether a particular event occurred at a particular time or within a particular time range according to the NetScaler system time.

The following table describes the expressions that you can create by using the `SYS.TIME` prefix.

Table 1. Expressions That Return NetScaler System Dates and Times

NetScaler Time Operation	Description
<code>SYS.TIME.BETWEEN</code> ( <code>&lt;time1&gt;</code> , <code>&lt;time2&gt;</code> )	<p>Returns a Boolean TRUE if the returned value is later than <code>&lt;time1&gt;</code> and earlier than <code>&lt;time2&gt;</code>.</p> <p>You format the <code>&lt;time1&gt;</code>, <code>&lt;time2&gt;</code> arguments as follows:</p> <ul style="list-style-type: none"> <li>They must both be GMT or both LOCAL.</li> <li><code>&lt;time2&gt;</code> must be later than <code>&lt;time1&gt;</code>.</li> </ul> <p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following:</p> <ul style="list-style-type: none"> <li><code>sys.time.between(GMT 2004, GMT 2006)</code></li> <li><code>sys.time.between(GMT 2004 Jan, GMT 2006 Nov)</code></li> <li><code>sys.time.between(GMT 2004 Jan, GMT 2006)</code></li> <li><code>sys.time.between(GMT 2005 May Sun_1, GMT 2005 May Sun_3)</code></li> <li><code>sys.time.between(GMT 2005 May 1, GMT May 2005 1)</code></li> <li><code>sys.time.between(LOCAL 2005 May 1, LOCAL May 2005 1)</code></li> </ul>
<code>SYS.TIME.DAY</code>	Returns the current day of the month as a number from 1 through 31.
<code>SYS.TIME.EQ(&lt;time&gt;)</code>	<p>Returns a Boolean TRUE if the current time is equal to the <code>&lt;time&gt;</code> argument.</p> <p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following (evaluation results are shown in parentheses):</p> <ul style="list-style-type: none"> <li><code>sys.time.eq(GMT 2005)</code> (TRUE in this example.)</li> <li><code>sys.time.eq(GMT 2005 Dec)</code> (FALSE in this example.)</li> <li><code>sys.time.eq(LOCAL 2005 May)</code> (Evaluates to TRUE or FALSE in this example, depending on the current time zone.)</li> <li><code>sys.time.eq(GMT 10h)</code> (TRUE in this example.)</li> <li><code>sys.time.eq(GMT 10h 30s)</code> (TRUE in this example.)</li> <li><code>sys.time.eq(GMT May 10h)</code> (TRUE in this example.)</li> <li><code>sys.time.eq(GMT Sun)</code> (TRUE in this example.)</li> <li><code>sys.time.eq(GMT May Sun_1)</code> (TRUE in this example.)</li> </ul>
<code>SYS.TIME.NE(&lt;time&gt;)</code>	Returns a Boolean TRUE if the current time is not equal to the <code>&lt;time&gt;</code> argument.
<code>SYS.TIME.GE(&lt;time&gt;)</code>	Returns a Boolean TRUE if the current time is later than or equal to <code>&lt;time&gt;</code> .

	<p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following (evaluation results are shown in parentheses):</p> <ul style="list-style-type: none"> <li>◦ <code>sys.time.ge(GMT 2004)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.ge(GMT 2005 Jan)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.ge(LOCAL 2005 May)</code> (TRUE or FALSE in this example, depending on the current time zone.)</li> <li>◦ <code>sys.time.ge(GMT 8h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.ge(GMT 30m)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.ge(GMT May 10h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.ge(GMT May 10h 0m)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.ge(GMT Sun)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.ge(GMT May Sun_1)</code> (TRUE in this example.)</li> </ul>
<code>SYS.TIME.GT(&lt;time&gt;)</code>	<p>Returns a Boolean TRUE if the time value is later than the &lt;time&gt; argument.</p> <p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following (evaluation results are shown in parentheses):</p> <ul style="list-style-type: none"> <li>◦ <code>sys.time.gt(GMT 2004)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.gt(GMT 2005 Jan)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.gt(LOCAL 2005 May)</code> (TRUE or FALSE, depending on the current time zone. )</li> <li>◦ <code>sys.time.gt(GMT 8h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.gt(GMT 30m)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.gt(GMT May 10h)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.gt(GMT May 10h 0m)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.gt(GMT Sun)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.gt(GMT May Sun_1)</code> (FALSE in this example.)</li> </ul>
<code>SYS.TIME.HOURS</code>	Returns the current hour as an integer from 0 to 23.
<code>SYS.TIME.LE(&lt;time&gt;)</code>	<p>Returns a Boolean TRUE if the current time value precedes or is equal to the &lt;time&gt; argument.</p> <p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following (evaluation results are shown in parentheses):</p> <ul style="list-style-type: none"> <li>◦ <code>sys.time.le(GMT 2006)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.le(GMT 2005 Dec)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.le(LOCAL 2005 May)</code> (TRUE or FALSE depending on the current timezone. )</li> <li>◦ <code>sys.time.le(GMT 8h)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.le(GMT 30m)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.le(GMT May 10h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.le(GMT Jun 11h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.le(GMT Wed)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.le(GMT May Sun_1)</code> (TRUE in this example.)</li> </ul>
<code>SYS.TIME.LT(&lt;time&gt;)</code>	Returns a Boolean TRUE if the current time value precedes the <time> argument.

	<p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following (evaluation results are shown in parentheses):</p> <ul style="list-style-type: none"> <li>◦ <code>sys.time.lt(GMT 2006)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.lt.time.lt(GMT 2005 Dec)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.lt(LOCAL 2005 May)</code> (TRUE or FALSE depending on the current time zone.)</li> <li>◦ <code>sys.time.lt(GMT 8h)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.lt(GMT 30m)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.lt(GMT May 10h)</code> (FALSE in this example.)</li> <li>◦ <code>sys.time.lt(GMT Jun 11h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.lt(GMT Wed)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.lt(GMT May Sun_1)</code> (FALSE in this example.)</li> </ul>
<code>SYS.TIME.MINUTES</code>	Returns the current minute as an integer from 0 to 59.
<code>SYS.TIME.MONTH</code>	Extracts the current month and returns an integer from 1 (January) to 12 (December).
<code>SYS.TIME.RELATIVE_BOOT</code>	<p>Calculates the number of seconds to the closest previous or scheduled reboot, and returns an integer.</p> <p>If the closest boot time is in the past, the integer is negative. If it is in the future, the integer is positive.</p>
<code>SYS.TIME.RELATIVE_NOW</code>	<p>Calculates the number of seconds between the current NetScaler system time and the specified time, and returns an integer showing the difference.</p> <p>If the designated time is in the past, the integer is negative; if it is in the future, the integer is positive.</p>
<code>SYS.TIME.SECONDS</code>	Extracts the seconds from the current NetScaler system time, and returns that value as an integer from 0 to 59.
<code>SYS.TIME.WEEKDAY</code>	Returns the current weekday as a value from 0 (Sunday) to 6 (Saturday).
<code>SYS.TIME.WITHIN</code> ( <code>&lt;time1&gt;</code> , <code>&lt;time2&gt;</code> )	<p>If you omit an element of time in <code>&lt;time1&gt;</code>, for example, the day or hour, it is assumed to have the lowest value in its range. If you omit an element in <code>&lt;time2&gt;</code>, it is assumed to have the highest value of its range.</p> <p>The ranges for the elements of time are as follows: month 1-12, day 1-31, weekday 0-6, hour 0-23, minutes 0-59 and seconds 0-59. If you specify the year, you must do so in both <code>&lt;time1&gt;</code> and <code>&lt;time2&gt;</code>.</p> <p>For example, if the time is GMT 2005 May 10 10h 15m 30s, and it is the second Tuesday of the month, you can specify the following (evaluation results are shown in parentheses):</p> <ul style="list-style-type: none"> <li>◦ <code>sys.time.within(GMT 2004, GMT 2006)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.within(GMT 2004 Jan, GMT 2006 Mar)</code> (FALSE, May is not in the range of January to March.)</li> <li>◦ <code>sys.time.within(GMT Feb, GMT)</code> (TRUE, May is in the range of February to December.)</li> </ul>



	<ul style="list-style-type: none"> <li>◦ <code>sys.time.within(GMT Sun_1, GMT Sun_3)</code> (TRUE, the second Tuesday is between the first Sunday and the third Sunday.)</li> <li>◦ <code>sys.time.within(GMT 2005 May 1 10h, GMT May 2005 1 17h)</code> (TRUE in this example.)</li> <li>◦ <code>sys.time.within(LOCAL 2005 May 1, LOCAL May 2005 1)</code> (TRUE or FALSE, depending on the NetScaler system time zone.)</li> </ul>
<code>SYS.TIME.YEAR</code>	Extracts the year from the current system time and returns that value as a four-digit integer.

## Expressions for SSL Certificate Dates

You can determine the validity period for SSL certificates by configuring an expression that contains the following prefix:

```
CLIENT.SSL.CLIENT_CERT
```

The following example expression matches a particular time for expiration with the information in the certificate:

```
client.ssl.client_cert.valid_not_after.eq(GMT 2009)
```

The following table describes time-based operations on SSL certificates. To obtain the expression you want, replace *certificate* in the expression in the first column with the prefix expression, `CLIENT.SSL.CLIENT_CERT`.

Table 1. Operations on Certificate (client.ssl.client\_cert) Dates and Times

SSL Certificate Operation	Description
<code>&lt;certificate&gt;.VALID_NOT_AFTER</code>	Returns the last day before certificate expiration. The return format is the number of seconds since GMT January 1, 1970 (0 hours, 0 minutes, 0 seconds).
<code>&lt;certificate&gt;.VALID_NOT_AFTER.BETWEEN (&lt;time1&gt;, &lt;time2&gt;)</code>	<p>Returns a Boolean TRUE value if the certificate validity is between the <code>&lt;time1&gt;</code> and <code>&lt;time2&gt;</code> arguments. Both <code>&lt;time1&gt;</code> and <code>&lt;time2&gt;</code> must be fully specified. Following are examples:</p> <p>GMT 1995 Jan is fully specified.</p> <p>GMT Jan is not fully specified</p> <p>GMT 1995 20 is not fully specified.</p> <p>GMT Jan Mon_2 is not fully specified.</p> <p>The <code>&lt;time1&gt;</code> and <code>&lt;time2&gt;</code> arguments must be both GMT or both LOCAL, and <code>&lt;time2&gt;</code> must be greater than <code>&lt;time1&gt;</code>.</p> <p>For example, if it is GMT 2005 May 1 10h 15m 30s, and the first Sunday of the month, you can specify the following (evaluation results are in parentheses).</p> <ul style="list-style-type: none"> <li><code>. . .between(GMT 2004, GMT 2006) (TRUE)</code></li> <li><code>. . .between(GMT 2004 Jan, GMT 2006 Nov) (TRUE)</code></li> <li><code>. . .between(GMT 2004 Jan, GMT 2006) (TRUE)</code></li> <li><code>. . .between(GMT 2005 May Sun_1, GMT 2005 May Sun_3) (TRUE)</code></li> <li><code>. . .between(GMT 2005 May 1, GMT May 2005 1) (TRUE)</code></li> <li><code>. . .between(LOCAL 2005 May 1, LOCAL May 2005 1) (TRUE or FALSE, depending on the NetScaler system time zone.)</code></li> </ul>
<code>&lt;certificate&gt;.VALID_NOT_AFTER.DAY</code>	

	<p>Extracts the last day of the month that the certificate is valid, and returns a number from 1 through 31, as appropriate for the date.</p>
<pre>&lt;certificate&gt;.VALID_NOT_AFTER.EQ(&lt;time&gt;)</pre>	<p>Returns a Boolean TRUE if the time is equal to the &lt;time&gt; argument.</p> <p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>○ . . .eq(GMT 2005) (TRUE)</li> <li>○ . . .eq(GMT 2005 Dec) (FALSE)</li> <li>○ . . .eq(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone)</li> <li>○ . . .eq(GMT 10h) (TRUE)</li> <li>○ . . .eq(GMT 10h 30s) (TRUE)</li> <li>○ . . .eq(GMT May 10h) (TRUE)</li> <li>○ . . .eq(GMT Sun) (TRUE)</li> <li>○ . . .eq(GMT May Sun_1) (TRUE)</li> </ul>
<pre>&lt;certificate&gt;.VALID_NOT_AFTER.GE(&lt;time&gt;)</pre>	<p>Returns a Boolean TRUE if the time value is greater than or equal to the argument &lt;time&gt;.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>○ . . .ge(GMT 2004) (TRUE)</li> <li>○ . . .ge(GMT 2005 Jan) (TRUE)</li> <li>○ . . .ge(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>○ . . .ge(GMT 8h) (TRUE)</li> <li>○ . . .ge(GMT 30m) (FALSE)</li> <li>○ . . .ge(GMT May 10h) (TRUE)</li> <li>○ . . .ge(GMT May 10h 0m) (TRUE)</li> <li>○ . . .ge(GMT Sun) (TRUE)</li> <li>○ . . .ge(GMT May Sun_1) (TRUE)</li> </ul>
<pre>&lt;certificate&gt;.VALID_NOT_AFTER.GT(&lt;time&gt;)</pre>	<p>Returns a Boolean TRUE if the time value is greater than the argument &lt;time&gt;.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>○ . . .gt(GMT 2004) (TRUE)</li> <li>○ . . .gt(GMT 2005 Jan) (TRUE)</li> <li>○ . . .gt(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>○ . . .gt(GMT 8h) (TRUE)</li> </ul>

	<ul style="list-style-type: none"> <li>◦ . . .gt(GMT 30m) (FALSE)</li> <li>◦ . . .gt(GMT May 10h) (FALSE)</li> <li>◦ . . .gt(GMT Sun) (FALSE)</li> <li>◦ . . .gt(GMT May Sun_1) (FALSE)</li> </ul>
<certificate>.VALID_NOT_AFTER.HOURS	Extracts the last hour that the certificate is valid and returns that value as an integer from 0 to 23.
<certificate>.VALID_NOT_AFTER.LE(<time>)	<p>Returns a Boolean TRUE if the time precedes or is equal to the &lt;time&gt; argument.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .le(GMT 2006) (TRUE)</li> <li>◦ . . .le(GMT 2005 Dec) (TRUE)</li> <li>◦ . . .le(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .le(GMT 8h) (FALSE)</li> <li>◦ . . .le(GMT 30m) (TRUE)</li> <li>◦ . . .le(GMT May 10h) (TRUE)</li> <li>◦ . . .le(GMT Jun 11h) (TRUE)</li> <li>◦ . . .le(GMT Wed) (TRUE)</li> <li>◦ . . .le(GMT May Sun_1) (TRUE)</li> </ul>
<certificate>.VALID_NOT_AFTER.LT(<time>)	<p>Returns a Boolean TRUE if the time precedes the &lt;time&gt; argument.</p> <p>For example, if the current time is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month, you can specify the following:</p> <ul style="list-style-type: none"> <li>◦ . . .lt(GMT 2006) (TRUE)</li> <li>◦ . . .lt(GMT 2005 Dec) (TRUE)</li> <li>◦ . . .lt(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .lt(GMT 8h) (FALSE)</li> <li>◦ . . .lt(GMT 30m) (TRUE)</li> <li>◦ . . .lt(GMT May 10h) (FALSE)</li> <li>◦ . . .lt(GMT Jun 11h) (TRUE)</li> <li>◦ . . .lt(GMT Wed) (TRUE)</li> <li>◦ . . .lt(GMT May Sun_1) (FALSE)</li> </ul>
<certificate>.VALID_NOT_AFTER.MINUTES	Extracts the last minute that the certificate is valid and returns that value as an integer from 0 to 59.
<certificate>.VALID_NOT_AFTER.MONTH	Extracts the last month that the certificate is valid and returns that value as an integer from 1 (January) to 12 (December).
<certificate>.VALID_NOT_AFTER.RELATIVE_BOOT	

	<p>Calculates the number of seconds to the closest previous or scheduled reboot and returns an integer. If the closest boot time is in the past, the integer is negative. If it is in the future, the integer is positive.</p>
<code>&lt;certificate&gt;.VALID_NOT_AFTER.RELATIVE_NOW</code>	<p>Calculates the number of seconds between the current system time and the specified time and returns an integer. If the time is in the past, the integer is negative; if it is in the future, the integer is positive.</p>
<code>&lt;certificate&gt;.VALID_NOT_AFTER.SECONDS</code>	<p>Extracts the last second that the certificate is valid and returns that value as an integer from 0 to 59.</p>
<code>&lt;certificate&gt;.VALID_NOT_AFTER.WEEKDAY</code>	<p>Extracts the last weekday that the certificate is valid. Returns a number between 0 (Sunday) and 6 (Saturday) to give the weekday in the time value.</p>
<code>&lt;certificate&gt;.VALID_NOT_AFTER.WITHIN(&lt;time1&gt; , &lt;time2&gt;)</code>	<p>Returns a Boolean TRUE if the time lies within all the ranges defined by the elements in &lt;time1&gt; and &lt;time2&gt;.</p> <p>If you omit an element of time from &lt;time1&gt;, it is assumed to have the lowest value in its range. If you omit an element from &lt;time2&gt;, it is assumed to have the highest value of its range. If you specify a year in &lt;time1&gt;, you must specify it in &lt;time2&gt;.</p> <p>The ranges for elements of time are as follows: month 1-12, day 1-31, weekday 0-6, hour 0-23, minutes 0-59 and seconds 0-59. For the result to be TRUE, each element in the time must exist in the corresponding range that you specify in &lt;time1&gt;, &lt;time2&gt;.</p> <p>For example, if time is GMT 2005 May 10 10h 15m 30s, and it is the second Tuesday of the month, you can specify the following (evaluation results are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .within(GMT 2004, GMT 2006) (TRUE)</li> <li>◦ . . .within(GMT 2004 Jan, GMT 2006 Mar) (FALSE, May is not in the range of January to March.)</li> <li>◦ . . .within(GMT Feb, GMT) (TRUE, May is in the range for February to December)</li> <li>◦ . . .within(GMT Sun_1, GMT Sun_3) (TRUE, the second Tuesday lies within the range of the first Sunday through the third Sunday)</li> <li>◦ . . .within(GMT 2005 May 1 10h, GMT May 2005 1 17h) (TRUE)</li> <li>◦ . . .within(LOCAL 2005 May 1, LOCAL May 2005 1) (TRUE or</li> </ul>

	FALSE, depending on the NetScaler system time zone)
<code>&lt;certificate&gt;.VALID_NOT_AFTER.YEAR</code>	Extracts the last year that the certificate is valid and returns a four-digit integer.
<code>&lt;certificate&gt;.VALID_NOT_BEFORE</code>	<p>Returns the date that the client certificate becomes valid.</p> <p>The return format is the number of seconds since GMT January 1, 1970 (0 hours, 0 minutes, 0 seconds).</p>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.BETWEEN (&lt;time1&gt;, &lt;time2&gt;)</code>	<p>Returns a Boolean TRUE if the time value is between the two time arguments. Both &lt;time1&gt; and &lt;time2&gt; arguments must be fully specified.</p> <p>Following are examples:</p> <ul style="list-style-type: none"> <li>◦ GMT 1995 Jan is fully specified.</li> <li>◦ GMT Jan is not fully specified.</li> <li>◦ GMT 1995 20 is not fully specified.</li> <li>◦ GMT Jan Mon_2 is not fully specified.</li> </ul> <p>The time arguments must be both GMT or both LOCAL, and &lt;time2&gt; must be greater than &lt;time1&gt;.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .between(GMT 2004, GMT 2006) (TRUE)</li> <li>◦ . . .between(GMT 2004 Jan, GMT 2006 Nov) (TRUE)</li> <li>◦ . . .between(GMT 2004 Jan, GMT 2006) (TRUE)</li> <li>◦ . . .between(GMT 2005 May Sun_1, GMT 2005 May Sun_3) (TRUE)</li> <li>◦ . . .between(GMT 2005 May 1, GMT May 2005 1) (TRUE)</li> <li>◦ . . .between(LOCAL 2005 May 1, LOCAL May 2005 1) (TRUE or FALSE, depending on the NetScaler system time zone.)</li> </ul>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.DAY</code>	Extracts the last day of the month that the certificate is valid and returns that value as a number from 1 through 31 representing that day.
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.EQ(&lt;time&gt;)</code>	<p>Returns a Boolean TRUE if the time is equal to the &lt;time&gt; argument.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you</p>

	<p>can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .eq(GMT 2005) (TRUE)</li> <li>◦ . . .eq(GMT 2005 Dec) (FALSE)</li> <li>◦ . . .eq(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .eq(GMT 10h) (TRUE)</li> <li>◦ . . .eq(GMT 10h 30s) (TRUE)</li> <li>◦ . . .eq(GMT May 10h) (TRUE)</li> <li>◦ . . .eq(GMT Sun) (TRUE)</li> <li>◦ . . .eq(GMT May Sun_1) (TRUE)</li> </ul>
<certificate>.VALID_NOT_BEFORE.GE(<time>)	<p>Returns a Boolean TRUE if the time is greater than (after) or equal to the &lt;time&gt; argument.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .ge(GMT 2004) (TRUE)</li> <li>◦ . . .ge(GMT 2005 Jan) (TRUE)</li> <li>◦ . . .ge(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .ge(GMT 8h) (TRUE)</li> <li>◦ . . .ge(GMT 30m) (FALSE)</li> <li>◦ . . .ge(GMT May 10h) (TRUE)</li> <li>◦ . . .ge(GMT May 10h 0m) (TRUE)</li> <li>◦ . . .ge(GMT Sun) (TRUE)</li> <li>◦ . . .ge(GMT May Sun_1) (TRUE)</li> </ul>
<certificate>.VALID_NOT_BEFORE.GT(<time>)	<p>Returns a Boolean TRUE if the time occurs after the &lt;time&gt; argument.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .gt(GMT 2004) (TRUE)</li> <li>◦ . . .gt(GMT 2005 Jan) (TRUE)</li> <li>◦ . . .gt(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .gt(GMT 8h) (TRUE)</li> <li>◦ . . .gt(GMT 30m) (FALSE)</li> <li>◦ . . .gt(GMT May 10h) (FALSE)</li> <li>◦ . . .gt(GMT May 10h 0m) (TRUE)</li> <li>◦ . . .gt(GMT Sun) (FALSE)</li> <li>◦ . . .gt(GMT May Sun_1) (FALSE)</li> </ul>
<certificate>.VALID_NOT_BEFORE.HOURS	<p>Extracts the last hour that the certificate is valid and returns that value as an integer from 0 to 23.</p>

<code>&lt;certificate&gt;.VALID_NOT_BEFORE.LE(&lt;time&gt;)</code>	<p>Returns a Boolean TRUE if the time precedes or is equal to the &lt;time&gt; argument.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .le(GMT 2006) (TRUE)</li> <li>◦ . . .le(GMT 2005 Dec) (TRUE)</li> <li>◦ . . .le(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .le(GMT 8h) (FALSE)</li> <li>◦ . . .le(GMT 30m) (TRUE)</li> <li>◦ . . .le(GMT May 10h) (TRUE)</li> <li>◦ . . .le(GMT Jun 11h) (TRUE)</li> <li>◦ . . .le(GMT Wed) (TRUE)</li> <li>◦ . . .le(GMT May Sun_1) (TRUE)</li> </ul>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.LT(&lt;time&gt;)</code>	<p>Returns a Boolean TRUE if the time precedes the &lt;time&gt; argument.</p> <p>For example, if the time value is GMT 2005 May 1 10h 15m 30s, and it is the first Sunday of the month of May in 2005, you can specify the following (evaluation results for this example are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .lt(GMT 2006) (TRUE)</li> <li>◦ . . .lt(GMT 2005 Dec) (TRUE)</li> <li>◦ . . .lt(LOCAL 2005 May) (TRUE or FALSE, depending on the current time zone.)</li> <li>◦ . . .lt(GMT 8h) (FALSE)</li> <li>◦ . . .lt(GMT 30m) (TRUE)</li> <li>◦ . . .lt(GMT May 10h) (FALSE)</li> <li>◦ . . .lt(GMT Jun 11h) (TRUE)</li> <li>◦ . . .lt(GMT Wed) (TRUE)</li> <li>◦ . . .lt(GMT May Sun_1) (FALSE)</li> </ul>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.MINUTES</code>	<p>Extracts the last minute that the certificate is valid. Returns the current minute as an integer from 0 to 59.</p>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.MONTH</code>	<p>Extracts the last month that the certificate is valid. Returns the current month as an integer from 1 (January) to 12 (December).</p>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.RELATIVE_BOOT</code>	<p>Calculates the number of seconds to the closest previous or scheduled NetScaler reboot and returns an integer. If the closest boot time is in the past, the integer is negative; if it is in the future, the integer is positive.</p>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.RELATIVE_NOW</code>	<p>Returns the number of seconds between the current NetScaler system time and the specified time as an integer. If the</p>



	designated time is in the past, the integer is negative. If it is in the future, the integer is positive.
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.SECONDS</code>	Extracts the last second that the certificate is valid. Returns the current second as an integer from 0 to 59.
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.WEEKDAY</code>	Extracts the last weekday that the certificate is valid. Returns the weekday as a number between 0 (Sunday) and 6 (Saturday).
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.WITHIN (&lt;time1&gt;, &lt;time2&gt;)</code>	<p>Returns a Boolean TRUE if each element of time exists within the range defined in the &lt;time1&gt;, &lt;time2&gt; arguments.</p> <p>If you omit an element of time from &lt;time1&gt;, it is assumed to have the lowest value in its range. If you omit an element of time from &lt;time2&gt;, it is assumed to have the highest value in its range. If you specify a year in &lt;time1&gt;, it must be specified in &lt;time2&gt;. The ranges for elements of time are as follows: month 1-12, day 1-31, weekday 0-6, hour 0-23, minutes 0-59 and seconds 0-59.</p> <p>For example, if the time is GMT 2005 May 10 10h 15m 30s, and it is the second Tuesday of the month, you can specify the following (evaluation results are in parentheses):</p> <ul style="list-style-type: none"> <li>◦ . . .within(GMT 2004, GMT 2006) (TRUE)</li> <li>◦ . . .within(GMT 2004 Jan, GMT 2006 Mar) (FALSE, May is not in the range of January to March.)</li> <li>◦ . . .within(GMT Feb, GMT) (TRUE, May is in the range of February to December.)</li> <li>◦ . . .within(GMT Sun_1, GMT Sun_3) (TRUE, the second Tuesday is between the first Sunday and the third Sunday.)</li> <li>◦ . . .within(GMT 2005 May 1 10h, GMT May 2005 1 17h) (TRUE)</li> <li>◦ . . .within(LOCAL 2005 May 1, LOCAL May 2005 1) (TRUE or FALSE, depending on the NetScaler system time zone)</li> </ul>
<code>&lt;certificate&gt;.VALID_NOT_BEFORE.YEAR</code>	Extracts the last year that the certificate is valid. Returns the current year as a four-digit integer.

## Expressions for HTTP Request and Response Dates

The following expression prefixes return the contents of the HTTP Date header as text or as a date object. These values can be evaluated as follows:

- As a number. The numeric value of an HTTP Date header is returned in the form of the number of seconds since Jan 1 1970.

For example, the expression `http.req.date.mod(86400)` returns the number of seconds since the beginning of the day. These values can be evaluated using the same operations as other non-date-related numeric data. For more information, see ["Expression Prefixes for Numeric Data Other Than Date and Time."](#)

- As an HTTP header. Date headers can be evaluated using the same operations as other HTTP headers.

For more information, see ["Default Syntax Expressions: Parsing HTTP, TCP, and UDP Data."](#)

- As text. Date headers can be evaluated using the same operations as other strings.

For more information, see ["Default Syntax Expressions: Evaluating Text."](#)

Table 1. Prefixes That Evaluate HTTP Date Headers

Prefix	Description
HTTP.REQ. DATE	Returns the contents of the HTTP Date header as text or as a date object. The date formats recognized are:  RFC822. Sun, 06 Jan 1980 08:49:37 GMT  RFC850. Sunday, 06-Jan-80 09:49:37 GMT  ASCTIME. Sun Jan 6 08:49:37 1980
HTTP.RES. DATE	Returns the contents of the HTTP Date header as text or as a date object. The date formats recognized are:  RFC822. Sun, 06 Jan 1980 8:49:37 GMT  RFC850. Sunday, 06-Jan-80 9:49:37 GMT  ASCTIME. Sun Jan 6 08:49:37 1980

## Generating the Day of the Week, as a String, in Short and Long Formats

The functions, `WEEKDAY_STRING_SHORT` and `WEEKDAY_STRING`, generate the day of the week, as a string, in short and long formats, respectively. The strings that are returned are always in English. The prefix used with these functions must return the day of the week in integer format and the acceptable range for the value returned by the prefix is 0-6. Therefore, you can use any prefix that returns an integer in the acceptable range. An `UNDEF` condition is raised if the returned value is not in this range or if memory allocation fails.

Following are the descriptions of the functions:

Table 1. Functions That Generate the Day of the Week, as a String, in Short and Long Formats

Function	Description
<code>&lt;prefix&gt;. WEEKDAY_STRING_SHORT</code>	Returns the day of the week in short format. The short form is always 3 characters long with an initial capital and the remaining characters in lower case. For example, <code>SYS.TIME.WEEKDAY.WEEKDAY_STRING_SHORT</code> returns <code>Sun</code> if the value returned by the <code>WEEKDAY</code> function is 0 and <code>Sat</code> if the value returned by the prefix is 6.
<code>&lt;prefix&gt;.WEEKDAY_STRING</code>	Returns the day of the week in long format. The long form always has an initial capital, with the remaining characters in lower case. For example, <code>SYS.TIME.WEEKDAY.WEEKDAY_STRING</code> returns <code>Sunday</code> if the value returned by the <code>WEEKDAY</code> function is 0 and <code>Saturday</code> if the value returned by the prefix is 6.

## Expression Prefixes for Numeric Data Other Than Date and Time

In addition to configuring expressions that operate on time, you can configure expressions for the following types of numeric data:

- The length of HTTP requests, the number of HTTP headers in a request, and so on.

For more information, see "[Expressions for Numeric HTTP Payload Data Other Than Dates.](#)"

- IP and MAC addresses.

For more information, see "[Expressions for IP Addresses and IP Subnets.](#)"

- Client and server data in regard to interface IDs and transaction throughput rate.

For more information, see "[Expressions for Numeric Client and Server Data.](#)"

- Numeric data in client certificates other than dates.

For information on these prefixes, including the number of days until certificate expiration and the encryption key size, see "[Prefixes for Numeric Data in SSL Certificates.](#)"

## Converting Numbers to Text

The following functions produce binary strings from a number returned by an expression prefix. These functions are particularly useful in the TCP rewrite feature as replacement strings for binary data. For more information about the TCP rewrite feature, see "Rewrite."

All the functions return a value of type `text`. The `endianness` that some of the functions accept as a parameter is either `LITTLE_ENDIAN` or `BIG_ENDIAN`.

Table 1. Functions That Produce a Binary String From a Number

Function	Description
<code>&lt;number&gt;.SIGNED8_STRING</code>	<p>Produces an 8-bit signed binary string represented.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).GET_SIGNED8(1)</pre>
<code>&lt;number&gt;.UNSIGNED8_STRING</code>	<p>Produces an 8-bit unsigned binary string represented.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).GET_UNSIGNED8</pre>
<code>&lt;number&gt;.SIGNED16_STRING(&lt;endianness&gt;)</code>	<p>Produces a 16-bit signed binary string represented.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).SKIP(12).GET_ (BIG_ENDIAN)</pre>
<code>&lt;number&gt;.UNSIGNED16_STRING(&lt;endianness&gt;)</code>	<p>Produces a 16-bit unsigned binary string represented.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).GET_UNSIGNED1 (LITTLE_ENDIAN)</pre>
<code>&lt;number&gt;.SIGNED32_STRING(&lt;endianness&gt;)</code>	<p>Produces a 32-bit signed binary string represented.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).AFTER_STR("de SIGNED32_STRING(BIG_ENDIAN)</pre>
<code>&lt;unsigned_long_number&gt;.UNSIGNED8_STRING</code>	<p>Produces an 8-bit unsigned binary string represented.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).GET_UNSIGNED8 UNSIGNED8_STRING</pre>
<code>&lt;unsigned_long_number&gt;.UNSIGNED16_STRING(&lt;endianness&gt;)</code>	<p>Produces a 16-bit unsigned binary string represented.</p> <p><b>Example</b></p>

	<pre>HTTP.REQ.BODY(100).GET_UNSIGNED16_STRING(LITTLE_ENDIAN)</pre>
<pre>&lt;unsigned_long_number&gt;.UNSIGNED32_STRING(&lt;endianness&gt;)</pre>	<p>Produces a 32-bit unsigned binary string reference. If the string is not 32 bits long, an exception is raised.</p> <p><b>Example</b></p> <pre>HTTP.REQ.BODY(100).AFTER_STR("de").UNSIGNED32_STRING(BIG_ENDIAN)</pre>

## Virtual Server Based Expressions

The `SYS.VSERVER(<vserver-name>)` expression prefix enables you to identify a virtual server. You can use the following functions with this prefix to retrieve information related to the specified virtual server:

- **THROUGHPUT.** Returns the throughput of the virtual server in Mbps (Megabits per second). The value returned is an unsigned long number.

**Usage:** `SYS.VSERVER("vserver").THROUGHPUT`

- **CONNECTIONS.** Returns the number of connections being managed by the virtual server. The value returned is an unsigned long number.

**Usage:** `SYS.VSERVER("vserver").CONNECTIONS`

- **STATE.** Returns the state of the virtual server. The value returned is `UP`, `DOWN`, or `OUT_OF_SERVICE`. One of these values can therefore be passed as an argument to the `EQ()` operator to perform a comparison that results in a Boolean `TRUE` or `FALSE`.

**Usage:** `SYS.VSERVER("vserver").STATE`

- **HEALTH.** Returns the percentage of services in an `UP` state for the specified virtual server. The value returned is an integer.

**Usage:** `SYS.VSERVER("vserver").HEALTH`

- **RESPTIME.** Returns the response time as an integer representing the number of microseconds. Response time is the average TTFB (Time To First Byte) from all the services bound to the virtual server.

**Usage:** `SYS.VSERVER("vserver").RESPTIME`

- **SURGECOUNT.** Returns the number of requests in the surge queue of the virtual server. The value returned is an integer.

**Usage:** `SYS.VSERVER("vserver").SURGECOUNT`

### Example 1

The following rewrite policy aborts rewrite processing if the number of connections at the load balancing virtual server `LBvserver` exceeds 10000:

```
add rewrite policy norewrite_pol sys.vserver("LBvserver").connections.gt(10000) norewrite
```

### Example 2

The following rewrite action inserts a custom header, `TP`, whose value is the throughput at the virtual server `LBvserver`:

```
add rewrite action tp_header insert_http_header TP SYS.VSERVER("LBvserver").THROUGHPUT
```

### Example 3

The following audit log message action writes the average TTFB of the services bound to a virtual server, to the `newslog` log file:

```
add audit messageaction log_vserver_resptime_act INFORMATIONAL "\"NS Response Time to Servers:\" + sys.vserver(\"ssl1b\").resptime + \" millisec\"" -logtoNewslog YES -bypassSafetyCheck YES
```

## Default Syntax Expressions: Parsing HTTP, TCP, and UDP Data

You can configure default syntax expressions to evaluate and process the payload in HTTP requests and responses. The payload associated with an HTTP connection includes the various HTTP headers (both standard and custom headers), the body, and other connection information such as the URL. Additionally, you can evaluate and process the payload in a TCP or UDP packet. For HTTP connections, for example, you can check whether a particular HTTP header is present or if the URL includes a particular query parameter.

You can configure expressions to transform the URL encoding and apply HTML or XML *escape* coding for subsequent evaluation. You can also use XPATH and JSON prefixes to evaluate data in XML and JSON files, respectively.

You can also use text-based and numeric default syntax expressions to evaluate HTTP request and response data. For more information, see ["Default Syntax Expressions: Evaluating Text"](#) and ["Default Syntax Expressions: Working with Dates, Times, and Numbers."](#)



## About Evaluating HTTP and TCP Payload

The payload of an HTTP request or response consists of HTTP protocol information such as headers, a URL, body content, and version and status information. When you configure a default syntax expression to evaluate HTTP payload, you use a default syntax expression prefix and, if necessary, an operator.

For example, you use the following expression, which includes the `http.req.header()` prefix and the `exists` operator, if you want to determine whether an HTTP connection includes a custom header named `myHeader`:

```
http.req.header("myHeader").exists
```

You can also combine multiple default syntax expressions with Boolean and arithmetic operators. For example, the following compound expression could be useful with various NetScaler features, such as Integrated Caching, Rewrite, and Responder. This expression first uses the `&&` Boolean operator to determine whether an HTTP connection includes the Content-Type header with a value of "text/html." If that operation returns a value of FALSE, the expression determines whether the HTTP connection includes a "Transfer-Encoding" or "Content-Length" header.

```
(http.req.header("Content-Type").exists && http.req.header("Content-Type").eq  
("text/html")) || (http.req.header("Transfer-Encoding").exists) || (http.req.header  
("Content-Length").exists)
```

The payload of a TCP or UDP packet is the data portion of the packet. You can configure default syntax expressions to examine features of a TCP or UDP packet, including the following:

- Source and destination domains
- Source and destination ports
- The text in the payload
- Record types

The following expression prefixes extract text from the body of the payload:

- `HTTP.REQ.BODY(integer)`. Returns the body of an HTTP request as a multiline text object, up to the character position designated in the *integer* argument. If there are fewer characters in the body than is specified in the argument, the entire body is returned.
- `HTTP.RES.BODY(integer)`. Returns a portion of the HTTP response body. The length of the returned text is equal to the number in the *integer* argument. If there are fewer characters in the body than is specified in integer, the entire body is returned.
- `CLIENT.TCP.PAYLOAD(integer)`. Returns TCP payload data as a string, starting with the first character in the payload and continuing for the number of characters in the *integer* argument.

Following is an example that evaluates to TRUE if a response body of 1024 bytes contains the string `https`, and this string occurs after the string `start string` and before the string `end string`:

```
http.res.body(1024).after_str("start_string").before_str("end_string").contains("https")
```

Note: You can apply any text operation to the payload body. For information on operations that you can apply to text, see "Default Syntax Expressions: Evaluating Text."

## Expressions for Identifying the Protocol in an Incoming IP Packet

The following table lists the expressions that you can use to identify the protocol in an incoming packet.

Expression	Description
CLIENT.IP.PROTOCOL	Identifies the protocol in IPv4 packets sent by clients.
CLIENT.IPV6.PROTOCOL	Identifies the protocol in IPv6 packets sent by clients.
SERVER.IP.PROTOCOL	Identifies the protocol in IPv4 packets sent by servers.
SERVER.IPV6.PROTOCOL	Identifies the protocol in IPv6 packets sent by servers.

## Arguments to the PROTOCOL function

You can pass the Internet Assigned Numbers Authority (IANA) protocol number to the `PROTOCOL` function. For example, if you want to determine whether the protocol in an incoming packet is TCP, you can use `CLIENT.IP.PROTOCOL.EQ(6)`, where 6 is the IANA-assigned protocol number for TCP. For some protocols, you can pass an enumeration value instead of the protocol number. For example, instead of `CLIENT.IP.PROTOCOL.EQ(6)`, you can use `CLIENT.IP.PROTOCOL.EQ(TCP)`. The following table lists the protocols for which you can use enumeration values, and the corresponding enumeration values for use with the `PROTOCOL` function.

Protocol	Enumeration value
Transmission Control Protocol (TCP)	TCP
User Datagram Protocol (UDP)	UDP
Internet Control Message Protocol (ICMP)	ICMP
IP Authentication Header (AH), for providing authentication services in IPv4 and IPv6	AH
Encapsulating Security Payload (ESP) protocol	ESP
General Routing Encapsulation (GRE)	GRE
IP-within-IP Encapsulation Protocol	IPIP
Internet Control Message Protocol for IPv6 (ICMPv6)	ICMPv6
Fragment Header for IPv6	FRAGMENT

## Use Case Scenarios

The protocol expressions can be used in both request-based and response-based policies. You can use the expressions in various NetScaler features, such as load balancing, WAN optimization, content switching, rewrite, and listen policies. You can use the expressions with functions such as `EQ()` and `NE()`, to identify the protocol in a policy and perform an action.

Following are some use cases for the expressions:

- In Branch Repeater load balancing configurations, you can use the expressions in a listen policy for the wildcard virtual server. For example, you can configure the wildcard virtual server with the listen policy `CLIENT.IP.PROTOCOL.EQ(TCP)` so that the virtual server processes only TCP traffic and simply bridges all non-TCP traffic. Even though you can use an Access Control List instead of the listen policy, the listen policy provides better control over what traffic is processed.
- For content switching virtual servers of type `ANY`, you can configure content switching policies that switch requests on the basis of the protocol in incoming packets. For example, you can configure content switching policies to direct all TCP traffic to one load balancing virtual server and all non-TCP traffic to another load balancing virtual server.
- You can use the client-based expressions to configure persistence based on the protocol. For example, you can use `CLIENT.IP.PROTOCOL` to configure persistence on the basis of the protocols in incoming IPv4 packets.

## Expressions for HTTP and Cache-Control Headers

One common method of evaluating HTTP traffic is to examine the headers in a request or a response. A header can perform a number of functions, including the following:

- Provide cookies that contain data about the sender.
- Identify the type of data that is being transmitted.
- Identify the route that the data has traveled (the Via header).

Note: Note that if an operation is used to evaluate both header and text data, the header-based operation always overrides the text-based operation. For example, the `AFTER_STR` operation, when applied to a header, overrides text-based `AFTER_STR` operations for all instances of the current header type.

## Prefixes for HTTP Headers

The following table describes expression prefixes that extract HTTP headers.

Table 1. Prefixes That Extract HTTP Headers

HTTP Header Prefix	Description
<code>HTTP.REQ.HEADER( "&lt;header_name&gt;" )</code>	Returns the contents of the HTTP header. The header name cannot exceed 32 characters.  Note that this prefix returns the header value, so you need to typecast it as follows:  <code>http.req.header( "host " )</code>  For more information on typecasting, see <a href="#">Typecasting</a> .
<code>HTTP.REQ.FULL_HEADER</code>	Returns the contents of the complete HTTP request, including the "GET /brochures/index.html HTTP/1.1" line.
<code>HTTP.REQ.DATE</code>	Returns the contents of the HTTP Date header. The header value must conform to RFC822, Sun, 06 Jan 1980 08:49:39 GMT, or RFC850, Sunday, 06-Jan-80 08:49:39 GMT, or ASCII TIME, Sun Jan 6 08:49:39 GMT.  To evaluate a Date header as a date, use the <a href="#">Date and Numbers</a> operation.
<code>HTTP.REQ.COOKIE</code>	(Name/Value List) Returns the contents of the HTTP Cookie header.
<code>HTTP.REQ.TXID</code>	Returns the HTTP transaction ID and system MAC address.
<code>HTTP.RES.HEADER( "&lt;header_name&gt;" )</code>	Returns the contents of the HTTP response header. The header name cannot exceed 32 characters.
<code>HTTP.RES.FULL_HEADER</code>	Returns the contents of the complete HTTP response, including the "HTTP/1.1 200 OK" line and the body.
<code>HTTP.RES.SET_COOKIE</code> or <code>HTTP.RES.SET_COOKIE2</code>	Returns the HTTP Set-Cookie header.

<pre>HTTP.RES.SET_COOKIE( "&lt;name&gt;" )</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2( "&lt;name&gt;" )</pre>	<p>Returns the cookie of the specified name. Returns UNDEF if more than one cookie is found in these headers.</p>
<pre>HTTP.RES.SET_COOKIE( "&lt;name&gt;" ).DOMAIN</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2( "&lt;name&gt;" ).DOMAIN</pre>	<p>Returns the value of the first Domain header. For example, if Customer = "ABC"; DOMAIN=example.com, the expression <code>http.res.set_cookie.cookie( "Customer" ).DOMAIN</code> returns a string of zero length.</p>
<pre>HTTP.RES.SET_COOKIE.EXISTS( "&lt;name&gt;" )</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.EXISTS( "&lt;name&gt;" )</pre>	<p>Returns a Boolean TRUE if a cookie header is found. This prefix returns UNDEF if no cookie header is found in the first 15 headers.</p>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;" ).EXPIRES</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;" ).EXPIRES</pre>	<p>Returns the Expires field of the cookie object, or as text. If multiple Expires headers are present, a text object of length 100 is returned. To evaluate the returned value, see <a href="#">Times, and Numbers.</a></p>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;" ).PATH   PATH.GET( n )</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;" ).PATH   PATH.GET( n )</pre>	<p>Returns the value of Path field. Backslash and slash are treated as single slash and returned.</p> <p>For example, the following is a Set-Cookie header:</p> <pre>Set-Cookie : Customer = ABC; Path = /a/b/c;</pre> <p>The following expression returns the value of Path field:</p> <pre>http.res.set_cookie.cookie( "Customer" ).PATH   PATH.GET( 1 )</pre> <p>The following expression returns the value of Path field:</p> <pre>http.res.set_cookie.cookie( "Customer" ).PATH   PATH.GET( 1 )</pre> <p>Quotes are stripped from the returned value if the quote is absent.</p>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;" ).PATH.IGNORE_EMPTY_ELEMENTS</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;" ).PATH.IGNORE_EMPTY_ELEMENTS</pre>	<p> Ignores the empty elements in the list, and the list has an empty element.</p> <p>As another example, in the following expression returns a list of the values of the Path field:</p> <pre>http.req.header( "Customer" ).PATH   PATH.GET( 1 )</pre> <p>The following expression returns the value of Path field:</p> <pre>http.req.header( "Customer" ).PATH   PATH.GET( 1 )</pre>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;" ).PORT</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;" ).PORT</pre>	<p>Returns the value of Port field.</p> <p>For example, the following expression returns the value of Port field:</p> <pre>"/a/b/c"; PORT= "80, 2580":</pre>



<p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).PATH. IGNORE_EMPTY_ELEMENTS</pre>	<p>As another example, in the following expression returns a</p> <pre>http.req.header( "Cust_ count</pre> <p>The following expression retur</p> <pre>http.req.header( "Cust_</pre>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).PORT</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).PORT</pre>	<p>Returns the value or values of following expression returns 8</p> <p>PORT= "80, 2580"</p> <pre>http.res.set_cookie.co</pre> <p>A string of zero length is return</p>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).PORT. IGNORE_EMPTY_ELEMENTS</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).PORT. IGNORE_EMPTY_ELEMENTS</pre>	<p>Ignores the empty elements in the list is , and the list has an empty element.</p> <p>As another example, in the following expression returns a</p> <pre>http.req.header( "Cust_ count</pre> <p>The following expression retur</p> <pre>http.req.header( "Cust_</pre>
<pre>HTTP.RES.SET_COOKIE.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).VERSION</pre> <p>or</p> <pre>HTTP.RES.SET_COOKIE2.COOKIE( "&lt;name&gt;", &lt;integer&gt; ).VERSION</pre>	<p>Returns the value of Version fi</p> <p>A string of zero length is return</p>
<pre>HTTP.RES.TXID</pre>	<p>Returns the HTTP transaction time and system MAC address:</p>

## Operations for HTTP Headers

The following table describes operations that you can specify with the prefixes for HTTP headers.

Table 2. Operations That Evaluate HTTP Headers

HTTP Header Operation	Description
<pre>http header .EXISTS</pre>	<p>Returns a Boolean TRUE if an instance of the specified header type exists.</p> <p>Following is an example:</p> <pre>http.req.header( "Cache-Control" ).exists</pre>
<pre>http header.CONTAINS" http header . CONTAINS ("&lt;string&gt;")</pre>	<p>Returns a Boolean TRUE if the &lt;string&gt; argument appears in any instance of the header value.</p> <p>Note: This operation overrides any text-based Contains operations on all instances of the current header type.</p> <p>Following is an example of request with two headers:</p> <pre>HTTP/1.1 200 OK\r\n</pre>

	<pre>MyHeader: abc\r\n Content-Length: 200\r\n MyHeader: def\r\n \r\n</pre> <p>The following returns a Boolean TRUE:</p> <pre>http.res.header("MyHeader").contains("de")</pre> <p>The following returns FALSE. Note that the NetScaler does not concatenate the different values.</p> <pre>http.res.header("MyHeader").contains("bcd")</pre>
<pre>http header .COUNT</pre>	<p>Returns the number of headers in a request or response, to a maximum of 15 headers of the same type. The result is undefined if there are more than 15 instances of the header.</p> <p>Following is sample data in a request:</p> <pre>HTTP/1.1 200 OK\r\n MyHeader: abc\r\n Content-Length: 200\r\n MyHeader: def\r\n \r\n</pre> <p>When evaluating the preceding request, the following returns a count of 2:</p> <pre>http.res.header("MyHeader").count</pre>
<pre>http header.AFTER_STR ("&lt;string&gt;")</pre>	<p>Extracts the text that follows the first occurrence of the &lt;string&gt; argument. The headers are evaluated from the last instance to the first.</p> <p>Following is an example of a request:</p> <pre>HTTP/1.1 200 OK\r\n MyHeader: 111abc\r\n Content-Length: 200\r\n MyHeader: 111def\r\n \r\n</pre> <p>The following extracts the string "def" from the last instance of MyHeader. This is value "111def."</p> <pre>http.res.header("MyHeader").after_str("111")</pre> <p>The following extracts the string "c" from the first instance of MyHeader. This is the value "abc111."</p> <pre>http.res.header("MyHeader").after_str("lab")</pre>
<pre>http header.BEFORE_STR ("&lt;string&gt;")</pre>	<p>Extracts the text that appears prior to the first occurrence of the input &lt;string&gt; argument. The headers are evaluated from the last instance to the first.</p>

	<p>Following is an example of a request that contains headers:</p> <pre>HTTP/1.1 200 OK\r\n MyHeader: abc111\r\n Content-Length: 200\r\n MyHeader: def111\r\n \r\n</pre> <p>The following extracts the string "def" from the last instance of MyHeader. This is the value "def111."</p> <pre>http.res.header("MyHeader").before_str("111")</pre> <p>The following extracts the string "a" from the first instance of MyHeader. This is the value "abc111."</p> <pre>http.res.header("MyHeader").before_str("bcl")</pre>
<pre>http header.INSTANCE (&lt;instance number&gt;)</pre>	<p>An HTTP header can occur multiple times in a request or a response. This operation returns the header that occurs &lt;instance number&gt; of places before the final instance. For example, instance(0) selects the last instance of the current type, instance(1) selects the next-to-last instance, and so on. This prefix cannot be used in bidirectional policies.</p> <p>The &lt;instance number&gt; argument cannot exceed 14. Following is an example of a request with two headers:</p> <pre>HTTP/1.1 200 OK\r\n MyHeader: abc\r\n Content-Length: 200\r\n MyHeader: def\r\n \r\n</pre> <p>The following returns a text object that refers to "MyHeader: abc\r\n":</p> <pre>http.res.header("MyHeader").instance(1)</pre>
<pre>http header.SUBSTR ("&lt;string&gt;")</pre>	<p>Extracts the text that matches the &lt;string&gt; argument. The headers are evaluated from the last instance to the first. Following is an example of a request with two headers that contain the string "111":</p> <pre>HTTP/1.1 200 OK\r\n MyHeader: abc111\r\n Content-Length: 200\r\n MyHeader: 111def\r\n \r\n</pre> <p>The following returns "111" from the last instance of MyHeader. This is the header with the value "111def."</p> <pre>http.res.header("MyHeader").substr("111")</pre>
<pre>http header.VALUE(&lt;instance number&gt;)</pre>	<p>An HTTP header can occur multiple times in a request or a response. VALUE(0) selects the value in the last instance, VALUE(1) selects</p>



the value in the next-to-last instance, and so on. The <instance number> argument cannot exceed 14.

Following is an example of a request with two headers:

```
HTTP/1.1 200 OK\r\n
MyHeader: abc\r\n
Content-Length: 200\r\n
MyHeader: def\r\n
\r\n
```

The following returns "abc\r\n":

```
http.res.header("MyHeader").value(1)
```

## Prefixes for Cache-Control Headers

The following prefixes apply specifically to Cache-Control headers.

Table 3. Prefixes That Extract Cache-Control Headers

HTTP Header Prefix	Description
HTTP.REQ.CACHE_CONTROL	Returns a Cache-Control header in an HTTP request.
HTTP.RES.CACHE_CONTROL	Returns a Cache-Control header in an HTTP response.

## Operations for Cache-Control Headers

You can apply any of the operations for HTTP headers to Cache-Control headers. For more information, see "Operations for HTTP Headers."

In addition, the following operations identify specific types of Cache-Control headers. See RFC 2616 for information about these header types.

Table 4. Operations That Evaluate Cache-Control Headers

HTTP Header Operation	Description
Cache-Control header.NAME (<integer>)	<p>Returns as a text value the name of the Cache-Control header that corresponds to the nth component in a name-value list, as specified by &lt;integer&gt;.</p> <p>The index of the name-value component is 0-based. If the &lt;integer&gt; that is specified by the integer argument is greater than the number of components in the list, a zero-length text object is returned.</p> <p>Following is an example:</p> <pre>http.req.cache_control.name(3).contains("some_text")</pre>
Cache-Control header.IS_INVALID	<p>Returns a Boolean TRUE if the Cache-Control header is not present in the request or response.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_invalid</pre>
Cache-Control header.IS_PRIVATE	<p>Returns a Boolean TRUE if the Cache-Control header has the value Private.</p>

	<p>Following is an example:</p> <pre>http.req.cache_control.is_private</pre>
Cache-Control header. IS_PUBLIC	<p>Returns a Boolean TRUE if the Cache-Control header has the value Private.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_public</pre>
Cache-Control header. IS_NO_STORE	<p>Returns a Boolean TRUE if the Cache-Control header has the value No-Store.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_no_store</pre>
Cache-Control header. IS_NO_CACHE	<p>Returns a Boolean TRUE if the Cache-Control header has the value No-Cache.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_no_cache</pre>
Cache-Control header. IS_MAX_AGE	<p>Returns a Boolean TRUE if the Cache-Control header has the value Max-Age.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_max_age</pre>
Cache-Control header. IS_MIN_FRESH	<p>Returns a Boolean TRUE if the Cache-Control header has the value Min-Fresh.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_min_fresh</pre>
Cache-Control header. IS_MAX_STALE	<p>Returns a Boolean TRUE if the Cache-Control header has the value Max-Stale.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_max_stale</pre>
Cache-Control header. IS_MUST_REVALIDATE	<p>Returns a Boolean TRUE if the Cache-Control header has the value Must-Revalidate.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_must_revalidate</pre>
Cache-Control header. IS_NO_TRANSFORM	<p>Returns a Boolean TRUE if the Cache-Control header has the value No-Transform.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_no_transform</pre>
Cache-Control header. IS_ONLY_IF_CACHED	<p>Returns a Boolean TRUE if the Cache-Control header has the value Only-If-Cached.</p>

	<p>Following is an example:</p> <pre>http.req.cache_control.is_only_if_cached</pre>
Cache-Control header. IS_PROXY_REVALIDATE	<p>Returns a Boolean TRUE if the Cache-Control header has the value Proxy-Revalidate.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_proxy_revalidate</pre>
Cache-Control header. IS_S_MAXAGE	<p>Returns a Boolean TRUE if the Cache-Control header has the value S-Maxage.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_s_maxage</pre>
Cache-Control header. IS_UNKNOWN	<p>Returns a Boolean TRUE if the Cache-Control header is of an unknown type.</p> <p>Following is an example:</p> <pre>http.req.cache_control.is_unknown</pre>
Cache-Control header. MAX_AGE	<p>Returns the value of the Cache-Control header Max-Age. If this header is absent or invalid, 0 is returned.</p> <p>Following is an example:</p> <pre>http.req.cache_control.max_age.le(3)</pre>
Cache-Control header. MAX_STALE	<p>Returns the value of the Cache-Control header Max-Stale. If this header is absent or invalid, 0 is returned.</p> <p>Following is an example:</p> <pre>http.req.cache_control.max_stale.le(3)</pre>
Cache-Control header. MIN_FRESH	<p>Returns the value of the Cache-Control header Min-Fresh. If this header is absent or invalid, 0 is returned.</p> <p>Following is an example:</p> <pre>http.req.cache_control.min_fresh.le(3)</pre>
Cache-Control header. S_MAXAGE	<p>Returns the value of the Cache-Control header S-Maxage. If this header is absent or invalid, 0 is returned.</p> <p>Following is an example:</p> <pre>http.req.cache_control.s_maxage.eq(2)</pre>

## Expressions for Extracting Segments of URLs

You can extract URLs and portions of URLs, such as the host name, or a segment of the URL path. For example, the following expression identifies HTTP requests for image files by extracting image file suffixes from the URL:

```
http.req.url.suffix.eq("jpeg") || http.req.url.suffix.eq("gif")
```

Most expressions for URLs operate on text and are described in "Expression Prefixes for Text in HTTP Requests and Responses." This section discusses the GET operation. The GET operation extracts text when used with the following prefixes:

- HTTP.REQ.URL.PATH
- VPN.BASEURL.PATH
- VPN.CLIENTLESS\_BASEURL.PATH

The following table describes prefixes for HTTP URLs.

Table 1. Prefixes That Extract URLs

URL Prefix	Description
HTTP.REQ.URL.PATH.GET(<n>)	<p>Returns a slash- (â€œ/â€•) separated list from the URL path. For example, consider the following URL:</p> <pre>http://www.mycompany.com/dir1/dir2/dir3/index.html?a=1</pre> <p>The following expression returns dir1 from this URL:</p> <pre>http.req.url.path.get(1)</pre> <p>The following expression returns dir2:</p> <pre>http.req.url.path.get(2)</pre>
HTTP.REQ.URL.PATH.GET_REVERSE(<n>)	<p>Returns a slash- (â€œ/â€•) separated list from the URL path, starting from the end of the path. For example, consider the following URL:</p> <pre>http://www.mycompany.com/dir1/dir2/dir3/index.html?a=1</pre> <p>The following expression returns index.html from this URL:</p> <pre>http.req.url.path.get_reverse(0)</pre> <p>The following expression returns dir3:</p> <pre>http.req.url.path.get_reverse(1)</pre>

## Expressions for HTTP Status Codes and Numeric HTTP Payload Data Other Than Dates

The following table describes prefixes for numeric values in HTTP data other than dates.

Table 1. Prefixes That Evaluate HTTP Request or Response Length

Prefix	Description
<code>HTTP.REQ.CONTENT_LENGTH</code>	<p>Returns the length of an HTTP request as a number.</p> <p>Following is an example:</p> <pre>http.req.content_length &lt; 500</pre>
<code>HTTP.RES.CONTENT_LENGTH</code>	<p>Returns the length of the HTTP response as a number.</p> <p>Following is an example:</p> <pre>http.res.content_length &lt;= 1000</pre>
<code>HTTP.RES.STATUS</code>	Returns the response status code
<code>HTTP.RES.IS_REDIRECT</code>	<p>Returns a Boolean <code>TRUE</code> if the response code is associated with a redirect. Following are the redirect response codes:</p> <ul style="list-style-type: none"> <li>300 (Multiple Choices)</li> <li>301 (Moved Permanently)</li> <li>302 (Found)</li> <li>303 (See Other)</li> <li>305 (Use Proxy)</li> <li>307 (Temporary Redirect)</li> </ul> <p>Note: Status code 304 is not considered a redirect HTTP response status code. Status code 306 is unused.</p> <p>In the following example, the rewrite action replaces <code>http</code> in the Location header of an HTTP response with <code>https</code> if the response is associated with an HTTP redirect.</p> <pre>add rewrite action redloc replace 'http.res.header ("Location").before_regex(re#://#)' '"https"'  add rewrite policy poll HTTP.RES.IS_REDIRECT red_location  bind rewrite global poll 100</pre>

## SIP Expressions

## Introduction

The NetScaler default expressions language contains a number of expressions that operate on Session Initiation Protocol (SIP) connections. These expressions are intended to be used in policies for any supported protocol that operates on a request/response basis. These expressions can be used in content switching, rate limiting, responder, and rewrite policies.

Certain limitations apply to SIP expressions used with responder policies. Only the DROP, NOOP or RESPONDWITH actions are allowed on a SIP load balancing virtual server. Responder policies can be bound to a load balancing virtual server, an override global bind point, a default global bind point, or a `sip_udp` policy label.

The header format used by the SIP protocol is similar to that used by the HTTP protocol, so many of the new expressions look and function much like their HTTP analogs. Each SIP header consists of a line that includes the SIP method, the URL, and the version, followed by a series of name-value pairs that look like HTTP headers.

Following is a sample SIP header that is referred to in the expressions tables beneath it:

```
INVITE sip:16@www.sip.com:5060;transport=udp SIP/2.0
Record-Route: <sip:200.200.100.22;lr=on>
Via: SIP/2.0/UDP 200.200.100.22;branch=z9hG4bK444b.c8e103d1.0;rport=5060;
    received=10.102.84.18
Via: SIP/2.0/UDP 10.102.84.180:5060;branch=z9hG4bK03e76d0b;rport=5060;
    received=10.102.84.160
From: "12" <sip:12@sip_example.com>;tag=00127f54ec85a6d90cc14f45-53cc0185
To: "16" <sip:16@sip_example.com>;tag=00127f54ec85a6d90cc14f45-53cc0185
Call-ID: 00127f54-ec850017-0e46f5b9-5ec149c2@10.102.84.180
Max-Forwards: 69CSeq: 101 INVITE
User-Agent: Cisco-CP7940G/8.0
Contact: <sip:12@10.102.84.180:5060;transport=udp>
Expires: 180
Accept: application/sdp
Allow: ACK,BYE,CANCEL,INVITE,NOTIFY,OPTIONS,REFER,REGISTER,UPDATE
Supported: replaces,join,norefersub
Content-Length: 277
Content-Type: application/sdp
Content-Disposition: session;handling=optiona
```

## SIP Reference Tables

The following tables contain lists of expressions that operate on SIP headers. The first table contains expressions that apply to request headers. Most response-based expressions are nearly the same as the corresponding request-based expressions. To create a response expression from the corresponding request expression, you change the first two sections of the expression from `SIP.REQ` to `SIP.RES`, and make other obvious adjustments. The second table contains those response expressions that are unique to responses and have no request equivalents. You can use any element in the following tables as a complete expression on its own, or you can use various operators to combine these expression elements with others to form more complex expressions.

Table 1. SIP Request Expressions

Expression	Description
<code>SIP.REQ.METHOD</code>	Operates on the request method of the request. If the request is a MESSAGE, NOTIFY, or INFO, the expression returns the request itself. If the request is a derivative of the template request of INVITE, the expression returns the request method.
<code>SIP.REQ.URL</code>	Operates on the request URI of the request. The expressions <code>SIP.REQ.URL</code> and <code>SIP.REQ.URL.PROTOCOL</code> are applicable to the request URI of the request. If the request is a MESSAGE, NOTIFY, or INFO, the expression returns the request URI. If the request is a derivative of the template request of INVITE, the expression returns the request URI.
<code>SIP.REQ.URL.PROTOCOL</code>	Returns the URL protocol of the request URI. If the request is a MESSAGE, NOTIFY, or INFO, the expression returns <code>sip</code> . If the request is a derivative of the template request of INVITE, the expression returns <code>sip</code> .

SIP.REQ.URL.HOSTNAME	Returns the hostn transport=udp
SIP.REQ.URL.HOSTNAME.PORT	Returns the port p For example, for a
SIP.REQ.URL.HOSTNAME.DOMAIN	Returns the doma incorrect result. Fo hostname of 192.
SIP.REQ.URL.HOSTNAME.SERVER	Returns the serve .
SIP.REQ.URL.USERNAME	Returns the usern transport=udp, thi
SIP.REQ.VERSION	Returns the SIP v 5060;transpor
SIP.REQ.VERSION.MAJOR	Returns the major this expression re
SIP.REQ.VERSION.MINOR	Returns the minor , this expression r
SIP.REQ.CONTENT_LENGTH	Returns the conte operations that ar Length: 277, th
SIP.REQ.TO	Returns the conte tag=00127f54e tag=00127f54e
SIP.REQ.TO.ADDRESS	Returns the SIP U example, for a SIF 53cc0185, this e;
SIP.REQ.TO.DISPLAY_NAME	Returns the displa 16@sip_exempl
SIP.REQ.TO.TAG	Returns the â€œet To: "16" <sip 00127f54ec85a
SIP.REQ.FROM	Returns the conte com>;tag=0012
SIP.REQ.FROM.ADDRESS	Returns the SIP U example, for a SIF 53cc0185, this e;
SIP.REQ.FROM.DISPLAY_NAME	Returns the displa 12@sip_exempl

SIP.REQ.FROM.TAG	Returns the "From" tag. From: "12" <00127f54ec85a
SIP.REQ.VIA	Returns the component for the two Via headers. branch=z9hG4bK
SIP.REQ.VIA.SENTBY_ADDRESS	Returns the address. branch=z9hG4bK
SIP.REQ.VIA.SENTBY_PORT	Returns the port. branch=z9hG4bK
SIP.REQ.VIA.RPORT	Returns the value. 5060;branch=z
SIP.REQ.VIA.BRANCH	Returns the value. 10.102.84.180 returns z9hG4bK
SIP.REQ.VIA.RECEIVED	Returns the value. 10.102.84.180 returns 10.102.8
SIP.REQ.CALLID	Returns the content. are available for S 0e46f5b9-5ec1 102.84.180.
SIP.REQ.CSEQ	Returns the CSeq expression. expression returns
SIP.REQ.HEADER( "<header_name>" )	Returns the specific header. return the SIP Fro
SIP.REQ.HEADER( "<header_name>" ).INSTANCE( <line_number> )	Returns the specific instance. want a specific instance number as the <1> ( "Via" ).INSTANCE returns the last instance.  For example, if us 10.102.84.180
SIP.REQ.HEADER( "<header_name>" ).VALUE( <line_number> )	Returns the content expression. expression. For example, VALUE( 1 ) returns
SIP.REQ.HEADER( "<header_name>" ).COUNT	Returns the number above, SIP.REQ.
SIP.REQ.HEADER( "<header_name>" ).EXISTS	Returns a boolean. the SIP header exists. Caller-ID ( )



<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).LIST</code>	<p>Returns the comma-separated list of header values for the specified header. Example: <code>SIP.REQ.HEADER( "Content-Length" ).LIST</code> returns <code>100</code>.</p> <p>You can append <code>UPDATE</code> to the end of the command to update the header value. Example: <code>SIP.REQ.HEADER( "Content-Length" ).LIST UPDATE 200</code> updates the Content-Length header to 200.</p> <p>Note: If the specified header name is not found, the command returns an empty list.</p>
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).TYPECAST_SIP_HEADER_T( "&lt;in_header_name&gt;" )</code>	<p>Typecasts the specified header value to the specified type. Example: <code>SIP.REQ.HEADER( "Content-Length" ).TYPECAST_SIP_HEADER_T( "int" )</code> returns <code>100</code>.</p> <p>For example, the <code>Content-Length</code> header is a string. If you typecast it to an integer, the result is <code>100</code>.</p>
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).CONTAINS( "&lt;string&gt;" )</code>	Returns boolean true if the specified string is contained in the header value. Example: <code>SIP.REQ.HEADER( "Content-Length" ).CONTAINS( "100" )</code> returns <code>true</code> .
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).EQUALS_ANY( &lt;patset&gt; )</code>	Returns boolean true if the header value equals any of the patterns in the specified pattern set. Operates on all the header values.
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).CONTAINS_ANY( &lt;patset&gt; )</code>	Returns Boolean true if the header value contains any of the patterns in the specified pattern set. Operates on all the header values.
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).CONTAINS_INDEX( &lt;patset&gt; )</code>	Returns the index of the first occurrence of the specified header value in the specified pattern set. Example: <code>SIP.REQ.HEADER( "Content-Length" ).CONTAINS_INDEX( "100" )</code> returns <code>1</code> .
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).EQUALS_INDEX( &lt;patset&gt; )</code>	Returns the index of the first occurrence of the specified header value in the specified pattern set. Operates on all the header values.
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).SUBSTR( "&lt;string&gt;" )</code>	If the specified string is found in the header value, returns the substring starting from the specified index. Example: <code>SIP.REQ.HEADER( "Content-Length" ).SUBSTR( "100" )</code> returns <code>100</code> .
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).AFTER_STR( "&lt;string&gt;" )</code>	If the specified string is found in the header value, returns the substring after the specified string. Example: <code>SIP.REQ.HEADER( "Content-Length" ).AFTER_STR( "Content-Length=" )</code> returns <code>100</code> .
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).REGEX_MATCH( &lt;regex&gt; )</code>	<p>Returns boolean true if the header value matches the specified regular expression. Example: <code>SIP.REQ.HEADER( "Content-Length" ).REGEX_MATCH( "[0-9]+" )</code> returns <code>true</code>.</p> <p>The regular expression is specified by the <code>&lt;regex&gt;</code> parameter. The regular expression is a string that defines a search pattern. The regular expression is used to search for a match in the header value.</p> <p>The regular expression is a string that defines a search pattern. The regular expression is used to search for a match in the header value.</p>
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).REGEX_SELECT( &lt;regex&gt; )</code>	If the specified regular expression is found in the header value, returns the substring selected by the regular expression. Example: <code>SIP.REQ.HEADER( "Content-Length" ).REGEX_SELECT( "[0-9]+" )</code> returns <code>100</code> .

<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).AFTER_REGEX( &lt;regex&gt; )</code>	If the specified req after that text. For branch=z9hG4b: AFTER_REGEX( ":
<code>SIP.REQ.HEADER( "&lt;header_name&gt;" ).BEFORE_REGEX( &lt;regex&gt; )</code>	If the specified req before that text. F branch=z9hG4b: BEFORE_REGEX(
<code>SIP.REQ.FULL_HEADER</code>	Returns the entire
<code>SIP.REQ.IS_VALID</code>	Returns boolean t
<code>SIP.REQ.BODY( &lt;length&gt; )</code>	Returns the requ expression returns:
<code>SIP.REQ.LB_VSERVER</code>	Returns the name
<code>SIP.REQ.CS_VSERVER</code>	Returns the name

Table 2. SIP Response Expressions

Expression	Description
<code>SIP.RES.STATUS</code>	Returns the SIP response status code. For example, if the first line of the response is <code>SIP/2.0 100 Trying</code> , this expression returns <code>100</code> .
<code>SIP.RES.STATUS_MSG</code>	Returns the SIP response status message. For example, if the first line of the response is <code>SIP/2.0 100 Trying</code> , this expression returns <code>Trying</code> .
<code>SIP.RES.IS_REDIRECT</code>	Returns boolean <code>true</code> if the response code is a redirect.
<code>SIP.RES.METHOD</code>	Returns the response method extracted from the request method string in the CSeq header.

## Operations for HTTP, HTML, and XML Encoding and "Safe" Characters

The following operations work with the encoding of HTML data in a request or response and XML data in a POST body.

Table 1. Operations That Evaluate HTML and XML Encoding

HTML or XML Operation	Description
<code>&lt;text&gt;.HTML_XML_SAFE</code>	<p>Transforms special characters. Examples:</p> <ul style="list-style-type: none"> <li>• A left-pointing angle bracket: "&lt;"</li> <li>• A right-pointing angle bracket: "&gt;"</li> <li>• An ampersand: "&amp;"</li> </ul> <p>This operation safeguards the maximum length of the transformed text.</p> <p>After applying the transformation, the following expressions are applied:</p> <pre>http.req.url.queryString = "myQueryString"</pre>
<code>&lt;text&gt;.HTTP_HEADER_SAFE</code>	<p>Converts all new line characters in the input to be used safely in an HTTP header.</p> <p>This operation safeguards the maximum length of the transformed text.</p> <p>The maximum length of the transformed text is only operation.</p>
<code>&lt;text&gt;.HTTP_URL_SAFE</code>	<p>Converts unsafe URL based representation characters. The ampersand (&amp;) is represented as %26 in the transformed text.</p> <p>Following are URL safe characters:</p> <ul style="list-style-type: none"> <li>• Alpha-numeric characters: "a-z", "A-Z", "0-9"</li> <li>• Asterisk: "*"</li> <li>• Ampersand: "&amp;"</li> <li>• At-sign: "@"</li> <li>• Colon: ":"</li> <li>• Comma: ","</li> <li>• Dollar: "\$"</li> <li>• Dot: "."</li> <li>• Equals: "="</li> <li>• Exclamation mark: "!"</li> <li>• Hyphen: "-"</li> <li>• Open and close square brackets: "[", "]"</li> <li>• Percent: "%"</li> <li>• Plus: "+"</li> <li>• Semicolon: ";"</li> <li>• Single quote: "'"</li> <li>• Slash: "/"</li> <li>• Question mark: "?"</li> <li>• Tilde: "~"</li> <li>• Underscore: "_"</li> </ul>
<code>&lt;text&gt;.MARK_SAFE</code>	Marks the text as safe

<pre>&lt;text&gt;.SET_TEXT_MODE(URLENCODED NOURLENCODED)</pre>	<p>Transforms all %HH e characters (not bytes) ASCII encoding. How can represent a chara</p> <p>In the following exam characters in a target.</p> <p>http.req.url.hos</p> <p>In the following exam target:</p> <p>http.req.url.hos (3)</p>
<pre>&lt;text&gt;.SET_TEXT_MODE(PLUS_AS_SPACE NO_PLUS_AS_SPACE)</pre>	<p>Specifies how to treat replaces a plus charac hello+worldâ€• becorr option leaves plus che</p>
<pre>&lt;text&gt;.SET_TEXT_MODE(BACKSLASH_ENCODED NO_BACKSLASH_ENCODED)</pre>	<p>Specifies whether or r represented by &lt;text&gt;</p> <p>If BACKSLASH_ENCODE performs the following</p> <ul style="list-style-type: none"> <li>• All occurrences â€œYâ€• (wher represents the A values for this ty encoded text "hl //", where the co 72â€•.</li> <li>• All occurrences â€œYâ€• (HH r denotes the AS( "http\x3a//" will t ASCII equivalen</li> <li>• All occurrences character seque distinct hexadec equivalents of V text "http%u3a2 //", where â€œ3 the colon (:) and equivalents resp</li> <li>• All occurrences corresponding A</li> </ul> <p>If NO_BACKSLASH_EN performed on the text</p>
<pre>&lt;text&gt;.SET_TEXT_MODE (BAD_ENCODE_RAISE_UNDEF NO_BAD_ENCODE_RAISE_UNDEF)</pre>	<p>Performs the associat BACKSLASH_ENCODE: specified encoding mc &lt;text&gt;.</p> <p>If NO_BAD_ENCODE_R action will not be perf object represented by</p>

## Expressions for TCP, UDP, and VLAN Data

TCP and UDP data take the form of a string or a number. For expression prefixes that return string values for TCP and UDP data, you can apply any text-based operations. For more information, see ["Default Syntax Expressions: Evaluating Text."](#)

For expression prefixes that return numeric value, such as a source port, you can apply an arithmetic operation. For more information, see ["Basic Operations on Expression Prefixes"](#) and ["Compound Operations for Numbers."](#)

The following table describes prefixes that extract TCP and UDP data.

Table 1. Prefixes That Extract TCP and UDP Data

GET Operation	Description
<code>CLIENT.TCP.PAYLOAD(&lt;integer&gt;)</code>	Returns TCP payload data as a string, starting with the first character in the payload and continuing for the number of characters in the <integer> argument.  You can apply any text-based operation to this prefix.
<code>CLIENT.TCP.SRCPORT</code>	Returns the ID of the current packet's source port as a number.
<code>CLIENT.TCP.DSTPORT</code>	Returns the ID of the current packet's destination port as a number.
<code>CLIENT.TCP.OPTIONS</code>	Returns the TCP options set by the client. Examples of TCP options are Maximum Segment Size (MSS), Window Scale, Selective Acknowledgements (SACK), and Time Stamp Option. The <code>COUNT</code> , <code>TYPE(&lt;type&gt;)</code> , and <code>TYPE_NAME(&lt;m&gt;)</code> operators can be used with this prefix. For the TCP options set by the server, see the <code>SERVER.TCP.OPTIONS</code> prefix.
<code>CLIENT.TCP.OPTIONS.COUNT</code>	Returns the number of TCP options that the client has set.
<code>CLIENT.TCP.OPTIONS.TYPE(&lt;type&gt;)</code>	Returns the value of the TCP option whose type (or <i>option kind</i> ) is specified as the argument. The value is returned as a string of bytes in big endian format (or <i>network byte order</i> ).  <b>Parameters:</b>  <code>type</code> - Type value
<code>CLIENT.TCP.OPTIONS.TYPE_NAME(&lt;m&gt;)</code>	Returns the value of the TCP option whose enumeration constant is specified as the argument. The enumeration constants that you can pass as the argument are REPEATER, TIMESTAMP, SACK_PERMITTED, WINDOW, and MAXSEG. To specify the TCP option kind instead of these enumeration constants, use <code>CLIENT.TCP.OPTIONS.TYPE(&lt;type&gt;)</code> . For other TCP options, you must use <code>CLIENT.TCP.OPTIONS.TYPE(&lt;type&gt;)</code> .  <b>Parameters:</b>  <code>m</code> - TCP option enumeration constant
<code>CLIENT.TCP.REPEATER_OPTION.EXISTS</code>	Returns a Boolean TRUE if Repeater TCP options exist.
<code>CLIENT.TCP.REPEATER_OPTION.IP</code>	

	Returns the branch repeater's IPv4 address from the Repeater TCP options.
<code>CLIENT.TCP.REPEATER_OPTION.MAC</code>	Returns the branch repeater's MAC address from the Repeater TCP options.
<code>CLIENT.UDP.DNS.DOMAIN</code>	Returns the DNS domain name.
<code>CLIENT.UDP.DNS.DOMAIN.EQ</code> ( "<hostname>" )	<p>Returns a Boolean TRUE if the domain name matches the &lt;hostname&gt; argument. The comparison is case insensitive.</p> <p>Following is an example:</p> <pre>client.udp.dns.domain.eq( "www.mycompany.com" )</pre>
<code>CLIENT.UDP.DNS.IS_AAAAREC</code>	Returns a Boolean TRUE if the record type is AAAA. These types of records indicate an IPv6 address in forward lookups.
<code>CLIENT.UDP.DNS.IS_ANYREC</code>	Returns a Boolean TRUE if it is of any record type.
<code>CLIENT.UDP.DNS.IS_AREC</code>	Returns a Boolean TRUE if the record is type A. Type A records provide the host address.
<code>CLIENT.UDP.DNS.IS_CNAMEREC</code>	Returns a Boolean TRUE if the record is of type CNAME. In systems that use multiple names to identify a resource, there is one canonical name and a number of aliases. The CNAME provides the canonical name.
<code>CLIENT.UDP.DNS.IS_MXREC</code>	Returns a Boolean TRUE if the record is of type MX (mail exchanger). This DNS record describes a priority and a host name. The MX records for the same domain name specify the email servers in the domain and the priority for each server.
<code>CLIENT.UDP.DNS.IS_NSREC</code>	Returns a Boolean TRUE if the record is of type NS. This is a name server record that includes a host name with an associated A record. This enables locating the domain name that is associated with the NS record.
<code>CLIENT.UDP.DNS.IS_PTRREC</code>	Returns a Boolean TRUE if the record is of type PTR. This is a domain name pointer and is often used to associate a domain name with an IPv4 address.
<code>CLIENT.UDP.DNS.IS_SOAREC</code>	Returns a Boolean TRUE if the record is of type SOA. This is a start of authority record.
<code>CLIENT.UDP.DNS.IS_SRVREC</code>	Returns a Boolean TRUE if the record is of type SRV. This is a more general version of the MX record.
<code>CLIENT.UDP.DSTPORT</code>	Returns the numeric ID of the current packet's UDP destination port.
<code>CLIENT.UDP.SRCPORT</code>	Returns the numeric ID of the current packet's UDP source port.

<code>CLIENT.UDP.RADIUS</code>	Returns RADIUS data for the current packet.
<code>CLIENT.UDP.RADIUS.ATTR_TYPE(&lt;type&gt;)</code>	Returns the value for the attribute type specified as the argument.
<code>CLIENT.UDP.RADIUS.USERNAME</code>	Returns the RADIUS user name.
<code>CLIENT.TCP.MSS</code>	Returns the maximum segment size (MSS) for the current connection as a number.
<code>CLIENT.VLAN.ID</code>	Returns the numeric ID of the VLAN through which the current packet entered the NetScaler.
<code>SERVER.TCP.DSTPORT</code>	Returns the numeric ID of the current packet's destination port.
<code>SERVER.TCP.SRCPORT</code>	Returns the numeric ID of the current packet's source port.
<code>SERVER.TCP.OPTIONS</code>	Returns the TCP options set by the server. Examples of TCP options are Maximum Segment Size (MSS), Window Scale, Selective Acknowledgements (SACK), and Time Stamp Option. The <code>COUNT</code> , <code>TYPE(&lt;type&gt;)</code> , and <code>TYPE_NAME(&lt;m&gt;)</code> operators can be used with this prefix. For the TCP options set by the client, see the <code>CLIENT.TCP.OPTIONS</code> prefix.
<code>SERVER.TCP.OPTIONS.COUNT</code>	Returns the number of TCP options that the server has set.
<code>SERVER.TCP.OPTIONS.TYPE(&lt;type&gt;)</code>	<p>Returns the value of the TCP option whose type (or <i>option kind</i>) is specified as the argument. The value is returned as a string of bytes in big endian format (or <i>network byte order</i>).</p> <p><b>Parameters:</b></p> <p><code>type</code> - Type value</p>
<code>SERVER.TCP.OPTIONS.TYPE_NAME(&lt;m&gt;)</code>	<p>Returns the value of the TCP option whose enumeration constant is specified as the argument. The enumeration constants that you can pass as the argument are REPEATER, TIMESTAMP, SACK_PERMITTED, WINDOW, and MAXSEG. To specify the TCP option kind instead of these enumeration constants, use <code>CLIENT.TCP.OPTIONS.TYPE(&lt;type&gt;)</code>. For other TCP options, you must use <code>CLIENT.TCP.OPTIONS.TYPE(&lt;type&gt;)</code>.</p> <p><b>Parameters:</b></p> <p><code>m</code> - TCP option enumeration constant</p>
<code>SERVER.VLAN</code>	Operates on the VLAN through which the current packet entered the NetScaler.
<code>SERVER.VLAN.ID</code>	Returns the numeric ID of the VLAN through which the current packet entered the NetScaler.

## Expressions for Evaluating a DNS Message and Identifying Its Carrier Protocol

You can evaluate DNS requests and responses by using expressions that begin with `DNS.REQ` and `DNS.RES`, respectively. You can also identify the transport layer protocol that is being used to send the DNS messages.

The following functions return the contents of a DNS query.

Table 1. Functions that return the contents of a DNS query

Function	Description
<code>DNS.REQ.QUESTION.DOMAIN</code>	Return the domain name (the value of the <code>QNAME</code> field) in the question section of the DNS query. The domain name is returned as a text string, which can be passed to <code>EQ()</code> , <code>NE()</code> , and any other functions that work with text.
<code>DNS.REQ.QUESTION.TYPE</code>	<p>Return the query type (the value of the <code>QTYPE</code> field) in the DNS query. The field indicates the type of resource record (for example, A, NS, or CNAME) for which the name server is being queried. The returned value can be compared to one of the following values by using the <code>EQ()</code> and <code>NE()</code> functions:</p> <ul style="list-style-type: none"><li>○ A</li><li>○ AAAA</li><li>○ NS</li><li>○ SRV</li><li>○ PTR</li><li>○ CNAME</li><li>○ SOA</li><li>○ MX</li><li>○ ANY</li></ul> <p>Note: You can use only the <code>EQ()</code> and <code>NE()</code> functions with the <code>TYPE</code> function.</p> <p><b>Example:</b></p> <p><code>DNS.REQ.QUESTION.TYPE.EQ(MX)</code></p>

The following functions return the contents of a DNS response.

Table 2. Functions that return the contents of a DNS response

Function	Description
<code>DNS.RES.HEADER.RCODE</code>	<p>Return the response code (the value of the <code>RCODE</code> field) in the header section of the DNS response. You can use only the <code>EQ()</code> and <code>NE()</code> functions with the <code>RCODE</code> function. Following are the possible values:</p> <ul style="list-style-type: none"><li>○ NOERROR</li><li>○ FORMERR</li><li>○ SERVFAIL</li><li>○ NXDOMAIN</li><li>○ NOTIMP</li><li>○ REFUSED</li></ul>
<code>DNS.RES.QUESTION.DOMAIN</code>	Return the domain name (the value of the <code>QNAME</code> field) in the question section of the DNS response. The domain name is returned as a text string, which can be passed to <code>EQ()</code> , <code>NE()</code> , and any other functions that work with text.
<code>DNS.RES.QUESTION.TYPE</code>	<p>Return the query type (the value of the <code>QTYPE</code> field) in the question section of the DNS response. The field indicates the type of resource record (for example, A, NS, or CNAME) that is contained in the response. The returned value can be compared to one of the following values by using the <code>EQ()</code> and <code>NE()</code> functions:</p> <ul style="list-style-type: none"><li>○ A</li><li>○ AAAA</li></ul>



- o NS
- o SRV
- o PTR
- o CNAME
- o SOA
- o MX
- o ANY

You can use only the EQ ( ) and NE ( ) functions with the TYPE function.

**Example:**

DNS.RES.QUESTION.TYPE.EQ(SOA)

The following functions return the transport layer protocol name.

Table 3. Functions that return the transport layer protocol name

Function	Description
DNS.REQ. TRANSPORT	<p>Return the name of the transport layer protocol that was used to send the DNS query. Possible values returned are TCP and UDP. You can use only the EQ ( ) and NE ( ) functions with the TRANSPORT function.</p> <p><b>Example:</b></p> <p>DNS.REQ.TRANSPORT.EQ(TCP)</p>
DNS.RES. TRANSPORT	<p>Return the name of the transport layer protocol that was used for the DNS response. Possible values returned are TCP and UDP. You can use only the EQ() and NE() functions with the TRANSPORT function.</p> <p><b>Example:</b></p> <p>DNS.RES.TRANSPORT.EQ(TCP)</p>

## XPath and HTML, XML, or JSON Expressions

The default syntax expression engine supports expressions for evaluating and retrieving data from HTML, XML, and JavaScript Object Notation (JSON) files. This enables you to find specific nodes in an HTML, XML, or JSON document, determine if a node exists in the file, locate nodes in XML contexts (for example, nodes that have specific parents or a specific attribute with a given value), and return the contents of such nodes. Additionally, you can use XPath expressions in rewrite expressions.

The default syntax expression implementation for XPath comprises a default syntax expression prefix (such as `HTTP.REQ.BODY`) that designates HTML or XML text, and the XPATH operator that takes the XPath expression as its argument.

HTML files are a largely free-form collection of tags and text elements. You can use the XPATH\_HTML operator, which takes an XPath expression as its argument, to process HTML files. JSON files are either a collection of name/value pairs or an ordered list of values. You can use the XPATH\_JSON operator, which takes an XPath expression as its argument, to process JSON files.

Table 1. XPath and JSON Expression Prefixes That Return Text

XPath Prefix	Description
<code>&lt;text&gt;.XPATH(xpathex)</code>	<p>Operate on an XML file and return a Boolean value.</p> <p>For example, the following expression returns a Boolean TRUE if <code>&lt;creator&gt;</code> exists under the node <code>&lt;Book&gt;</code> within the first 1000 bytes of the XML file:</p> <pre>HTTP.REQ.BODY(1000).XPATH(xpathex)</pre> <p>Parameters:</p> <p>xpathex - XPath Boolean expression</p>
<code>&lt;text&gt;.XPATH(xpathex)</code>	<p>Operate on an XML file and return a value of data type <code>double</code>.</p> <p>For example, the following expression converts the string <code>36.5</code> (a value of data type <code>string</code>) to a value of data type <code>double</code> if the string is in the first 1000 bytes of the XML file:</p> <pre>HTTP.REQ.BODY(1000).XPATH(xpathex)</pre> <p>Parameters:</p> <p>xpathex - XPath numeric expression</p>
<code>&lt;text&gt;.XPATH(xpathex)</code>	<p>Operate on an XML file and return a node-set or a string. Node-sets are converted to corresponding strings by using the standard XPath <code>string()</code> routine.</p> <p>For example, the following expression selects all the nodes that are <code>&lt;Book/creator&gt;</code> (a node-set) in the first 1000 bytes of the body of the request:</p> <pre>HTTP.REQ.BODY(1000).XPATH(xpathex)</pre> <p>Parameters:</p> <p>xpathex - XPath expression</p>
<code>&lt;text&gt;.XPATH_HTML(xpathex)</code>	<p>Operate on an HTML file and return a text value.</p> <p>For example, the following expression operates on an HTML file and returns the text enclosed in <code>&lt;title&gt;&lt;/title&gt;</code> tags if the title HTML element is found in the first 1000 bytes:</p> <pre>HTTP.REQ.BODY(1000).XPATH_HTML(xpathex)</pre>

	<p>Parameters:</p> <p>xpathex - XPath text expression</p>
<p>&lt;text&gt;.XPATH_HTML_WITH_MARKUP(xpathex)</p>	<p>Operate on an HTML file and return a string that contains the entire portion of the document, including markup such as including the element tags.</p> <p>The following expression operates on the HTML file and selects the &lt;title&gt; tag, including markup.</p> <pre>HTTP.REQ.BODY(1000).XPATH_HTML_WITH_MARKUP( xp% /html/head/title%)</pre> <p>The portion of the HTML body that is selected by the expression is further processing.</p> <p>Parameters:</p> <p>xpathex - XPath expression</p>
<p>&lt;text&gt;.XPATH_JSON(xpathex)</p>	<p>Operate on a JSON file and return a Boolean value.</p> <p>For example, consider the following JSON file:</p> <pre>{ "Book":{ "creator":{ "person":{ "name":'&lt;name&gt;' } }, "title":'&lt;title&gt;' }</pre> <p>The following expression operates on the JSON file and returns a Boolean value if the JSON file contains a node named "creator" whose parent is "Book" in the first 1000 bytes:</p> <pre>HTTP.REQ.BODY(1000).XPATH_JSON( xp%boolean( /Book/creator ) )</pre> <p>Parameters:</p> <p>xpathex - XPath Boolean expression</p>
<p>&lt;text&gt;.XPATH_JSON(xpathex)</p>	<p>Operate on a JSON file and return a value of data type "double".</p> <p>For example, consider the following JSON file:</p> <pre>{ "Book":{ "creator":{ "person":{ "name":'&lt;name&gt;' } }, "title":'&lt;title&gt;' }</pre> <p>The following expression operates on the JSON file and converts the string value of the "price" node to a value of data type "double" if the string is present in the first 1000 bytes of the JSON file.</p> <pre>HTTP.REQ.BODY(1000).XPATH_JSON( xp%number( /Book/price ) )</pre> <p>Parameters:</p> <p>xpathex - XPath numeric expression</p>
<p>&lt;text&gt;.XPATH_JSON(xpathex)</p>	<p>Operate on a JSON file and return a node-set or a string. Node-sets are converted to corresponding strings by using the standard XPath string() routine.</p> <p>For example, consider the following JSON file:</p> <pre>{ "Book":{ "creator":{ "person":{ "name":'&lt;name&gt;' } }, "title":'&lt;title&gt;' }</pre> <p>The following expression selects all the nodes that are enclosed in double quotes (node-set) in the first 1000 bytes of the body of the JSON file and returns the corresponding string value, which is "&lt;name&gt;&lt;title&gt;":</p> <pre>HTTP.REQ.BODY(1000).XPATH_JSON( xp%"/Book/creator/person/name" )</pre> <p>Parameters:</p>

	xpathex - XPath expression
<code>&lt;text&gt;.XPATH_JSON_WITH_MARKUP (xpathex)</code>	<p>Operate on an XML file and return a string that contains the entire document for the result node, including markup such as including element tags.</p> <p>For example, consider the following JSON file:</p> <pre>{ "Book": { "creator": { "person": { "name": '&lt;name&gt;' } }, "title": '&lt;title&gt;' }</pre> <p>The following expression operates on the JSON file and selects a are enclosed by "Book/creator" in the first 1000 bytes of the t</p> <pre>HTTP.REQ.BODY(1000).XPATH_JSON_WITH_MARKUP (xp% /Book/creator%)</pre> <p>The portion of the JSON body that is selected by the expression is: further processing.</p> <p>Parameters:</p> <p>xpathex - XPath expression</p>
<code>&lt;text&gt;.XPATH_WITH_MARKUP (xpathex)</code>	<p>Operate on an XML file and return a string that contains the entire document for the result node, including markup such as including element tags.</p> <p>For example, the following expression operates on an XML file an nodes enclosed by "Book/creator" in the first 1000 bytes of th</p> <pre>HTTP.REQ.BODY(1000).XPATH_WITH_MARKUP (xp% /Book/c:</pre> <p>The portion of the JSON body that is selected by the expression is: further processing.</p> <p>Parameters:</p> <p>xpathex - XPath expression</p>

## Encrypting and Decrypting XML Payloads

You can use the `XML_ENCRYPT()` and `XML_DECRYPT()` functions in default syntax expressions to encrypt and decrypt, respectively, XML data. These functions conform to the W3C XML Encryption standard defined at "<http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/>." `XML_ENCRYPT()` and `XML_DECRYPT()` support a subset of the XML Encryption specification. In the subset, data encryption uses a bulk cipher method (RC4, DES3, AES128, AES192, or AES256), and an RSA public key is used to encrypt the bulk cipher key.

Note: If you want to encrypt and decrypt text in a payload, you must use the `ENCRYPT` and `DECRYPT` functions. For more information about these functions, see "[Encrypting and Decrypting Text](#)."

The `XML_ENCRYPT()` and `XML_DECRYPT()` functions are not dependent on the encryption/decryption service that is used by the `ENCRYPT` and `DECRYPT` commands for text. The cipher method is specified explicitly as an argument to the `XML_ENCRYPT()` function. The `XML_DECRYPT()` function obtains the information about the specified cipher method from the `<xenc:EncryptedData>` element. Following are synopses of the XML encryption and decryption functions:

- **`XML_ENCRYPT(<certKeyName>, <method> [, <flags>])`**. Returns an `<xenc:EncryptedData>` element that contains the encrypted input text and the encryption key, which is itself encrypted by using RSA.
- **`XML_DECRYPT(<certKeyName>)`**. Returns the decrypted text from the input `<xenc:EncryptedData>` element, which includes the cipher method and the RSA-encrypted key.

Note: The `<xenc:EncryptedData>` element is defined in the W3C XML Encryption specification.

Following are descriptions of the arguments:

**certKeyName**

Selects an X.509 certificate with an RSA public key for `XML_ENCRYPT()` or an RSA private key for `XML_DECRYPT()`. The certificate key must have been previously created by an `add ssl certKey` command.

**method**

Specifies which cipher method to use for encrypting the XML data. Possible values: RC4, DES3, AES128, AES192, AES256.

**flags**

A bitmask specifying the following optional key information (`<ds:KeyInfo>`) to be included in the `<xenc:EncryptedData>` element that is generated by `XML_ENCRYPT()`:

- **1** - Include a `KeyName` element with the `certKeyName`. The element is `<ds:KeyName>`.
- **2** - Include a `KeyValue` element with the RSA public key from the certificate. The element is `<ds:KeyValue>`.
- **4** - Include an `X509IssuerSerial` element with the certificate serial number and issuer DN. The element is `<ds:X509IssuerSerial>`.
- **8** - Include an `X509SubjectName` element with the certificate subject DN. The element is `<ds:X509SubjectName>`.
- **16** - Include an `X509Certificate` element with the entire certificate. The element is `<ds:X509Certificate>`.

## Using the `XML_ENCRYPT()` and `XML_DECRYPT()` Functions in Expressions

The XML encryption feature uses SSL certificate-key pairs to provide X.509 certificates (with RSA public keys) for key encryption and RSA private keys for key decryption. Therefore, before you use the `XML_ENCRYPT()` function in an expression, you must create an SSL certificate-key pair. The following command creates an SSL certificate-key pair, `my-certkey`, with the X.509 certificate, `my-cert.pem`, and the private key file, `my-key.pem`.

```
add ssl certKey my-certkey -cert my-cert.pem -key my-key.pem -passcrypt kxPeMRyNitY=
```

The following CLI commands create rewrite actions and policies for encrypting and decrypting XML content.

```
add rewrite action my-xml-encrypt-action replace "HTTP.RES.BODY(10000).XPATH_WITH_MARKUP
(xp%/)" "HTTP.RES.BODY(10000).XPATH_WITH_MARKUP(xp%/).XML_ENCRYPT(\"my-certkey\",
AES256, 31)" -bypassSafetyCheck YES
```

```

add rewrite action my-xml-decrypt-action replace "HTTP.REQ.BODY(10000).XPATH_WITH_MARKUP
(xp%/xenc:EncryptedData%)" "HTTP.REQ.BODY(10000).XPATH_WITH_MARKUP(xp%/xenc:
EncryptedData%).XML_DECRYPT(\"my-certkey\")" -bypassSafetyCheck YES

add rewrite policy my-xml-encrypt-policy "HTTP.REQ.URL.CONTAINS(\"xml-encrypt\")" my-xml-
encrypt-action

add rewrite policy my-xml-decrypt-policy "HTTP.REQ.BODY(10000).XPATH(xp%boolean(//xenc:
EncryptedData%))" my-xml-decrypt-action

bind rewrite global my-xml-encrypt-policy 30

bind rewrite global my-xml-decrypt-policy 30

```

In the above example, the rewrite action `my-xml-encrypt-action` encrypts the entire XML document (`XPATH_WITH_MARKUP(xp%/%)`) in the request by using the AES-256 bulk encryption method and the RSA public key from `my-certkey` to encrypt the bulk encryption key. The action replaces the document with an `<xenc:EncryptedData>` element containing the encrypted data and an encrypted key. The flags represented by 31 include all of the optional `<ds:KeyInfo>` elements.

The action `my-xml-decrypt-action` decrypts the first `<xenc:EncryptedData>` element in the response (`XPATH_WITH_MARKUP(xp%/xenc:EncryptedData%)`). This requires the prior addition of the `xenc` XML namespace by use of the following CLI command:

```
add ns xmlnspace xenc http://www.w3.org/2001/04/xmlenc#
```

The `my-xml-decrypt-action` action uses the RSA private key in `my-certkey` to decrypt the encrypted key and then uses the bulk encryption method specified in the element to decrypt the encrypted contents. Finally, the action replaces the encrypted data element with the decrypted content.

The rewrite policy `my-xml-encrypt-policy` applies `my-xml-encrypt-action` to requests for URLs containing `xml-encrypt`. The action encrypts the entire response from a service configured on the NetScaler appliance.

The rewrite policy `my-xml-decrypt-policy` applies `my-xml-decrypt-action` to requests that contain an `<xenc:EncryptedData>` element (`(XPATH(xp%/xenc:EncryptedData%))` returns a non-empty string). The action decrypts the encrypted data in requests that are bound for a service configured on the NetScaler appliance.

## Default Syntax Expressions: Parsing SSL Certificates

You can use default syntax expressions to evaluate X.509 Secure Sockets Layer (SSL) client certificates. A client certificate is an electronic document that can be used to authenticate a user's identity. A client certificate contains (at a minimum) version information, a serial number, a signature algorithm ID, an issuer name, a validity period, a subject (user) name, a public key, and signatures.

You can examine both SSL connections and data in client certificates. For example, you may want to send SSL requests that use low-strength ciphers to a particular load balancing virtual server farm. The following command is an example of a Content Switching policy that parses the cipher strength in a request and matches cipher strengths that are less than or equal to 40:

```
add cs policy p1 -rule "client.ssl.cipher_bits.le(40)"
```

As another example, you can configure a policy that determines whether a request contains a client certificate:

```
add cs policy p2 -rule "client.ssl.client_cert EXISTS"
```

Or, you might want to configure a policy that examines particular information in a client certificate. For example, the following policy verifies that the certificate has one or more days before expiration:

```
add cs policy p2 -rule "client.ssl.client_cert exists && client.ssl.client_cert.  
days_to_expire.ge(1)"
```

Note: For information on parsing dates and times in a certificate, see ["Format of Dates and Times in an Expression"](#) and ["Expressions for SSL Certificate Dates."](#)

This document includes the following details:

- [Prefixes for Text-Based SSL and Certificate Data](#)
- [Prefixes for Numeric Data in SSL Certificates](#)
- [Expressions for SSL Certificates](#)

## Prefixes for Text-Based SSL and Certificate Data

The following table describes expression prefixes that identify text-based items in SSL transactions and client certificates.

Table 1. Prefixes That Return Text or Boolean Values for SSL and Client Certificate Data

Prefix	Description
CLIENT.SSL.CLIENT_CERT	Returns the SSL client certificate in the current SSL transaction.
CLIENT.SSL.CLIENT_CERT. TO_PEM	Returns the SSL client certificate in binary format.
CLIENT.SSL.CIPHER_EXPORTABLE	Returns a Boolean TRUE if the SSL cryptographic SSL cryptographic cipher is exportable.
CLIENT.SSL.CIPHER_NAME	Returns the name of the SSL Cipher if invoked from an SSL connection, and a NULL string if invoked from a non-SSL connection.
CLIENT.SSL.IS_SSL	Returns a Boolean TRUE if the current connection is SSL-based.

## Prefixes for Numeric Data in SSL Certificates

Updated: 2013-09-02

The following table describes prefixes that evaluate numeric data other than dates in SSL certificates. These prefixes can be used with the operations that are described in ["Basic Operations on Expression Prefixes"](#) and ["Compound Operations for Numbers."](#)

Table 2. Prefixes That Evaluate Numeric Data Other Than Dates in SSL Certificates

|--|--|

Prefix	Description
<code>CLIENT.SSL.CLIENT_CERT.DAYS_TO_EXPIRE</code>	Returns the number of days that the certificate is valid, or returns -1 for expired certificates.
<code>CLIENT.SSL.CLIENT_CERT.PK_SIZE</code>	Returns the size of the public key used in the certificate.
<code>CLIENT.SSL.CLIENT_CERT.VERSION</code>	Returns the version number of the certificate. If the connection is not SSL-based, returns zero (0).
<code>CLIENT.SSL.CIPHER_BITS</code>	Returns the number of bits in the cryptographic key. Returns 0 if the connection is not SSL-based.
<code>CLIENT.SSL.VERSION</code>	Returns a number that represents the SSL protocol version, as follows: <ul style="list-style-type: none"> <li>0. The transaction is not SSL-based.</li> <li>0x002. The transaction is SSLv2.</li> <li>0x300. The transaction is SSLv3.</li> <li>0x301. The transaction is TLSv1.</li> </ul>

Note: For expressions related to expiration dates in a certificate, see ["Expressions for SSL Certificate Dates."](#)

## Expressions for SSL Certificates

Updated: 2013-09-02

You can parse SSL certificates by configuring expressions that use the following prefix:

`CLIENT.SSL.CLIENT_CERT`

This section discusses the expressions that you can configure for certificates, with the exception of expressions that examine certificate expiration. Time-based operations are described in ["Default Syntax Expressions: Working with Dates, Times, and Numbers."](#)

The following table describes operations that you can specify for the `CLIENT.SSL.CLIENT_CERT` prefix.

Table 3. Operations That Can Be Specified with the `CLIENT.SSL.CLIENT_CERT` Prefix

SSL Certificate Operation	Description
<code>&lt;certificate&gt;.EXISTS</code>	Returns a Boolean TRUE if the client has a certificate.
<code>&lt;certificate&gt;.ISSUER</code>	Returns the Distinguished Name (DN) of the issuer, with a backslash (\) as the delimiter for the name and the value, as in the following example:  Following is an example of the returned value:  /C=US/O=myCompany/OU=www.mycompany.com
<code>&lt;certificate&gt;.ISSUER. IGNORE_EMPTY_ELEMENTS</code>	Returns the Issuer and ignores the empty elements in the issuer's DN.  Cert-Issuer: /c=in/st=kar//l=losangeles/emailAddress=myuserid@mycompany.com  The following Rewrite action returns a string that contains the issuer's DN:  sh rewrite action insert_ssl  Name: insert_ssl



	<p>Operation: insert_http_header</p> <p>Value: CLIENT.SSL.CLIENT_CERT</p> <p>However, if you change the value to the following, the operation returns the following error:</p> <p>CLIENT.SSL.CLIENT_CERT.ISSUER_NAME</p>
<certificate>.AUTH_KEYID	Returns a string that contains the Auth KeyID field of the certificate.
<certificate>.AUTH_KEYID.CERT_SERIALNUMBER	Returns the SerialNumber field of the certificate.
<certificate>.AUTH_KEYID.EXISTS	Returns a Boolean TRUE if the certificate exists.
<certificate>.AUTH_KEYID.ISSUER_NAME	<p>Returns the Issuer Distinguished Name and the value, and the slash (/) character.</p> <p>Following is an example:</p> <p>/C=US/O=myCompany/OU=www.mycompany.com</p>
<certificate>.AUTH_KEYID.ISSUER_NAME.IGNORE_EMPTY_ELEMENTS	<p>Returns the Issuer Distinguished Name and the value, and the slash (/) character.</p> <p>For example, the following name-value pair is returned:</p> <p>a=10;;b=11; ;c=89</p> <p>The element following b=11 is not contained in the output.</p>
<certificate>.AUTH_KEYID.KEYID	Returns the keyIdentifier field of the Auth KeyID.
<certificate>.CERT_POLICY	Returns a string that contains the client certificate policy.
<certificate>.KEY_USAGE(string)	<p>Returns a Boolean value to indicate whether the key usage is set. The string argument is one of the following:</p> <ul style="list-style-type: none"> <li>DIGITAL_SIGNATURE. Returns TRUE.</li> <li>NONREPUDIATION. Returns TRUE.</li> <li>KEYENCIPHERMENT. Returns TRUE.</li> <li>DATAENCIPHERMENT. Returns TRUE.</li> <li>KEYAGREEMENT. Returns TRUE.</li> <li>KEYCERTSIGN. Returns TRUE.</li> <li>CRLSIGN. Returns TRUE if the certificate is a CRL.</li> <li>ENCIPHERONLY. Returns TRUE.</li> <li>DECIPHERONLY. Returns TRUE.</li> </ul>
<certificate>.PK_ALGORITHM	Returns the name of the public key algorithm.
<certificate>.PK_SIZE	Returns the size of the public key used.
<certificate>.SERIALNUMBER	Returns the serial number of the client certificate. If the client certificate is not found, the operation returns an empty string.
<certificate>.SIGNATURE_ALGORITHM	Returns the name of the cryptographic algorithm.

<certificate>.SUBJECT	<p>Returns the Distinguished Name of the slash (â€œ/â€•) delimits name-value pairs.</p> <p>Following is an example:</p> <pre>/C=US/O=myCompany/OU=www.myc.com</pre>
<certificate>.SUBJECT.IGNORE_EMPTY_ELEMENTS	<p>Returns the Subject as a name-value list.</p> <p>Cert-Issuer: /c=in/st=kar//l... /emailAddress=myuserid@mycom...</p> <p>The following Rewrite action returns a list of the subject's name-value pairs.</p> <pre>sh rewrite action insert_ssl</pre> <p>Name: insert_ssl</p> <p>Operation: insert_http_header</p> <p>Value: CLIENT.SSL.CLIENT_CERT</p> <p>However, if you change the value to the issuer's name-value pairs, the action returns the issuer's name-value pairs.</p> <pre>CLIENT.SSL.CLIENT_CERT.ISSUER</pre>
<certificate>.SUBJECT_KEYID	<p>Returns the Subject KeyID of the client certificate.</p>

## Default Syntax Expressions: IP and MAC Addresses, Throughput, VLAN IDs

You can use default syntax expression prefixes that return IPv4 and IPv6 addresses, MAC addresses, IP subnets, useful client and server data such as the throughput rates at the interface ports (Rx, Tx, and RxTx), and the IDs of the VLANs through which packets are received. You can then use various operators to evaluate the data that is returned by these expression prefixes.

This document includes the following details:

- [Expressions for IP Addresses and IP Subnets](#)
- [Expressions for MAC Addresses](#)
- [Expressions for Numeric Client and Server Data](#)

### Expressions for IP Addresses and IP Subnets

Updated: 2013-09-02

You can use default syntax expressions to evaluate addresses and subnets that are in Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6) format. Expression prefixes for IPv6 addresses and subnets include IPv6 in the prefix. Expression prefixes for IPv4 addresses and subnets include IP in the prefix. Following is an example of an expression that identifies whether a request has originated from a particular IPv4 subnet.

```
client.ip.src.in_subnet(147.1.0.0/16)
```

Following are two examples of Rewrite policies that examine the subnet from which the packet is received and perform a rewrite action on the Host header. With these two policies configured, the rewrite action that is performed depends on the subnet in the request. These two policies evaluate IP addresses that are in the IPv4 address format.

```
add rewrite action URL1-rewrite-action replace "http.req.header(\"Host\")" "\"www.mycompany1.com\"
add rewrite policy URL1-rewrite-policy "http.req.header(\"Host\").contains(\"www.test1.com\")"
add rewrite action URL2-rewrite-action replace "http.req.header(\"Host\")" "\"www.mycompany2.com\"
add rewrite policy URL2-rewrite-policy "http.req.header(\"Host\").contains(\"www.test2.com\")"
```

Note: The preceding examples are commands that you type at the NetScaler command-line interface (CLI) and, therefore, each quotation mark must be preceded by a backslash (\). For more information, see ["Configuring Default Syntax Expressions in a Policy."](#)

### Prefixes for IPV4 Addresses and IP Subnets

Updated: 2013-09-02

The following table describes prefixes that return IPv4 addresses and subnets, and segments of IPv4 addresses. You can use numeric operators and operators that are specific to IPv4 addresses with these prefixes. For more information about numeric operations, see ["Basic Operations on Expression Prefixes"](#) and ["Compound Operations for Numbers."](#)

Table 1. Prefixes That Evaluate IP and MAC Addresses

Prefix	Description
CLIENT.IP.SRC	Returns the source IP of the current packet as an IP address or as a number.
CLIENT.IP.DST	Returns the destination IP of the current packet as an IP address or as a number.
SERVER.IP.SRC	Returns the source IP of the current packet as an IP address or as a number.
SERVER.IP.DST	Returns the destination IP of the current packet as an IP address or as a number.

### Operations for IPV4 Addresses

The following table describes the operators that can be used with prefixes that return an IPv4 address.

Table 2. Operations on IPV4 Addresses

Prefix	Description

<code>&lt;ip address&gt;.EQ(&lt;address&gt;)</code>	<p>Returns a Boolean TRUE if the IP address value is same as the &lt;a argument. The following example checks whether the client's destination address is equal to 10.100.10.100:</p> <pre>client.ip.dst.eq(10.100.10.100)</pre>
<code>&lt;ip address&gt;.GET1. . .GET4</code>	<p>Returns a portion of an IP address as a numeric value. For example, if the IP address value is 10.100.200.1, the following is returned:</p> <pre>client.ip.src.get1 Returns 10 client.ip.src.get2 returns 100 client.ip.src.get3 returns 200</pre>
<code>&lt;ip address&gt;.IN_SUBNET(&lt;subnet&gt;)</code>	<p>Returns a Boolean TRUE if the &lt;subnet&gt; argument matches the source IP address value. For example, the following determines whether the destination IP address subnet is 10.100.10.100/18:</p> <pre>client.ip.dst.eq(10.100.10.100/18)</pre>
<code>&lt;ip address&gt;.SUBNET(&lt;n&gt;)</code>	<p>Returns the IP address after applying the subnet mask specified as argument. The subnet mask can take values between 0 and 32.</p> <p>For example:</p> <pre>CLIENT.IP.SRC.SUBNET(24) returns 192.168.1.0 if the IP address represented by the prefix is 192.168.1.10</pre>
<code>&lt;ip address&gt;.IS_IPV6</code>	<p>Returns a Boolean TRUE if this is an Internet Protocol version 6 (IPv6) address. The following is an example:</p> <pre>client.ip.src.is_ipv6</pre>
<code>&lt;ip address&gt;.MATCHES(&lt;hostname&gt;)</code>	<p>Returns a Boolean TRUE if the IP address for the host specified in &lt;hostname&gt; matches the current IP address. The &lt;hostname&gt; can be up to 255 characters.</p>
<code>&lt;ip address&gt;.MATCHES_LOCATION(&lt;location&gt;)</code>	<p>Returns a Boolean TRUE if the location of the IP address matches the &lt;location&gt; argument. The Location string can take the following format: <i>qual2.qual3.qual4.qual5.qual6</i>, for example: NorthAmerica.CA.*</p> <p>Following is an example:</p> <pre>client.ip.src.matches_location(\"Europe.GB.17.London.*\")</pre>

## About IPv6 Expressions

The IPv6 address format allows more flexibility than the older IPv4 format. IPv6 addresses are in the hexadecimal format (RFC 2373). In the following examples, Example 1 is an IPv6 address, Example 2 is a URL that includes the IPv6 address, and Example 3 includes the IPv6 address and a port number.

### Example 1:

```
9901:0ab1:22a2:88a3:3333:4a4b:5555:6666
```

### Example 2:

```
http://[9901:0ab1:22a2:88a3:3333:4a4b:5555:6666]/
```

### Example 3:

```
https://[9901:0ab1:22a2:88a3:3333:4a4b:5555:6666]:8080/
```

In Example 3, the brackets separate the IP address from the port number (8080).

Note that you can only use the '+' operator to combine IPv6 expressions with other expressions. The output is a concatenation of the string values that are returned from the individual expressions. You cannot use any other arithmetic operator with an IPv6 expression. The following syntax is an example:

```
client.ipv6.src + server.ip.dst
```

For example, if the client source IPv6 address is ABCD:1234::ABCD, and the server destination IPv4 address is 10.100.10.100, the preceding expression returns "ABCD:1234::ABCD10.100.10.100".

Note that when the NetScaler appliance receives an IPv6 packet, it assigns a temporary IPv4 address from an unused IPv4 address range and changes the source address of the packet to this temporary address. At response time, the outgoing packet's source address is replaced with the original IPv6 address.

Note: You can combine an IPv6 expression with any other expression except an expression that produces a Boolean result.

## Expression Prefixes for IPv6 Addresses

The IPv6 addresses that are returned by the expression prefixes in the following table can be treated as text data. For example, the prefix `client.ipv6.dst` returns the destination IPv6 address as a string that can be evaluated as text.

The following table describes expression prefixes that return an IPv6 address.

Table 3. IPv6 Expression Prefixes That Return Text

Prefix	Description
<code>CLIENT.IPV6</code>	Operates on the IPv6 address in with the current packet.
<code>CLIENT.IPV6.DST</code>	Returns the IPv6 address in the destination field of the IP header.
<code>CLIENT.IPV6.SRC</code>	Returns the IPv6 address in the source field of the IP header. Following are examples:  <code>client.ipv6.src.in_subnet(2007::2008/64)</code>  <code>client.ipv6.src.get1.le(2008)</code>
<code>SERVER.IPV6</code>	Operates on the IPv6 address in with the current packet.
<code>SERVER.IPV6.DST</code>	Returns the IPv6 address in the destination field of the IP header.
<code>SERVER.IPV6.SRC</code>	Returns the IPv6 address in the source field of the IP header. Following are examples:  <code>server.ipv6.src.in_subnet(2007::2008/64)</code>  <code>server.ipv6.src.get1.le(2008)</code>

## Operations for IPv6 Prefixes

The following table describes the operators that can be used with prefixes that return an IPv6 address:

Table 4. Operations That Evaluate IPv6 Addresses

IPv6 Operation	Description
<code>&lt;ipv6&gt;.EQ(&lt;IPv6_address&gt;)</code>	Returns a Boolean TRUE if the IP address value is same as the <code>&lt;IPv6_address&gt;</code> argument.  Following is an example:  <code>client.ipv6.dst.eq(ABCD:1234::ABCD)</code>
<code>&lt;ipv6&gt;.GET1. . .GET8</code>	Returns a segment of an IPv6 address as a number.

	<p>The following example expressions retrieve segments from the ipv6 address 1000:1001:CD10:0000:0000:89AB:4567:CDEF:</p> <ul style="list-style-type: none"> <li>◦ <code>client.ipv6.dst.get5</code> extracts 0000, which is the fifth set of bits in the address.</li> <li>◦ <code>client.ipv6.dst.get6</code> extracts 89AB.</li> <li>◦ <code>client.ipv6.dst.get7</code> extracts 4567.</li> </ul> <p>You can perform numeric operations on these segments. Note that you cannot perform numeric operations when you retrieve an entire IPv6 address. This is because expressions that return an entire IPv6 address, such as <code>CLIENT.IPV6.SRC</code>, return the address in text format.</p>
<code>&lt;ipv6&gt;.IN_SUBNET(&lt;subnet&gt;)</code>	<p>Returns a Boolean TRUE if the IPv6 address value is in the subnet specified by the <code>&lt;subnet&gt;</code> argument.</p> <p>Following is an example:</p> <pre>client.ipv6.dst.eq(1000:1001:CD10:0000:0000:89AB:4567:CDEF/60)</pre>
<code>&lt;ipv6&gt;.IS_IPV4</code>	<p>Returns a Boolean TRUE if this is an IPv4 client, and returns a Boolean FALSE if it is not.</p>
<code>&lt;ipv6&gt;.SUBNET(&lt;n&gt;)</code>	<p>Returns the IPv6 address after applying the subnet mask specified as the argument. The subnet mask can take values between 0 and 128.</p> <p>For example:</p> <pre>CLIENT.IPV6.SRC.SUBNET(24)</pre>

## Expressions for MAC Addresses

A MAC address consists of colon-delimited hexadecimal values in the format `##:##:##:##:##:##`, where each `â€œ#â€` represents either a number from 0 through 9 or a letter from A through F. Default syntax expression prefixes and operators are available for evaluating source and destination MAC addresses.

### Prefixes for MAC Addresses

The following table describes prefixes that return MAC addresses.

Table 5. Prefixes That Evaluate MAC Addresses

Prefix	Description
<code>client.ether.dstmac</code>	Returns the MAC address in the destination field of the Ethernet header.
<code>client.ether.srcmac</code>	Returns the MAC address in the source field of the Ethernet header.

### Operations for MAC Addresses

The following table describes the operators that can be used with prefixes that return a MAC address.

Table 6. Operations on MAC Addresses

Prefix	Description
<code>&lt;mac address&gt;.EQ(&lt;address&gt;)</code>	Returns a Boolean TRUE if the MAC address value is same as the <code>&lt;address&gt;</code> argument.
<code>&lt;mac address&gt;.GET1. .GET4</code>	Returns a numeric value extracted from the segment of the MAC address that is specified in the GET operation.

For example, if the MAC address is 12:34:56:78:9a:bc, the following returns 34:

```
client.ether.dstmac.get2
```

## Expressions for Numeric Client and Server Data

The following table describes prefixes for working with numeric client and server data, including throughput, port numbers, and VLAN IDs.

Table 7. Prefixes That Evaluate Numeric Client and Server Data

Prefix	Description
<code>client.interface.rxthroughput</code>	Returns an integer representing the raw received traffic throughput in kilobytes per second (KBps) for the previous seven seconds.
<code>client.interface.txthroughput</code>	Returns an integer representing the raw transmitted traffic throughput in KBps for the previous seven seconds.
<code>client.interface.rxtxthroughput</code>	Returns an integer representing the raw received and transmitted traffic throughput in KBps for the previous seven seconds.
<code>server.interface.rxthroughput</code>	Returns an integer representing the raw received traffic throughput in KBps for the previous seven seconds.
<code>server.interface.txthroughput</code>	Returns an integer representing the raw transmitted traffic throughput in KBps for the previous seven seconds.
<code>server.interface.rxtxthroughput</code>	Returns an integer representing the raw received and transmitted traffic throughput in KBps for the previous seven seconds.
<code>server.vlan.id</code>	Returns a numeric ID of the VLAN through which the current packet entered the NetScaler.
<code>client.vlan.id</code>	Returns a numeric ID for the VLAN through which the current packet entered the NetScaler.

## Default Syntax Expressions: Stream Analytics Functions

Stream Analytics expressions begin with the `ANALYTICS.STREAM(<identifier_name>)` prefix. The following list describes the functions that can be used with this prefix.

### `COLLECT_STATS`

Collect statistical data from the requests that are evaluated against the policy and create a record for each request.

### `REQUESTS`

Return the number of requests that exist for the specified record grouping. The value returned is of type unsigned long.

### `BANDWIDTH`

Return the bandwidth statistic for the specified record grouping. The value returned is of type unsigned long.

### `RESPTIME`

Return the response time statistic for the specified record grouping. The value returned is of type unsigned long.

### `CONNECTIONS`

Return the number of concurrent connections that exist for the specified record grouping. The value returned is of type unsigned long.

### `IS_TOP(n)`

Return a Boolean `TRUE` if the statistical value for the specified record grouping is one among the top `n` groups. Otherwise, return a Boolean `FALSE`.

### `CHECK_LIMIT`

Return a Boolean `TRUE` if the statistic for the specified record grouping has hit the preconfigured limit. Otherwise, return a Boolean `FALSE`.



## Default Syntax Expressions: DataStream

The policy infrastructure on the Citrix NetScaler appliance includes expressions that you can use to evaluate and process database server traffic when the appliance is deployed between a farm of application servers and their associated database servers.

This document includes the following details:

- Expressions for the MySQL Protocol
- Expressions for Evaluating Microsoft SQL Server Connections

## Expressions for the MySQL Protocol

The following expressions evaluate traffic associated with MySQL database servers. You can use the request-based expressions (expressions that begin with `MYSQL.CLIENT` and `MYSQL.REQ`) in policies to make request switching decisions at the content switching virtual server bind point and the response-based expressions (expressions that begin with `MYSQL.RES`) to evaluate server responses to user-configured health monitors.

- **MYSQL.CLIENT**. Operates on the client properties of a MySQL connection.
- **MYSQL.CLIENT.CAPABILITIES**. Returns the set of flags that the client has set in the capabilities field of the handshake initialization packet during authentication. Examples of the flags that are set are `CLIENT_FOUND_ROWS`, `CLIENT_COMPRESS`, and `CLIENT_SSL`.
- **MYSQL.CLIENT.CHAR\_SET**. Returns the enumeration constant assigned to the character set that the client uses. The `EQ(<m>)` and `NE(<m>)` operators, which return Boolean values to indicate the result of a comparison, are used with this prefix. Following are the character set enumeration constants:

`LATIN2_CZECH_CS`  
`DEC8_SWEDISH_CI`  
`CP850_GENERAL_CI`  
`GREEK_GENERAL_CI`  
`LATIN1_GERMAN1_CI`  
`HP8_ENGLISH_CI`  
`KOI8R_GENERAL_CI`  
`LATIN1_SWEDISH_CI`  
`LATIN2_GENERAL_CI`  
`SWE7_SWEDISH_CI`  
`ASCII_GENERAL_CI`  
`CP1251_BULGARIAN_CI`  
`LATIN1_DANISH_CI`  
`HEBREW_GENERAL_CI`  
`LATIN7_ESTONIAN_CS`  
`LATIN2_HUNGARIAN_CI`  
`KOI8U_GENERAL_CI`  
`CP1251_UKRAINIAN_CI`  
`CP1250_GENERAL_CI`  
`LATIN2_CROATIAN_CI`  
`CP1257_LITHUANIAN_CI`  
`LATIN5_TURKISH_CI`  
`LATIN1_GERMAN2_CI`  
`ARMSCII8_GENERAL_CI`  
`UTF8_GENERAL_CI`  
`CP1250_CZECH_CS`  
`CP866_GENERAL_CI`  
`KEYBCS2_GENERAL_CI`  
`MACCE_GENERAL_CI`  
`MACROMAN_GENERAL_CI`  
`CP852_GENERAL_CI`  
`LATIN7_GENERAL_CI`  
`LATIN7_GENERAL_CS`  
`MACCE_BIN`  
`CP1250_CROATIAN_CI`  
`LATIN1_BIN`  
`LATIN1_GENERAL_CI`

LATIN1\_GENERAL\_CS  
 CP1251\_BIN  
 CP1251\_GENERAL\_CI  
 CP1251\_GENERAL\_CS  
 MACROMAN\_BIN  
 CP1256\_GENERAL\_CI  
 CP1257\_BIN  
 CP1257\_GENERAL\_CI  
 ARMScii8\_BIN  
 ASCII\_BIN  
 CP1250\_BIN  
 CP1256\_BIN  
 CP866\_BIN  
 DEC8\_BIN  
 GREEK\_BIN  
 HEBREW\_BIN  
 HP8\_BIN  
 KEYBCS2\_BIN  
 KOI8R\_BIN  
 KOI8U\_BIN  
 LATIN2\_BIN  
 LATIN5\_BIN  
 LATIN7\_BIN  
 CP850\_BIN  
 CP852\_BIN  
 SWE7\_BIN  
 UTF8\_BIN  
 GEOSTD8\_GENERAL\_CI  
 GEOSTD8\_BIN  
 LATIN1\_SPANISH\_CI  
 UTF8\_UNICODE\_CI  
 UTF8\_ICELANDIC\_CI  
 UTF8\_LATVIAN\_CI  
 UTF8\_ROMANIAN\_CI  
 UTF8\_SLOVENIAN\_CI  
 UTF8\_POLISH\_CI  
 UTF8\_ESTONIAN\_CI  
 UTF8\_SPANISH\_CI  
 UTF8\_SWEDISH\_CI  
 UTF8\_TURKISH\_CI  
 UTF8\_CZECH\_CI  
 UTF8\_DANISH\_CI  
 UTF8\_LITHUANIAN\_CI  
 UTF8\_SLOVAK\_CI  
 UTF8\_SPANISH2\_CI  
 UTF8\_ROMAN\_CI  
 UTF8\_PERSIAN\_CI  
 UTF8\_ESPERANTO\_CI  
 UTF8\_HUNGARIAN\_CI  
 INVALID\_CHARSET

- **MYSQL.CLIENT.DATABASE.** Returns the name of the database specified in the authentication packet that the client sends to the database server. This is the `databasename` attribute.
- **MYSQL.CLIENT.USER.** Returns the user name (in the authentication packet) with which the client is attempting to connect to the database. This is the `user` attribute.
- **MYSQL.REQ.** Operates on a MySQL request.
- **MYSQL.REQ.COMMAND.** Identifies the enumeration constant assigned to the type of command in the request. The `EQ(<m>)` and `NE(<m>)` operators, which return Boolean values to indicate the result of a comparison, are used with this prefix. Following are the enumeration constant values:

SLEEP  
 QUIT  
 INIT\_DB  
 QUERY

FIELD\_LIST  
 CREATE\_DB  
 DROP\_DB  
 REFRESH  
 SHUTDOWN  
 STATISTICS  
 PROCESS\_INFO  
 CONNECT  
 PROCESS\_KILL  
 DEBUG  
 PING  
 TIME  
 DELAYED\_INSERT  
 CHANGE\_USER  
 BINLOG\_DUMP  
 TABLE\_DUMP  
 CONNECT\_OUT  
 REGISTER\_SLAVE  
 STMT\_PREPARE  
 STMT\_EXECUTE  
 STMT\_SEND\_LONG\_DATA  
 STMT\_CLOSE  
 STMT\_RESET  
 SET\_OPTION  
 STMT\_FETCH

- **MYSQL.REQ.QUERY**. Identifies the query in the MySQL request.
- **MYSQL.REQ.QUERY.COMMAND**. Returns the first keyword in the MySQL query.
- **MYSQL.REQ.QUERY.SIZE**. Returns the size of the request query in integer format. The **SIZE** method is similar to the **CONTENT\_LENGTH** method that returns the length of an HTTP request or response.
- **MYSQL.REQ.QUERY.TEXT**. Returns a string covering the entire query.
- **MYSQL.REQ.QUERY.TEXT(<n>)**. Returns the first *n* bytes of the MySQL query as a string. This is similar to **HTTP.BODY(<n>)**.

#### Parameters:

*n* - Number of bytes to be returned

- **MYSQL.RES**. Operates on a MySQL response.
- **MYSQL.RES.ATLEAST\_ROWS\_COUNT(<i>)**. Checks whether the response has at least *i* number of rows and returns a Boolean **TRUE** or **FALSE** to indicate the result.

#### Parameters:

*i* - Number of rows

- **MYSQL.RES.ERROR**. Identifies the MySQL error object. The error object includes the error number and the error message.
- **MYSQL.RES.ERROR.MESSAGE**. Returns the error message that is retrieved from the server's error response.
- **MYSQL.RES.ERROR.NUM**. Returns the error number that is retrieved from the server's error response.
- **MYSQL.RES.ERROR.SQLSTATE**. Returns the value of the **SQLSTATE** field in the server's error response. The MySQL server translates error number values to **SQLSTATE** values.
- **MYSQL.RES.FIELD(<i>)**. Identifies the packet that corresponds to the *i*<sup>th</sup> individual field in the server's response. Each field packet describes the properties of the associated column. The packet count (*i*) begins at 0.

#### Parameters:

*i* - Packet number

- **MYSQL.RES.FIELD(<i>).CATALOG**. Returns the **catalog** property of the field packet.

- **MYSQL.RES.FIELD(<i>).CHAR\_SET.** Returns the character set of the column. The `EQ(<m>)` and `NE(<m>)` operators, which return Boolean values to indicate the result of a comparison, are used with this prefix.
- **MYSQL.RES.FIELD(<i>).DATATYPE.** Returns an enumeration constant that represents the data type of the column. This is the `type` (also called `enum_field_type`) attribute of the column. The `EQ(<m>)` and `NE(<m>)` operators, which return Boolean values to indicate the result of a comparison, are used with this prefix. The possible values for the various data types are:
  - DECIMAL
  - TINY
  - SHORT
  - LONG
  - FLOAT
  - DOUBLE
  - NULL
  - TIMESTAMP
  - LONGLONG
  - INT24
  - DATE
  - TIME
  - DATETIME
  - YEAR
  - NEWDATE
  - VARCHAR (new in MySQL 5.0)
  - BIT (new in MySQL 5.0)
  - NEWDECIMAL (new in MySQL 5.0)
  - ENUM
  - SET
  - TINY\_BLOB
  - MEDIUM\_BLOB
  - LONG\_BLOB
  - BLOB
  - VAR\_STRING
  - STRING
  - GEOMETRY
- **MYSQL.RES.FIELD(<i>).DB.** Returns the database identifier (`db`) attribute of the field packet.
- **MYSQL.RES.FIELD(<i>).DECIMALS.** Returns the number of positions after the decimal point if the type is `DECIMAL` or `NUMERIC`. This is the `decimals` attribute of the field packet.
- **MYSQL.RES.FIELD(<i>).FLAGS.** Returns the `flags` property of the field packet. Following are the possible hexadecimal flag values:
  - 0001: NOT\_NULL\_FLAG
  - 0002: PRI\_KEY\_FLAG
  - 0004: UNIQUE\_KEY\_FLAG
  - 0008: MULTIPLE\_KEY\_FLAG
  - 0010: BLOB\_FLAG
  - 0020: UNSIGNED\_FLAG
  - 0040: ZEROFILL\_FLAG
  - 0080: BINARY\_FLAG
  - 0100: ENUM\_FLAG
  - 0200: AUTO\_INCREMENT\_FLAG
  - 0400: TIMESTAMP\_FLAG
  - 0800: SET\_FLAG
- **MYSQL.RES.FIELD(<i>).LENGTH.** Returns the length of the column. This is the value of the `length` attribute of the field packet. The value that is returned might be larger than the actual value. For example, an instance of a `VARCHAR(2)` column might return a value of 2 even when it contains only one character.
- **MYSQL.RES.FIELD(<i>).NAME.** Returns the column identifier (the name after the `AS` clause, if any). This is the `name` attribute of the field packet.
- **MYSQL.RES.FIELD(<i>).ORIGINAL\_NAME.** Returns the original column identifier (before the `AS` clause, if any). This is the `org_name` attribute of the field packet.
- **MYSQL.RES.FIELD(<i>).ORIGINAL\_TABLE.** Returns the original table identifier of the column (before the `AS` clause, if any). This is the `org_table` attribute of the field packet.

- **MYSQL.RES.FIELD(<i>).TABLE.** Returns the table identifier of the column (after the AS clause, if any). This is the `table` attribute of the field packet.
- **MYSQL.RES.FIELDS\_COUNT.** Returns the number of field packets in the response (the `field_count` attribute of the OK packet).
- **MYSQL.RES.OK.** Identifies the OK packet sent by the database server.
- **MYSQL.RES.OK.AFFECTED\_ROWS.** Returns the number of rows affected by an INSERT, UPDATE, or DELETE query. This is the value of the `affected_rows` attribute of the OK packet.
- **MYSQL.RES.OK.INSERT\_ID.** Identifies the `unique_id` attribute of the OK packet. If an auto-increment identity is not generated by the current MySQL statement or query, the value of `unique_id`, and hence the value returned by the expression, is 0.
- **MYSQL.RES.OK.MESSAGE.** Returns the message property of the OK packet.
- **MYSQL.RES.OK.STATUS.** Identifies the bit string in the `server_status` attribute of the OK packet. Clients can use the server status to check whether the current command is a part of a running transaction. The bits in the `server_status` bit string correspond to the following fields (in the given order):
  - IN TRANSACTION
  - AUTO\_COMMIT
  - MORE\_RESULTS
  - MULTI\_QUERY
  - BAD\_INDEX\_USED
  - NO\_INDEX\_USED
  - CURSOR\_EXISTS
  - LAST\_ROW\_SEEN
  - DATABASE\_DROPPED
  - NO\_BACKSLASH\_ESCAPES
- **MYSQL.RES.OK.WARNING\_COUNT.** Returns the `warning_count` attribute of the OK packet.
- **MYSQL.RES.ROW(<i>).** Identifies the packet that corresponds to the  $i^{\text{th}}$  individual row in the database server's response.

#### Parameters:

$i$  - Row number

- **MYSQL.RES.ROW(<i>).DOUBLE\_ELEM(<j>).** Checks whether the  $j^{\text{th}}$  column of the  $i^{\text{th}}$  row of the table is NULL. Following C conventions, both indexes  $i$  and  $j$  start from 0. Therefore, row  $i$  and column  $j$  are actually the  $(i+1)^{\text{th}}$  row and the  $(j+1)^{\text{th}}$  column, respectively.

#### Parameters:

$i$  - Row number

$j$  - Column number

- **MYSQL.RES.ROW(<i>).IS\_NULL\_ELEM( $j$ ).** Checks whether the  $j^{\text{th}}$  column of the  $i^{\text{th}}$  row of the table is NULL. Following C conventions, both indexes  $i$  and  $j$  start from 0. Therefore, row  $i$  and column  $j$  are actually the  $(i+1)^{\text{th}}$  row and the  $(j+1)^{\text{th}}$  column, respectively.

#### Parameters:

$i$  - Row number

$j$  - Column number

- **MYSQL.RES.ROW(<i>).NUM\_ELEM(<j>).** Returns an integer value from the  $j^{\text{th}}$  column of the  $i^{\text{th}}$  row of the table. Following C conventions, both indexes  $i$  and  $j$  start from 0. Therefore, row  $i$  and column  $j$  are actually the  $(i+1)^{\text{th}}$  row and the  $(j+1)^{\text{th}}$  column, respectively.

#### Parameters:

$i$  - Row number

$j$  - Column number

- o **MYSQ.L.RES.ROW(<i>).TEXT\_ELEM(j)**. Returns a string from the  $j^{\text{th}}$  column of the  $i^{\text{th}}$  row of the table. Following C conventions, both indexes  $i$  and  $j$  start from 0. Therefore, row  $i$  and column  $j$  are actually the  $(i+1)^{\text{th}}$  row and the  $(j+1)^{\text{th}}$  column, respectively.

#### Parameters:

$i$  - Row number

$j$  - Column number

- o **MYSQ.L.RES.TYPE**. Returns an enumeration constant for the response type. Its values can be `ERROR`, `OK`, and `RESULT_SET`. The `EQ(<m>)` and `NE(<m>)` operators, which return Boolean values to indicate the result of a comparison, are used with this prefix.

## Expressions for Evaluating Microsoft SQL Server Connections

The following expressions evaluate traffic associated with Microsoft SQL Server database servers. You can use the request-based expressions (expressions that begin with `MSSQL.CLIENT` and `MSSQL.REQ`) in policies to make request switching decisions at the content switching virtual server bind point and the response-based expressions (expressions that begin with `MSSQL.RES`) to evaluate server responses to user-configured health monitors.

Table 1. Expressions for Evaluating Microsoft SQL Server Connections

Expression	Description
<code>MSSQL.CLIENT.CAPABILITIES</code>	Returns the <code>OptionFlags1</code> , <code>OptionFlags2</code> , <code>OptionFlags3</code> , and <code>TypeFlags</code> fields of the <code>LOGIN7</code> authentication packet, in that order, as a 4-byte integer. Each field is 1 byte long and specifies a set of client capabilities.
<code>MSSQL.CLIENT.DATABASE</code>	Returns the name of the client database. The value returned is of type <code>text</code> .
<code>MSSQL.CLIENT.USER</code>	Returns the user name with which the client authenticated. The value returned is of type <code>text</code> .
<code>MSSQL.REQ.COMMAND</code>	<p>Returns an enumeration constant that identifies the type of command in the request sent to a Microsoft SQL Server database server. The value returned is of type <code>text</code>.</p> <p>Examples of the values of the enumeration constant are <code>QUERY</code>, <code>RESPONSE</code>, <code>RPC</code>, and <code>ATTENTION</code>.</p> <p>The <code>EQ(&lt;m&gt;)</code> and <code>NE(&lt;m&gt;)</code> operators, which return Boolean values to indicate the result of a comparison, are used with this expression.</p>
<code>MSSQL.REQ.QUERY.COMMAND</code>	Returns the first keyword in the SQL query. The value returned is of type <code>text</code> .
<code>MSSQL.REQ.QUERY.SIZE</code>	Returns the size of the SQL query in the request. The value returned is a number.
<code>MSSQL.REQ.QUERY.TEXT</code>	Returns the entire SQL query as a string. The value returned is of type <code>text</code> .
<code>MSSQL.REQ.QUERY.TEXT(&lt;n&gt;)</code>	<p>Returns the first <math>n</math> bytes of the SQL query. The value returned is of type <code>text</code>.</p> <p><b>Parameters:</b></p> <p><math>n</math> - Number of bytes</p>

<code>MSSQL.REQ.RPC.NAME</code>	Returns the name of the procedure that is being called in a remote procedure call (RPC) request. The name is returned as a string.
<code>MSSQL.REQ.RPC.IS_PROCID</code>	Returns a Boolean value that indicates whether the remote procedure call (RPC) request contains a procedure ID or an RPC name. A return value of <code>TRUE</code> indicates that the request contains a procedure ID and a return value of <code>FALSE</code> indicates that the request contains an RPC name.
<code>MSSQL.REQ.RPC.PROCID</code>	Returns the procedure ID of the remote procedure call (RPC) request as an integer.
<code>MSSQL.REQ.RPC.BODY</code> Note: Not available for releases before 10.1.	Returns the body of the SQL request as a string in the form of parameters represented as "a=b" clauses separated by commas, where "a" is the RPC parameter name and "b" is its value.
<code>MSSQL.REQ.RPC.BODY(n)</code> Note: Not available for releases before 10.1.	Returns part of the body of the SQL request as a string in the form of parameters represented as "a=b" clauses separated by commas, where "a" is the RPC parameter name and "b" is its value. Parameters are returned from only the first "n" bytes of the request, skipping the SQL header. Only complete name-value pairs are returned.
<code>MSSQL.RES.ATLEAST_ROWS_COUNT(i)</code>	Checks whether the response has at least <i>i</i> number of rows. The value returned is a Boolean <code>TRUE</code> or <code>FALSE</code> value.  <b>Parameters:</b>  <i>i</i> - Number of rows
<code>MSSQL.RES.DONE.ROWCOUNT</code>	Returns a count of the number of rows affected by an <code>INSERT</code> , <code>UPDATE</code> , or <code>DELETE</code> query. The value returned is of type <code>unsigned long</code> .
<code>MSSQL.RES.DONE.STATUS</code>	Returns the status field from the <code>DONE</code> token sent by a Microsoft SQL Server database server. The value returned is a number.
<code>MSSQL.RES.ERROR.MESSAGE</code>	Returns the error message from the <code>ERROR</code> token sent by a Microsoft SQL Server database server. This is the value of the <code>MsgText</code> field in the <code>ERROR</code> token. The value returned is of type <code>text</code> .
<code>MSSQL.RES.ERROR.NUM</code>	Returns the error number from the <code>ERROR</code> token sent by a Microsoft SQL Server database server. This is the value of the <code>Number</code> field in the <code>ERROR</code> token. The value returned is a number.
<code>MSSQL.RES.ERROR.STATE</code>	Returns the error state from the <code>ERROR</code> token sent by a Microsoft SQL Server database server. This is the value of the <code>State</code> field in the <code>ERROR</code> token. The value returned is a number.
<code>MSSQL.RES.FIELD(&lt;i&gt;).DATATYPE</code>	Returns the data type of the <i>i</i> <sup>th</sup> field in the server response. The <code>EQ(&lt;m&gt;)</code> and <code>NE(&lt;m&gt;)</code> functions, which return Boolean values to indicate the result of a comparison, are used with this prefix.  For example, the following expression returns a Boolean <code>TRUE</code> if the <code>DATATYPE</code> function returns a value of <code>datetime</code> for the third field in the response:

	<p><code>MSSQL.RES.FIELD(&lt;2&gt;).DATATYPE.EQ(datetime)</code></p> <p><b>Parameters:</b></p> <p><code>i</code> - Row number</p>
<code>MSSQL.RES.FIELD(&lt;i&gt;).LENGTH</code>	<p>Returns the maximum possible length of the <math>i^{\text{th}}</math> field in the server response. The value returned is a number.</p> <p><b>Parameters:</b></p> <p><code>i</code> - Row number</p>
<code>MSSQL.RES.FIELD(&lt;i&gt;).NAME</code>	<p>Returns the name of the <math>i^{\text{th}}</math> field in the server response. The value returned is of type <code>text</code>.</p> <p><b>Parameters:</b></p> <p><code>i</code> - Row number</p>
<code>MSSQL.RES.ROW(&lt;i&gt;).DOUBLE_ELEM(&lt;j&gt;)</code>	<p>Returns a value of type <code>double</code> from the <math>j^{\text{th}}</math> column of the <math>i^{\text{th}}</math> row of the table. If the value is not a double value, an <code>UNDEF</code> condition is raised. Following C conventions, both indexes <code>i</code> and <code>j</code> start from 0 (zero). Therefore, row <code>i</code> and column <code>j</code> are actually the <math>(i + 1)^{\text{th}}</math> row and the <math>(j + 1)^{\text{th}}</math> column, respectively.</p> <p><b>Parameters:</b></p> <p><code>i</code> - Row number</p> <p><code>j</code> - Column number</p>
<code>MSSQL.RES.ROW(&lt;i&gt;).NUM_ELEM(j)</code>	<p>Returns an integer value from the <math>j^{\text{th}}</math> column of <math>i^{\text{th}}</math> row of the table. If the value is not an integer value, an <code>UNDEF</code> condition is raised. Following C conventions, both indexes <code>i</code> and <code>j</code> start from 0 (zero). Therefore, row <code>i</code> and column <code>j</code> are actually the <math>(i + 1)^{\text{th}}</math> row and the <math>(j + 1)^{\text{th}}</math> column, respectively.</p> <p><b>Parameters:</b></p> <p><code>i</code> - Row number</p> <p><code>j</code> - Column number</p>
<code>MSSQL.RES.ROW(&lt;i&gt;).IS_NULL_ELEM(j)</code>	<p>Checks whether the <math>j^{\text{th}}</math> column of the <math>i^{\text{th}}</math> row of the table is <code>NULL</code> and returns a Boolean <code>TRUE</code> or <code>FALSE</code> to indicate the result. Following C conventions, both indexes <code>i</code> and <code>j</code> start from 0 (zero). Therefore, row <code>i</code> and column <code>j</code> are actually the <math>(i + 1)^{\text{th}}</math> row and the <math>(j + 1)^{\text{th}}</math> column, respectively.</p> <p><b>Parameters:</b></p> <p><code>i</code> - Row number</p> <p><code>j</code> - Column number</p>
<code>MSSQL.RES.ROW(&lt;i&gt;).TEXT_ELEM(j)</code>	<p>Returns a text string from the <math>j^{\text{th}}</math> column of <math>i^{\text{th}}</math> row of the table. Following C conventions, both indexes <code>i</code> and <code>j</code> start from 0 (zero). Therefore, row <code>i</code> and column <code>j</code> are actually the <math>(i + 1)^{\text{th}}</math> row and the <math>(j + 1)^{\text{th}}</math> column, respectively.</p>



	<b>Parameters:</b>  i - Row number  j - Column number
MSSQL.RES.TYPE	Returns an enumeration constant that identifies the response type. Following are the possible return values: <ul style="list-style-type: none"> <li>• ERROR</li> <li>• OK</li> <li>• RESULT_SET</li> </ul> The EQ ( <m> ) and NE ( <m> ) operators, which return Boolean values to indicate the result of a comparison, are used with this expression.

## Typecasting Data

You can extract data of one type (for example, text or an integer) from requests and responses and transform it to data of another type. For example, you can extract a string and transform the string to time format. You can also extract a string from an HTTP request body and treat it like an HTTP header or extract a value from one type of request header and insert it in a response header of a different type.

After typecasting the data, you can apply any operation that is appropriate for the new data type. For example, if you typecast text to an HTTP header, you can apply any operation that is applicable to HTTP headers to the returned value.

The following table describes various typecasting operations.

Table 1. Typecasting Functions

Function	Description
<pre>&lt;text&gt;.TYPECAST_LIST_T(&lt;separator&gt;)</pre>	<p>Treats the text in an HTTP request or response character in the &lt;separator&gt; argument. Index va</p> <p>Text mode settings have no effect on the separa IGNORECASE, and the separator is the letter â separator.</p> <p>The following example creates a Rewrite action extracts the fourth item in the list:</p> <pre>add rewrite action myreplace_action 'http.req.body(100).typecast_list_  set rewrite policy myreplace_policy</pre> <p>This policy returns the string â€œfourth itemâ€•</p> <pre>GET?first item?second item?third it</pre> <p>The following example extracts the fourth-from-l</p> <pre>add rewrite action myreplace_action 'http.req.body(100).typecast_list_  set rewrite policy myreplace_policy</pre> <p>This policy returns the string â€œfirst itemâ€• fr</p> <pre>GET?first item?second item?third it</pre>
<pre>&lt;text&gt;.TYPECAST_NVLIST_T(&lt;separator&gt;, &lt;delimiter&gt;)</pre> <p>or</p> <pre>text.TYPECAST_NVLIST_T(&lt;separator&gt;, &lt;delimiter&gt;, &lt;quote&gt;)</pre>	<p>Treats the text as a name-value list. The &lt;separ name and the value. The &lt;delimiter&gt; argument pair. The &lt;quote&gt; character is required when ty strings. Any delimiters that appear within the qu</p> <p>The text mode has no effect on the delimiters. F you specify â€œâ€• as the delimiter, an upper</p> <p>For example, the following policy counts the nur named name-value-count:</p> <pre>add rewrite action mycount_action i: 'http.req.header("Cookie").typecas  set rewrite policy mycount_policy -</pre> <p>This policy can extract a count of arguments in ( count header:</p> <pre>Cookie: name=name1; rank=rank1</pre>

<pre>&lt;text&gt;.TYPECAST_TIME_T</pre>	<p>Treats the designated text as a date string. The</p> <ul style="list-style-type: none"> <li>o RFC822: Sun, 06 Nov 1994 08:49:37 C</li> <li>o RFC850: Sunday, 06-Nov-94 08:49:37</li> <li>o ASCII TIME: Sun Nov 6 08:49:37 1994</li> <li>o HTTP Set-Cookie Expiry date: Sun, 06-</li> </ul> <p>For example, the following policy converts the s matches all requests that have a day value less</p> <pre>Add rewrite policy mytime_policy "h .typecast_time_t.day.le(10)" mytime bind rewrite global mytime_policy 1</pre>
<pre>&lt;numeric string&gt;.TYPECAST_IP_ ADDRESS_T</pre>	<p>Treats a numeric string as an IP address.</p> <p>For example, the following policy matches HTTP 12.34.56.78\r\n.</p> <pre>set rewrite policy ip_check_policy .value("ip").typecast_ip_address_t. bind rewrite global ip_check_policy</pre>
<pre>&lt;numeric string&gt;.TYPECAST_IPV6_ADDRESS_T</pre>	<p>Treats a string as an IPv6 address in the followi</p> <p>0000:0000:CD00:0000:0000:00AB:0000:CDEF</p>
<pre>&lt;text&gt;.TYPECAST_HTTP_ URL_T</pre>	<p>Treats the designated text as the URL in the first [ &lt;protocol&gt;://&lt;hostname&gt;]&lt;path&gt;?&lt;qu default.</p> <p>For example, the following policy replaces a UR</p> <pre>add rewrite action replace_header_s "http.req.header(\"Test\").typecas .before_str(\"123\").after_str(\"AB add rewrite policy rewrite_test_he bind rewrite global rewrite_test_he.</pre> <p>Consider the following header:</p> <pre>Test: ABC%12123\r\n</pre> <p>This policy would replace the preceding header</p>
<pre>&lt;text&gt;.TYPECAST_HTTP_ HOSTNAME_T</pre>	<p>Provides operations for parsing an HTTP host n name is abc.def.com:8080.</p>
<pre>&lt;text&gt;.TYPECAST_HTTP_ METHOD_T</pre>	<p>Converts text to an HTTP method.</p> <p>For example, the following policy matches any t equal to POST:</p> <pre>Add rewrite policy method_policy "h .typecast_http_method_t.eq(POST)" a</pre>

<code>&lt;text&gt;.TYPECAST_DNS_ DOMAIN_T</code>	Enables the designated text to be parsed like a
<code>&lt;text&gt;.TYPECAST_HTTP_ HEADER_T( "&lt;name&gt;" )</code>	<p>Converts the designated text to a multi-line HTTP header.</p> <p>For example, the following expression converts</p> <pre>http.req.header( "MyHeader" ).typcast.</pre> <p>Typically, text operations that you specify in this header, with some exceptions. For example, the in instances of this header type.</p>
<code>&lt;text&gt;.TYPECAST_COOKIE_T</code>	<p>Treats the designated text as an HTTP cookie and can apply name-value list operations as well as can designate equals (=) as the name-value delimiter.</p> <p>If you apply name-value list operations, the list is effect.</p> <p>Each cookie begins with a cookie-name=cookie-value pairs that are separated by a semicolon, as follows:</p> <pre>cookie1=value1;version=n.n;value;do</pre> <p>If the same attribute appears more than once in the cookie, only the last one is returned.</p>
<code>&lt;number&gt;.TYPECAST_DOUBLE_AT</code>	Transforms the number to a value of data type c
<code>&lt;number&gt;.TYPECAST_IP_ADDRESS_AT</code>	Converts the number to an IP address.
<code>&lt;number&gt;.TYPECAST_TIME_AT</code>	Converts the number to time format.
<code>&lt;number&gt;.TYPECAST_TIME_AT.BETWEEN( &lt;time1&gt; , &lt;time2&gt; )</code>	<p>Returns a Boolean value (TRUE or FALSE) that is between the lower and upper time value arguments.</p> <p>The following are prerequisites for this function:</p> <ul style="list-style-type: none"> <li>Both the lower and upper time argument is fully specified. But GMT Jan, GMT 19</li> <li>Both arguments must be either GMT or</li> <li>The day of the week must not be present and be specified as the first, second, third, or third Wednesday of the month).</li> <li>The upper time argument, &lt;time2&gt;, must</li> </ul> <p>The following examples assume that the current day is the first Sunday of the month of May in 2005 example.</p> <pre>BETWEEN(GMT 2004, GMT 2006): TRUE BETWEEN(GMT 2004 Jan, GMT 2006 Nov) BETWEEN(GMT 2004 Jan, GMT 2006): TR BETWEEN(GMT 2005 May Sun_1, GMT 200 BETWEEN(GMT 2005 May 1, GMT May 200 BETWEEN(LOCAL 2005 May 1, LOCAL May on the NetScaler system's timezone</pre> <p>Parameters:</p> <p>&lt;time1&gt; - Lower time value</p> <p>&lt;time2&gt; - Upper time value</p>

<code>&lt;number&gt;.TYPECAST_TIME_AT.DAY</code>	<p>Extracts the day of the month from the current s corresponds to the day of the month. The return</p>
<code>&lt;number&gt;.TYPECAST_TIME_AT.EQ(&lt;t&gt;)</code>	<p>Returns a Boolean value (TRUE or FALSE) that is equal to the time value argument &lt;t&gt;.</p> <p>The following examples assume that the current day is the 1st Sunday of the month of May in 20 example.</p> <pre>EQ(GMT 2005): TRUE EQ(GMT 2005 Dec): FALSE EQ(Local 2005 May): TRUE or FALSE, ( EQ(GMT 10h): TRUE EQ(GMT 10h 30s): TRUE EQ(GMT May 10h): TRUE EQ(GMT Sun): TRUE EQ(GMT May Sun_1): TRUE</pre> <p>Parameters:</p> <p>&lt;t&gt; - Time</p>
<code>&lt;number&gt;.TYPECAST_TIME_AT.GE(&lt;t&gt;)</code>	<p>Returns a Boolean value (TRUE or FALSE) that is greater than or equal to the time value argum</p> <p>The following examples assume that the current day is the 1st Sunday of the month of May in 20 example.</p> <pre>GE(GMT 2004): TRUE GE(GMT 2005 Jan): TRUE GE(Local 2005 May): TRUE or FALSE, ( GE(GMT 8h): TRUE GE(GMT 30m): FALSE GE(GMT May 10h): TRUE GE(GMT May 10h 0m): TRUE GE(GMT Sun): TRUE GE(GMT May Sun_1): TRUE</pre> <p>Parameters:</p> <p>&lt;t&gt; - Time</p>
<code>&lt;number&gt;.TYPECAST_TIME_AT.GT(&lt;t&gt;)</code>	<p>Returns a Boolean value (TRUE or FALSE) that is greater than the time value argument &lt;t&gt;.</p> <p>The following examples assume that the current day is the 1st Sunday of the month of May in 20 example.</p> <pre>GT(GMT 2004): TRUE GT(GMT 2005 Jan): TRUE GT(Local 2005 May): TRUE or FALSE, ( GT(GMT 8h): TRUE GT(GMT 30m): FALSE GT(GMT May 10h): FALSE GT(GMT May 10h 0m): TRUE GT(GMT Sun): FALSE GT(GMT May Sun_1): FALSE</pre> <p>Parameters:</p> <p>&lt;t&gt; - Time</p>

<code>&lt;number&gt;.TYPECAST_TIME_AT.HOURS</code>	Extracts the hour from the current system time and returns a value in the range from 0 to 23.
<code>&lt;number&gt;.TYPECAST_TIME_AT.LE(&lt;t&gt;)</code>	<p>Returns a Boolean value (TRUE or FALSE) that is lesser than or equal to the time value argument &lt;t&gt;.</p> <p>The following examples assume that the current day is the 1st Sunday of the month of May in 2006.</p> <pre>LE(GMT 2006): TRUE LE(GMT 2005 Dec): TRUE LE(Local 2005 May): TRUE or FALSE, (depending on locale) LE(GMT 8h): FALSE LE(GMT 30m): TRUE LE(GMT May 10h): TRUE LE(GMT Jun 11h): TRUE LE(GMT Wed): TRUE LE(GMT May Sun_1): TRUE</pre> <p>Parameters:</p> <p>&lt;t&gt; - Time</p>
<code>&lt;number&gt;.TYPECAST_TIME_AT.LT(&lt;t&gt;)</code>	<p>Returns a Boolean value (TRUE or FALSE) that is lesser than the time value argument &lt;t&gt;.</p> <p>The following examples assume that the current day is the 1st Sunday of the month of May in 2006.</p> <pre>LT(GMT 2006): TRUE LT(GMT 2005 Dec): TRUE LT(Local 2005 May): TRUE or FALSE, (depending on locale) LT(GMT 8h): FALSE LT(GMT 30m): TRUE LT(GMT May 10h): FALSE LT(GMT Jun 11h): TRUE LT(GMT Wed): TRUE LT(GMT May Sun_1): FALSE</pre> <p>Parameters:</p> <p>&lt;t&gt; - Time</p>
<code>&lt;number&gt;.TYPECAST_TIME_AT.MINUTES</code>	Extracts the minute from the current system time and returns a value from 0 to 59.
<code>&lt;number&gt;.TYPECAST_TIME_AT.MONTH</code>	Extracts the month from the current system time and returns a value from 1 (January) to 12 (December).
<code>&lt;number&gt;.TYPECAST_TIME_AT.RELATIVE_BOOT</code>	Calculates the number of seconds that have elapsed since the last scheduled reboot, depending on the current time. If the closest boot time is in the past, the value is negative. If the closest boot time is in the future (scheduled reboot time), the integer is positive.
<code>&lt;number&gt;.TYPECAST_TIME_AT.RELATIVE_NOW</code>	Calculates the number of seconds between the current time and the designated time. If the designated time is in the future, the integer is positive. If the designated time is in the past, the integer is negative.
<code>&lt;number&gt;.TYPECAST_TIME_AT.SECONDS</code>	Extracts the seconds from the current system time and returns a value from 0 to 59.

<code>&lt;number&gt;.TYPECAST_TIME_AT.WEEKDAY</code>	Returns an integer that corresponds to the day of the week.
<code>&lt;number&gt;.TYPECAST_TIME_AT.WITHIN(&lt;time1&gt;,&lt;time2&gt;)</code>	<p>Returns a Boolean value (TRUE or FALSE) that indicates whether the time value designated by &lt;time1&gt; lies within all the ranges defined by lower and upper time values.</p> <p>If an element of time such as the day or the hour is assumed to have the lowest value possible for its range, the upper time value is assumed to have the highest value possible for its range.</p> <p>If the year is specified in one of the arguments, the time values are assumed to be in the same year.</p> <p>Following are the ranges for different elements of time:</p> <ul style="list-style-type: none"> <li>month: 1-12</li> <li>day: 1-31</li> <li>weekday: 0-6</li> <li>hour: 0-23</li> <li>minutes: 0-59</li> <li>seconds: 0-59.</li> </ul> <p>Each element of time in the lower time value argument must correspond to the same element in the upper time value argument. If the time value designated by &lt;number&gt; must lie within the range defined by the upper arguments.</p> <p>The following examples assume that the current time is the second Tuesday of the month. The first example shows how to determine if the current time is within the range of GMT 2004 to GMT 2006.</p> <pre>WITHIN(GMT 2004, GMT 2006): TRUE WITHIN(GMT 2004 Jan, GMT 2006 Mar): TRUE WITHIN(GMT Feb, GMT): TRUE (May fall outside the range) WITHIN(GMT Sun_1, GMT Sun_3): TRUE (the 1st and the 3rd Sunday.)</pre> <p>The second example shows how to determine if the current time is within the range of GMT 2005 May 1 10h to GMT May 1 10h.</p> <pre>WITHIN(GMT 2005 May 1 10h, GMT May 1 10h): TRUE WITHIN(LOCAL 2005 May 1, LOCAL May 1): TRUE (on the NetScaler system's timezone)</pre> <p>Parameters:</p> <p>&lt;time1&gt; - Lower time value</p> <p>&lt;time2&gt; - Upper time value</p>
<code>&lt;number&gt;.TYPECAST_TIME_AT.YEAR</code>	Extracts the year from the current system time and returns it as an integer.
<code>&lt;prefix&gt;.TYPECAST_NUM_T(&lt;type&gt;)</code>	<p>Casts numeric string data to a signed 32-bit numeric value.</p> <ul style="list-style-type: none"> <li>DECIMAL. Treat the string as a decimal value.</li> <li>HEX. Treat the string as a hexadecimal value.</li> <li>DECIMAL_PREFIX. Consider the part of the string before the first non-valid decimal character and cast it as a decimal value.</li> <li>HEX_PREFIX. Consider the part of the string before the first non-valid hexadecimal character and cast it as a hexadecimal value.</li> </ul> <p>For example, the following policy extracts a numeric value from the query string and inserts an HTTP header named Company with that value.</p> <pre>add rewrite action myadd_action ins: "http.req.url.query.typecast_num_t" add rewrite policy myadd_policy true bind rewrite global myadd_policy 30</pre>

For example, this policy would extract "4444" from the URL `/test/file.html?4444`.

The action that is associated with the policy would be `Company: 4448\r\n`.

<code>&lt;prefix&gt;.TYPECAST_NUM_AT</code>	Casts a number of any data type to a number of
<code>&lt;prefix&gt;.TYPECAST_DOUBLE_AT</code>	Casts a number of any data type to a number of
<code>&lt;prefix&gt;.TYPECAST_UNSIGNED_LONG_AT</code>	Casts a number of any data type to a number of
<code>&lt;prefix&gt;.TYPECAST_NUM_T(&lt;type&gt;,&lt;default&gt;)</code>	Casts string data to a signed 32-bit number. If the condition, the function returns the value specified for <code>TYPECAST_NUM_T(&lt;type&gt;)</code> .
<code>&lt;prefix&gt;.TYPECAST_UNSIGNED_LONG_T(&lt;type&gt;)</code>	<p>Casts string data to data of type unsigned long.</p> <ul style="list-style-type: none"> <li>DECIMAL. Treat the string as a decimal</li> <li>HEX. Treat the string as a hexadecimal</li> <li>DECIMAL_PREFIX. Consider the part of the string that is not a valid decimal character and cast it as a decimal</li> <li>HEX_PREFIX. Consider the part of the string that is not a valid hexadecimal character and cast it as a hexadecimal</li> </ul>
<code>&lt;prefix&gt;.TYPECAST_UNSIGNED_LONG_T(&lt;type&gt;,&lt;default&gt;)</code>	Casts string data to data of type unsigned long. If the condition, the function returns the value specified for <code>TYPECAST_UNSIGNED_LONG_T(&lt;type&gt;)</code> .



## Regular Expressions

When you want to perform string matching operations that are more complex than the operations that you perform with the `CONTAINS(â€œ<string>â€•)` or `EQ(â€œ<string>â€•)` operators, you use regular expressions. The policy infrastructure on the Citrix® NetScaler® appliance includes operators to which you can pass regular expressions as arguments for text matching. The names of the operators that work with regular expressions include the string `REGEX`. The regular expressions that you pass as arguments must conform to the regular expression syntax that is described in "<http://www.pcre.org/pcre.txt>." You can learn more about regular expressions at "<http://www.regular-expressions.info/quickstart.html>" and at "<http://www.silverstones.com/thebat/Regex.html>."

The target text for an operator that works with regular expressions can be either text or the value of an HTTP header. Following is the format of a default syntax expression that uses a regular expression operator to operate on text:

```
<text>.<regex_operator>(re<delimiter><regex_pattern><delimiter>)
```

The string `<text>` represents the default syntax expression prefix that identifies a text string in a packet (for example, `HTTP.REQ.URL`). The string `<regex_operator>` represents the regular expression operator. The regular expression always begins with the string `re`. A pair of matching delimiters, represented by `<delimiter>`, enclose the string `<regex_pattern>`, which represents the regular expression.

The following example expression checks whether the URL in an HTTP packet contains the string `*.jpeg` (where `*` is a wildcard) and returns a Boolean `TRUE` or `FALSE` to indicate the result. The regular expression is enclosed within a pair of slash marks (`/`), which act as delimiters.

```
http.req.url.regex_match(re/*.jpeg/)
```

Regular expression operators can be combined to define or refine the scope of a search. For example, `<text> . AFTER_REGEX(re/regex_pattern1/).BEFORE_REGEX(re/regex_pattern2/)` specifies that the target for string matching is the text between the patterns `regex_pattern1` and `regex_pattern2`. You can use a text operator on the scope that is defined by the regular expression operators. For example, you can use the `CONTAINS(â€œ<string>â€•)` operator to check whether the defined scope contains the string `abc`:

```
<text> .AFTER_REGEX(re/regex_pattern1/).BEFORE_REGEX(re/regex_pattern2/).CONTAINS(â€œabcâ€•)
```

**Note:** The process of evaluating a regular expression inherently takes more time than that for an operator such as `CONTAINS(â€œ<string>â€•)` or `EQ(â€œ<string>â€•)`, which work with simple string arguments. You should use regular expressions only if your requirement is beyond the scope of other operators.

## Basic Characteristics of Regular Expressions

Following are notable characteristics of regular expressions as defined on the NetScaler appliance:

- A regular expression always begins with the string `re#` followed by a pair of delimiting characters (called delimiters) that enclose the regular expression that you want to use.

For example, `re#<regex_pattern>#` uses the number sign (#) as a delimiter.

- A regular expression cannot exceed 1499 characters.
- Digit matching can be done by using the string `\d` (a backslash followed by d).
- White space can be represented by using `\s` (a backslash followed by s).
- A regular expression can contain white spaces.

Following are the differences between the NetScaler syntax and the PCRE syntax:

- The NetScaler does not allow back references in regular expressions.
- You should not use recursive regular expressions.
- The dot meta-character also matches the newline character.
- Unicode is not supported.
- The operation `SET_TEXT_MODE(IGNORECASE)` overrides the `(?i)` internal option in the regular expression.

## Operations for Regular Expressions

The following table describes the operators that work with regular expressions. The operation performed by a regular expression operator in a given default syntax expression depends on whether the expression prefix identifies text or HTTP headers. Operations that evaluate headers override any text-based operations for all instances of the specified header type. When you use an operator, replace <text> with the default syntax expression prefix that you want to configure for identifying text.

Table 1. Default Syntax Expression Operators That Work with Regular Expressions

Regular Expression Operation	Description
<text>.BEFORE_REGEX (<regular expression>)	<p>Selects the text that precedes the string that matches the &lt;regular expression&gt;. If the regular expression does not match any data in the target, the expression text object of length 0.</p> <p>The following expression selects the string "text" from "text/plain".</p> <pre>http.res.header("content-type").before_regex(re#/#)</pre>
<text>.AFTER_REGEX (<regular expression>)	<p>Selects the text that follows the string that matches the &lt;regular expression&gt; at the regular expression does not match any text in the target, the expression ret object of length 0.</p> <p>The following expression extracts "Example" from "myExample":</p> <pre>http.req.header("etag").after_regex(re/my/)</pre>
<text>.REGEX_SELECT (<regular expression>)	<p>Selects a string that matches the &lt;regular expression&gt; argument. If the regular expression does not match the target, a text object of length 0 is returned.</p> <p>The following example extracts the string "NS-CACHE-9.0: 90" from a Via header:</p> <pre>http.req.header("via").regex_select(re!NS-CACHE-\d\.\d:\s*\{1,3}!)</pre>
<text>.REGEX_MATCH (<regular expression>)	<p>Returns TRUE if the target matches a &lt;regular expression&gt; argument of up to 1024 characters.</p> <p>The regular expression must be of the following format:</p> <pre>re&lt;delimiter&gt;regular expression&lt; delimiter&gt;</pre> <p>Both delimiters must be the same. Additionally, the regular expression must conform to the Perl-compatible (PCRE) regular expression library syntax. For more information, see <a href="http://www.pcre.org/pcr.txt">http://www.pcre.org/pcr.txt</a>. In particular, see the pcrepattern man page. However, the following:</p> <ul style="list-style-type: none"> <li>Back-references are not allowed.</li> <li>Recursive regular expressions are not recommended.</li> <li>The dot metacharacter also matches the newline character.</li> <li>The Unicode character set is not supported.</li> <li>SET_TEXT_MODE(IGNORECASE) overrides the (?i) internal option for the regular expression.</li> </ul> <p>The following are examples:</p> <pre>http.req.hostname.regex_match(re/[[:alpha:]]+(abc){2,3}/) http.req.url.set_text_mode(urlencoded).regex_match(re#(a*b</pre> <p>The following example matches ab and aB:</p> <pre>http.req.url.regex_match(re/a(?i)b/)</pre>

The following example matches ab, aB, Ab and AB:

```
http.req.url.set_text_mode(ignorecase).regex_match(re/ab/)
```

The following example performs a case-insensitive, multiline match in which the character also matches a newline character:

```
http.req.body.regex_match(re/(?ixm) (^ab (.*) cd$) /)
```

## Configuring Classic Policies and Expressions

Some NetScaler features use classic policies and classic expressions. As with default syntax policies, classic policies can be either global or specific to a virtual server. However, to a certain extent, the configuration method and bind points for classic policies are different from those of default syntax policies. As with default syntax expressions, you can configure named expressions and use a named expression in multiple classic policies.

The following table summarizes NetScaler features that can be configured by using classic policies.

Table 1. Policy Type and Bind Points for Policies in Features That Use Classic Policies

Feature	Virtual Servers	Supported Policies	Policy Bind Points	How You Use the Policies
System features, Authentication	None	Authentication policies	Global	For the Authentication feature, policies contain authentication schemes for different authentication methods. For example, you can configure LDAP and certificate-based authentication schemes.
SSL	None	SSL policies	<ul style="list-style-type: none"> <li>Global</li> <li>Load Balancing virtual server</li> </ul>	<p>To determine when to apply an encryption function and add certificate information to clear text.</p> <p>To provide end-to-end security. After a message is decrypted, the SSL feature re-encrypts clear text and uses SSL to communicate with back-end Web servers.</p>
Content Switching  (Can use either classic or default syntax policies, but not both)	Content Switching virtual server	Content Switching policies	<ul style="list-style-type: none"> <li>Content Switching virtual server</li> <li>Cache Redirection virtual server</li> </ul>	<p>To determine what server or group of servers is responsible for serving responses, based on characteristics of an incoming request.</p> <p>Request characteristics include device type, language, cookies, HTTP method, content type and associated cache server.</p>
Compression	None	HTTP Compression policies	<ul style="list-style-type: none"> <li>Global</li> <li>Content Switching virtual server</li> <li>Load Balancing virtual server</li> <li>SSL Offload virtual server</li> <li>Service</li> </ul>	To determine what type of HTTP traffic is compressed.
	None	Content Filtering policies	<ul style="list-style-type: none"> <li>Global</li> </ul>	To configure the behavior of the filter function.

Protection features, Filter			<ul style="list-style-type: none"> <li>Content Switching virtual server</li> <li>Load Balancing virtual server</li> <li>SSL Offload virtual server</li> <li>Service</li> </ul>	
Protection features, SureConnect	None	SureConnect policies	<ul style="list-style-type: none"> <li>Load Balancing virtual server</li> <li>SSL Offload virtual server</li> <li>Service</li> </ul>	To configure the behavior of the SureConnect function.
Protection features, Priority Queuing	None	Priority Queuing policies	<ul style="list-style-type: none"> <li>Load Balancing virtual server</li> <li>SSL Offload virtual server</li> </ul>	To configure the behavior of the Priority Queuing function.
HTML Injection	None	HTML Injection Policies	<ul style="list-style-type: none"> <li>Global</li> <li>Load Balancing virtual server</li> <li>Content Switching virtual server</li> <li>SSL Offload virtual server</li> </ul>	To enable the NetScaler to insert text or scripts into an HTTP response that it serves to a client.
AAA - Traffic Management	None	Authentication, Authorization, Auditing, and Session policies	<ul style="list-style-type: none"> <li>Authentication virtual server (authentication, session, and auditing policies)</li> <li>Load Balancing or Content Switching virtual server (authorization and auditing policies)</li> <li>Global (session and audit policies)</li> <li>AAA group or user (session, auditing, and authorization policies)</li> </ul>	To configure rules for user access to specific sessions and auditing of user access.

Cache Redirection	Cache Redirection virtual server	Cache Redirection policies Map policies	Cache Redirection virtual server	To determine whether HTTP responses are served from a cache or an origin server.
Application firewall	None	Application firewall policies	Global	To identify characteristics of traffic and data that should or should not be admitted through the firewall.
NetScaler Gateway	VPN server	Pre-Authentication policies	<ul style="list-style-type: none"> <li>AAA Global</li> <li>VPN vserver</li> </ul>	To determine how the NetScaler Gateway performs authentication, authorization, auditing, and other functions, and to define rewrite rules for general Web access using the NetScaler Gateway.
		Authentication policies	<ul style="list-style-type: none"> <li>System Global</li> <li>AAA Global</li> <li>VPN vserver</li> </ul>	
		Auditing policies	<ul style="list-style-type: none"> <li>User</li> <li>User group</li> <li>VPN vserver</li> </ul>	
		Session policies	<ul style="list-style-type: none"> <li>VPN Global</li> <li>User</li> <li>User Group</li> <li>VPN vserver</li> </ul>	
		Authorization policies	<ul style="list-style-type: none"> <li>User</li> <li>User Group</li> </ul>	
		Traffic policies	<ul style="list-style-type: none"> <li>VPN Global</li> <li>User</li> <li>User Group</li> <li>VPN vserver</li> </ul>	
		TCP Compression policies	VPN Global	

## Configuring a Classic Policy

You can configure classic policies and classic expressions by using either the configuration utility or the command-line interface. A policy rule cannot exceed 1,499 characters. When configuring the policy rule, you can use named classic expressions. For more information about named expressions, see ["Creating Named Classic Expressions."](#) After configuring the policy, you bind it either globally or to a virtual server.

Note that there are small variations in the policy configuration methods for various NetScaler features.

Note: You can embed a classic expression in a default syntax expression by using the syntax `SYS.EVAL_CLASSIC_EXPR(classic_expression)`, specifying the *classic\_expression* as the argument.

## To create a classic policy by using the command line interface

At the command prompt, type the following commands to set the parameters and verify the configuration:

- o add cmp policy <name> -rule <expression> -action <action>
- o show cmp policy [<policyName>]

### Example

The following commands first create a compression action and then create a compression policy that applies the action:

```
> add cmp action cmp-act-compress compress
Done
> show cmp action cmp-act-compress
1)      Name: cmp-act-compress  Compression Type: compress
Done
> add cmp pol cmp-pol-compress -rule ExpCheckIp -resAction cmp-act-compress
Done
> show cmp pol cmp-pol-compress
1)      Name: cmp-pol-compress  Rule: ExpCheckIp
        Response action: cmp-act-compress      Hits: 0
Done
>
```

## To create a policy with classic expressions by using the configuration utility

1. In the navigation pane, expand the feature for which you want to configure a policy and, depending on the feature, do the following:
  - o For Content Switching, Cache Redirection, and the application firewall, click Policies.
  - o For SSL, click Policies, and then in the details pane, click the Policies tab.
  - o For System Authentication, click Authentication, and then in the details pane, click the Policies tab.
  - o For Filter, SureConnect, and Priority Queuing, expand Protection Features, select the desired function, and then in the details pane, click the Policies tab.
  - o For the NetScaler Gateway, expand NetScaler Gateway, expand Policies, select the desired function, and then in the details pane, click the Policies tab.
2. For most features, click the Add button.
3. In the Create <feature name> Policy dialog box, in the Name\* text box, enter a name for the policy.

Note: Note: You must begin a policy name with a letter or underscore. A policy name can consist of 1 to 31 characters, including letters, numbers, hyphen (-), period (.), pound sign (#), space ( ), and underscore (\_).

4. For most features, you associate an action or a profile. For example, you may be required to select an action, or, in the case of an NetScaler Gateway or application firewall policy, you select a profile to associate with the policy. A profile is a set of configuration options that operate as a set of actions that are applied when the data being analyzed matches the policy rule.
5. Create an expression that describes the type of data that you want this policy to match.

Depending on the type of policy you want to create, you can choose a predefined expression, or you can create a new expression. For instructions on how to create an expression for most types of classic policies, see ["Configuring a Classic Expression."](#)

Named expressions are predefined expressions that you can reference by name in a policy rule. For more information about named expressions, see ["Creating Named Classic Expressions."](#) For a list of all the default named expressions and a definition of each, see ["Expressions Reference."](#)

6. Click Create to create your new policy.
7. Click Close to return to the Policies screen for the type of policy you were creating.



## Configuring a Classic Expression

Classic expressions consist of the following expression elements, listed in hierarchical order:

- **Flow Type.** Specifies whether the connection is incoming or outgoing. The flow type is REQ for incoming connections and RES for outgoing connections.
- **Protocol.** Specifies the protocol, the choices for which are HTTP, SSL, TCP, and IP.
- **Qualifier.** The protocol attribute, which depends on the selected protocol.
- **Operator.** The type of test you want to perform on the connection data. Your choice of operator depends upon the connection information you are testing. If the connection information you are testing is text, you use text operators. If it is a number, you use standard numeric operators.
- **Value.** The string or number against which the connection data element—defined by the flow type, protocol, and qualifier—is tested. The value can be either a literal or an expression. The literal or expression must match the data type of the connection data element.

In a policy, classic expressions can be combined to create more complex expressions using Boolean and comparative operators.

Expression elements are parsed from left to right. The leftmost element is either REQ or RES and designates a request or a response, respectively. Successive terms define a specific connection type and a specific attribute for that connection type. Each term is separated from any preceding or following term by a period. Arguments appear in parentheses and follow the expression element to which they are passed.

The following classic expression fragment returns the client source IP for an incoming connection.

```
REQ.IP.SOURCEIP
```

The example identifies an IP address in a request. The expression element SOURCEIP designates the source IP address. This expression fragment may not be useful by itself. You can use an additional expression element, an operator, to determine whether the returned value meets specific criteria. The following expression tests whether the client IP is in the subnet 200.0.0.0/8 and returns a Boolean TRUE or FALSE:

```
REQ.IP.SOURCEIP == 200.0.0.0 -netmask 255.0.0.0
```

## To create a classic policy expression by using the command line interface

At the command prompt, type the following commands to set the parameters and verify the configuration:

- set appfw policy <name> -rule <expression> -action <action>
- show appfw policy <name>

### Example

```
> set appfw policy GenericApplicationSSL_ 'HTTP.REQ.METHOD.EQ("get")' APPFW_DROP
Done
> show appfw policy GenericApplicationSSL_
  Name: GenericApplicationSSL_      Rule: HTTP.REQ.METHOD.EQ("get")
  Profile: APPFW_DROP              Hits: 0
  Undef Hits: 0
  Policy is bound to following entities
  1) REQ VSERVER app_u_GenericApplicationSSLPortalPages  PRIORITY : 100
Done
```

## To add an expression for a classic policy by using the configuration utility

This procedure documents the Add Expression dialog box. Depending on the feature for which you are configuring a policy, the route by which you arrive at this dialog box may be different.

1. Perform steps 1-4 in "To create a policy with classic expressions by using the configuration utility."
2. In the Add Expression dialog box, in Expression Type, click the type of expression you want to create.
3. Under Flow Type, click the down arrow and choose a flow type.

The flow type is typically REQ or RES. The REQ option specifies that the policy applies to all incoming connections or requests. The RES option applies the policy to all outgoing connections or responses.

For Application Firewall policies, you should leave the expression type set to General Expression, and the flow type set to REQ. The Application Firewall treats each request and response as a single paired entity, so all Application Firewall policies begin with REQ.

4. Under Protocol, click the down arrow and choose the protocol you want for your policy expression. Your choices are:
  - HTTP. Evaluates HTTP requests that are sent to a Web server. For classic expressions, HTTP includes HTTPS requests.
  - SSL. Evaluates SSL data associated with the current connection.
  - TCP. Evaluates the TCP data associated with the current connection.
  - IP. Evaluates the IP addresses associated with the current connection.
5. Under Qualifier, click the down arrow and choose a qualifier for your policy.

The qualifier defines the type of data to be evaluated. The list of qualifiers that appears depends on which protocol you selected in step 4.

The following list describes the qualifier choices for the HTTP protocol. For a complete list of protocols and qualifiers, see "[Classic Expressions](#)."

The following choices appear for the HTTP protocol:

- METHOD. Filters HTTP requests that use a particular HTTP method.
  - URL. Filters HTTP requests for a specific Web page.
  - URLQUERY. Filters HTTP requests that contain a particular query string.
  - VERSION. Filters HTTP requests on the basis of the specified HTTP protocol version.
  - HEADER. Filters on the basis of a particular HTTP header.
  - URLLLEN. Filters on the basis of the length of the URL.
  - URLQUERY. Filters on the basis of the query portion of the URL.
  - URLQUERYLEN. Filters on the basis of the length of the query portion of the URL only.
6. Under Operator, click the down arrow and choose the operator for your policy expression. For a complete list of choices see the "Operators" table in "[Classic Expressions](#)." Some common operators are:

Operator	Description
==	Matches the specified value exactly or is exactly equal to the specified value.
!=	Does not match the specified value.
>	Is greater than the specified value.
<	Is less than the specified value.
>=	Is greater than or equal to the specified value.
<=	Is less than or equal to the specified value.
<b>CONTAINS</b>	Contains the specified value.
<b>CONTENTS</b>	Returns the contents of the designated header, URL, or URL query.
<b>EXISTS</b>	The specified header or query exists.
<b>NOTCONTAINS</b>	Does not contain the specified value.
<b>NOTEXISTS</b>	The specified header or query does not exist.

7. If a Value text box appears, type a string or numeric value, as appropriate. For example, chose REQ as the Flow Type , HTTP as the Protocol, and HEADER as the qualifier, and then type the value of the header string in the Value field and the header type for which you want to match the string in the Header Name text box.
8. Click OK.

9. To create a compound expression, click Add. Note that the type of compounding that is done depends on the following choices in the Create Policy dialog box:
- **Match Any Expression.** The expressions are in a logical OR relationship.
  - **Match All Expressions.** The expressions are in a logical AND relationship.
  - **Tabular Expressions.** Click the AND, OR, and parentheses buttons to control evaluation.
  - **Advanced Free-Form.** Enter the expressions components directly into the Expression field, and click the AND, OR, and parentheses buttons to control evaluation.

## Binding a Classic Policy

Depending on the policy type, you can bind a classic policy either globally or to a virtual server. Policy bind points are described in the table, "Policy Type and Bind Points for Policies in Features That Use Classic Policies."

Note: You can bind a classic policy to multiple bind points.

### To bind a classic policy globally by using the command line interface

At the command prompt, type the following commands to set the parameters and verify the configuration:

- `bind cmp global <policyName> [-priority <positive_integer>]`
- `show cmp global`

#### Example

```
> bind cmp global cmp-pol-compress -priority 2
Done
> show cmp global
1)      Policy Name: cmp-pol-compress      Priority: 2
2)      Policy Name: ns_nocmp_xml_ie      Priority: 8700
3)      Policy Name: ns_nocmp_mozilla_47      Priority: 8800
4)      Policy Name: ns_cmp_mscss      Priority: 8900
5)      Policy Name: ns_cmp_msapp      Priority: 9000
6)      Policy Name: ns_cmp_content_type      Priority: 10000
Done
>
```

### To bind a classic policy to a virtual server by using the command line interface

At the command prompt, type the following commands to set the parameters and verify the configuration:

- `bind lb vserver <name> [<targetVserver>] [-policyName <string>] [-priority <positive_integer>]`
- `show lb vserver<name>`

#### Example

```
> bind lb vserver lbtemp -policyName cmp-pol-compress -priority 1
Done
> show lb vserver lbtemp
  lbtemp (10.102.29.101:80) - HTTP      Type: ADDRESS
  State: UP
  Last state change was at Tue Oct 27 06:40:38 2009 (+557 ms)
  Time since last state change: 0 days, 02:00:40.330
  Effective State: UP
  Client Idle Timeout: 180 sec
  Down state flush: ENABLED
  Disable Primary Vserver On Down : DISABLED
  Port Rewrite : DISABLED
  No. of Bound Services : 1 (Total)      1 (Active)
  Configured Method: LEASTCONNECTION
  Current Method: Round Robin, Reason: Bound service's state changed to UP
  Group: vserver-grp
  Mode: IP
  Persistence: COOKIEINSERT (version 0)      Persistence Backup: SOURCEIP      Persistence M
  Persistence Timeout: 2 min      Backup Persistence Timeout: 2 min
  Vserver IP and Port insertion: OFF
  Push: DISABLED      Push VServer:
  Push Multi Clients: NO
  Push Label Rule: none
1) http-one (10.102.29.252: 80) - HTTP State: UP      Weight: 1
  Persistence Cookie Value : NSC_wtfswfs-hsq=ffffffff096e03ed45525d5f4f58455e445a4a423
1)      Policy : cmp-pol-compress Priority:1
Done
>
```

### To bind a classic policy globally by using the configuration utility

Note: This procedure documents the Global Bindings dialog box. Depending on the feature for which you want to globally bind a policy, the route by which you arrive at this dialog box may be different.

1. In the navigation pane, expand the feature for which you want to globally bind a classic policy, and then locate the policy that you want to bind globally.

Note: You cannot globally bind policies for Content Switching, Cache Redirection, SureConnect, Priority Queuing, or NetScaler Gateway Authorization.

2. In the details pane, click Global Bindings.
3. In the Bind/Unbind <feature name> Policy(s) to Global dialog box, click Insert Policy.
4. In the Policy Name column, click the name of an existing policy that you want to globally bind, or click New Policy to open the Create <feature name> Policy dialog box.
5. After you have selected the policy or created a new policy, in the Priority column, type the priority value.

The lower the number, the sooner this policy is applied relative to other policies. For example, a policy assigned a priority of 10 is applied before a policy with a priority of 100. You can use the same priority for different policies. All features that use classic policies implement only the first policy that a connection matches, so policy priority is important for getting the results you intend.

As a best practice, leave room to add policies by setting priorities with intervals of 50 (or 100) between each policy.

6. Click OK.

## To bind a classic policy to a virtual server by using the configuration utility

1. In the navigation pane, expand the feature that contains the virtual server to which you want to bind a classic policy (for example, if you want to bind a classic policy to a content switching virtual server, expand Traffic Management > Content Switching), and then click Virtual Servers.
2. In the details pane, select the virtual server, and then click Open.
3. In the Configure <Feature> Virtual Server dialog box, on the Policies tab, click the feature icon for the type policy that you want, and then click Insert Policy.
4. In the Policy Name column, click the name of an existing policy that you want to bind to a virtual server, or click A to open the Create <feature name> Policy dialog box.
5. After you have selected the policy or created a new policy, in the Priority column, set the priority.

If you are binding a policy to a content switching virtual server, in the Target column, select a load balancing virtual server to which traffic that matches the policy should be sent.

6. Click OK.

## Viewing Classic Policies

You can view classic policies by using either the configuration utility or the command line. You can view details such as the policy's name, expression, and bindings.

### To view a classic policy and its binding information by using the command line interface

At the command prompt, type the following commands to view a classic policy and its binding information:

show <featureName> policy [policyName]

#### Example

```
> show appfw policy GenericApplicationSSL_
  Name: GenericApplicationSSL_      Rule: ns_only_get_adv
  Profile: GenericApplicationSSL_Profl  Hits: 0
  Undef Hits: 0
  Policy is bound to following entities
  1) REQ VSERVER app_u_GenericApplicationSSLPortalPages  PRIORITY : 100
Done
```

Note: If you omit the policy name, all policies are listed without the binding details.

### To view classic policies and policy bindings by using the configuration utility

1. In the navigation pane, expand the feature whose policies you want to view, (for example, if you want to view application firewall policies, expand Application Firewall), and then click Policies.
2. In the details pane, do one or more of the following:
  - o To view details for a specific policy, click the policy. Details appear in the Details area of the configuration pane.
  - o To view bindings for a specific policy, click the policy, and then click Show Bindings.
  - o To view global bindings, click the policy, and then click Global Bindings. Note that you cannot bind a Content Switching, Cache Redirection, SureConnect, Priority Queuing, or NetScaler Gateway Authorization policy globally.

## Creating Named Classic Expressions

A named classic expression is a classic expression that can be referenced through an assigned name. Often, you need to configure classic expressions that are large or complex and form a part of a larger compound expression. You might also configure classic expressions that you need to use frequently and in multiple compound expressions or classic policies. In these scenarios, you can create the classic expression you want, save it with a name of your choice, and then reference the expression from compound expressions or policies through its name. This saves configuration time and improves the readability of complex compound expressions. Additionally, any modifications to a named classic expression need to be made only once.

Some named expressions are built-in, and a subset of these are read-only. Built-in named expressions are divided into four categories: General, Anti-Virus, Personal Firewall, and Internet Security. General named expressions have a wide variety of uses. For example, from the General category, you can use the expressions `ns_true` and `ns_false` to specify a value of TRUE or FALSE, respectively, to be returned for all traffic. You can also identify data of a particular type (for example, HTM, DOC, or GIF files), determine whether caching headers are present, or determine whether the round trip time for packets between a client and the NetScaler is high (over 80 milliseconds).

Anti-Virus, Personal Firewall, and Internet Security named expressions test clients for the presence of a particular program and version and are used primarily in NetScaler Gateway policies.

For descriptions of the built-in named expressions, see "[Classic Expressions](#)."

Note: You cannot modify or delete built-in named expressions.

## To create a named classic expression by using the command line interface

At the command prompt, type the following commands to set the parameters and verify the configuration:

- add expression <name> <value> [-comment <string>] [-clientSecurityMessage <string>]
- show expression [<name> | -type CLASSIC]

### Example

```
> add expression classic_ne "REQ.HTTP.URL CONTAINS www.example1.com" -comment "Checking the
Done
> show expression classic_ne
1)      Name: classic_ne  Expr: REQ.HTTP.URL CONTAINS www.example1.com  Hits: 0 Type : CLASS
      Comment: "Checking the URL for www.example1.com"
Done
>
```

## To create a named classic expression by using the configuration utility

1. In the navigation pane, expand AppExpert, expand Expressions, and then click Classic Expressions.
2. In the details pane, click Add.

Note: Some of the built-in expressions in the Expressions list are read-only.

3. In the Create Policy Expression dialog box, specify values for the following parameters:

- Expression Name\*â€™name
- Client Security Messageâ€™clientSecurityMessage
- Commentsâ€™comment

\* A required parameter

4. To create the expression, do one of the following:
  - You can choose inputs to this expression from the Named Expressions drop-down list.
  - You can create a new expression, as described in "[To add an expression for a classic policy by using the configuration utility](#)."
5. When you are done, click Close. Verify that your new expression was created by scrolling to the bottom of the Classic Expressions list to view it.

## Expressions Reference-Default Syntax Expressions

The following table is a listing of default syntax expression prefixes, with cross-references to descriptions of these prefixes and the operators that you can specify for them. Note that some prefixes can work with multiple types of operators. For example, a cookie can be parsed by using operators for text or operators for HTTP headers.

You can use any element in the following tables as a complete expression on its own, or you can use various operators to combine these expression elements with others to form more complex expressions.

Note: The Description column in the following table contains cross-references to additional information about prefix usage and applicable operators for the prefix.

Expression Prefix	Links to Relevant Information, with Applicable Notes and Operator Descriptions
<code>CLIENT.ETHER</code>	<a href="#">"Prefixes for MAC Addresses."</a> <a href="#">"Operations for MAC Addresses."</a>
<code>CLIENT.ETHER.[DSTMAC   SRCMAC]</code>	<a href="#">"Prefixes for MAC Addresses."</a> <a href="#">"Operations for MAC Addresses."</a>
<code>CLIENT.INTERFACE</code>	Designates an expression that refers to the ID of the network interface through which the current packet entered the Application Switch. See the other <code>CLIENT.INTERFACE</code> prefix descriptions in this table.
<code>CLIENT.INTERFACE.ID</code>	Extracts the ID of the network interface that received the current packet of data. See the other <code>CLIENT.INTERFACE</code> prefix descriptions in this table.
<code>CLIENT.INTERFACE.ID.EQ("id")</code>	Returns Boolean TRUE if the interface's ID matches the ID that is passed as the argument. For example:  <code>CLIENT.INTERFACE.ID.EQ("1/1")</code>  <a href="#">See "Booleans in Compound Expressions."</a>
<code>CLIENT.INTERFACE.[RXTHROU GHPUT   RXTXTHROUGHPUT   TXTHROUGHPUT]</code>	<a href="#">"Expressions for Numeric Client and Server Data."</a>  <a href="#">"Compound Operations for Numbers."</a>
<code>CLIENT.IP</code>	Operates on the IP protocol data associated with the current packet. See the other <code>CLIENT.IP</code> prefixes in this table.
<code>CLIENT.IP.DST</code>	<a href="#">"Prefixes for IPV4 Addresses and IP Subnets."</a>  <a href="#">"Operations for IPV4 Addresses."</a>  <a href="#">"Compound Operations for Numbers."</a>



<code>CLIENT.IP.SRC</code>	<p>"Prefixes for IPV4 Addresses and IP Subnets."</p> <p>"Operations for IPV4 Addresses."</p> <p>"Compound Operations for Numbers."</p>
<code>CLIENT.IPV6</code>	Operates on IPv6 protocol data. See the other <code>CLIENT.IPV6</code> prefixes in this table.
<code>CLIENT.IPV6.DST</code>	<p>"Expression Prefixes for IPv6 Addresses."</p> <p>"Operations for IPV6 Prefixes."</p>
<code>CLIENT.IPV6.SRC</code>	<p>"Expression Prefixes for IPv6 Addresses."</p> <p>"Operations for IPV6 Prefixes."</p>
<code>CLIENT.SSL</code>	Operates on the SSL protocol data for the current packet. See the other <code>CLIENT.SSL</code> prefixes in this table.
<code>CLIENT.SSL.CIPHER_BITS</code>	<p>"Prefixes for Numeric Data in SSL Certificates."</p> <p>"Compound Operations for Numbers."</p>
<code>CLIENT.SSL.CIPHER_EXPORTABLE</code>	<p>"Prefixes for Text-Based SSL and Certificate Data."</p> <p>"Booleans in Compound Expressions."</p>
<code>CLIENT.SSL.CLIENT_CERT</code>	<p>"Expressions for SSL Certificates."</p> <p>"Expressions for SSL Certificate Dates."</p>
<code>CLIENT.SSL.IS_SSL</code>	<p>"Prefixes for Text-Based SSL and Certificate Data."</p> <p>"Booleans in Compound Expressions."</p>
<code>CLIENT.SSL.VERSION</code>	<p>"Prefixes for Numeric Data in SSL Certificates."</p> <p>"Compound Operations for Numbers."</p>
<code>CLIENT.TCP</code>	Operates on TCP protocol data. See the other <code>CLIENT.TCP</code> prefixes in this table.
<code>CLIENT.TCP.[DSTPORT   MSS   SRCPORT]</code>	<p>"Expressions for TCP, UDP, and VLAN Data."</p> <p>"Compound Operations for Numbers."</p>
<code>CLIENT.TCP.PAYLOAD( integer )</code>	"Expressions for TCP, UDP, and VLAN Data."

	"Default Syntax Expressions: Evaluating Text."
CLIENT.UDP	Operates on the UDP protocol data associated with the current packet. See the other CLIENT.UDP prefixes in this table.
CLIENT.UDP.DNS.DOMAIN	"Expressions for TCP, UDP, and VLAN Data." "Default Syntax Expressions: Evaluating Text."
CLIENT.UDP.DNS.DOMAIN.EQ( "hostname" )	"Expressions for TCP, UDP, and VLAN Data." "Booleans in Compound Expressions."
CLIENT.UDP.DNS. [IS_AAAAREC   IS_ANYREC   IS_AREC   IS_CNAMEREC   IS_MXREC   IS_NSREC   IS_PTRREC   IS_SOAREC   IS_SRVREC]	"Expressions for TCP, UDP, and VLAN Data." "Booleans in Compound Expressions."
CLIENT.UDP.[DSTPORT   SRCPORT]	"Expressions for TCP, UDP, and VLAN Data." "Compound Operations for Numbers."
CLIENT.VLAN	Operates on the VLAN through which the current packet entered the NetScaler. See the other CLIENT.VLAN prefixes in this table.
CLIENT.VLAN.ID	"Expressions for TCP, UDP, and VLAN Data." "Compound Operations for Numbers."
HTTP.REQ	Operates on HTTP requests. See the other HTTP.REQ prefixes in this table.
HTTP.REQ.BODY(integer)	"Expression Prefixes for Text in HTTP Requests and Responses." "Basic Operations on Text."
HTTP.REQ.CACHE_CONTROL	"Prefixes for Cache-Control Headers." "Operations for Cache-Control Headers."
HTTP.REQ.CONTENT_LENGTH	"Expressions for Numeric HTTP Payload Data Other Than Dates." "Compound Operations for Numbers."
HTTP.REQ.COOKIE	"Prefixes for HTTP Headers." "Operations for HTTP Headers."

	"Default Syntax Expressions: Evaluating Text."
HTTP.REQ.DATE	<p>"Format of Dates and Times in an Expression."</p> <p>"Expressions for HTTP Request and Response Dates."</p> <p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.REQ.HEADER( "header_name" )	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.REQ.FULL_HEADER( "header_name" )	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.REQ.HOSTNAME	"Expression Prefixes for Text in HTTP Requests and Responses."
HTTP.REQ.HOSTNAME.[DOMAIN   Server]	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Basic Operations on Text."</p>
HTTP.REQ.HOSTNAME.EQ( "hostname" )	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Booleans in Compound Expressions."</p> <p>"â€œBasic Operations on Expression Prefixesâ€•."</p>
HTTP.REQ.HOSTNAME.PORT	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Compound Operations for Numbers."</p>
HTTP.REQ.IS_VALID	Returns TRUE if the HTTP request is properly formed. See "Booleans in Compound Expressions."
HTTP.REQ.METHOD	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Basic Operations on Text."</p> <p>"Complex Operations on Text."</p>
HTTP.REQ.TRACKING	Returns the HTTP body tracking mechanism. See the descriptions of

	other HTTP.REQ.TRACKING prefixes in this table.
HTTP.REQ.TRACKING.EQ( <i>tracking_mechanism</i> )	Returns TRUE or FALSE. See "Booleans in Compound Expressions."
HTTP.REQ.URL	Obtains the HTTP URL object from the request and sets the text mode to URLENCODED by default.  See "Expression Prefixes for Text in HTTP Requests and Responses."
HTTP.REQ.URL.[CVPN_ENCODE   HOSTNAME   HOSTNAME.DOMAIN   SERVER   PATH   PATH_AND_QUERY   PROTOCOL   QUERY   SUFFIX   VERSION]	"Expression Prefixes for Text in HTTP Requests and Responses."  "Basic Operations on Text."  "Complex Operations on Text."
HTTP.REQ.URL.HOSTNAME.EQ( <i>hostname</i> )	"Expression Prefixes for Text in HTTP Requests and Responses."  "Booleans in Compound Expressions."
HTTP.REQ.URL.HOSTNAME.PORT	"Expression Prefixes for Text in HTTP Requests and Responses."  "Compound Operations for Numbers."
HTTP.REQ.URL.PATH.IGNORE_ EMPTY_ELEMENTS	Ignores spaces in the data. See the table "HTTP Expression Prefixes that Return Text."
HTTP.REQ.URL.QUERY.IGNORE_ EMPTY_ELEMENTS	Ignores spaces in the data. See the table "HTTP Expression Prefixes that Return Text."
HTTP.REQ.USER.IS_MEMBER_OF	"HTTP Expression Prefixes that Return Text."
HTTP.REQ.USER.NAME	"HTTP Expression Prefixes that Return Text."
HTTP.REQ.VERSION	"Expression Prefixes for Text in HTTP Requests and Responses."
HTTP.REQ.VERSION.[MAJOR   MINOR]	Operates on the major or minor HTTP version string. See "Expression Prefixes for Text in HTTP Requests and Responses" and "Compound Operations for Numbers."
HTTP.RES	Operates on HTTP responses.
HTTP.RES.BODY( <i>integer</i> )	"Expression Prefixes for Text in HTTP Requests and Responses."

	<p>"Basic Operations on Text."</p> <p>"Complex Operations on Text."</p>
HTTP.RES.CACHE_CONTROL	<p>"Prefixes for Cache-Control Headers."</p> <p>"Operations for Cache-Control Headers."</p>
HTTP.RES.CONTENT_LENGTH	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Operations for HTTP Headers."</p> <p>"Compound Operations for Numbers."</p>
HTTP.RES.DATE	<p>"Format of Dates and Times in an Expression."</p> <p>"Expressions for HTTP Request and Response Dates."</p> <p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Compound Operations for Numbers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.RES.HEADER( "header_name" )	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.REQ.FULL_HEADER( "header_name" )	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.REQ.TXID	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>
HTTP.RES.IS_VALID	<p>Returns TRUE if the HTTP response is properly formed. See "Booleans in Compound Expressions."</p>
HTTP.RES.SET_COOKIE	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE.COOKIE( "name" )	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
	<p>"Prefixes for HTTP Headers."</p>

HTTP.RES.SET_COOKIE.COOKIE.[DOMAIN   PATH   PORT ]	<p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE.COOKIE.EXPIRES	<p>Obtains the Expires field of the cookie as a date string. The value of the Expires attribute can be operated upon as a time object. If multiple Expires fields are present, this expression operates on the first one. If the Expires attribute is absent, a string of length zero is returned.</p> <p>Also see:</p> <p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p>
HTTP.RES.SET_COOKIE.COOKIE.PATH.IGNORE_EMPTY_ELEMENTS	<p>Ignores spaces in the data. For an example, see the table "Expression Prefixes for Text in HTTP Requests and Responses."</p>
HTTP.RES.SET_COOKIE.COOKIE.PORT.IGNORE_EMPTY_ELEMENTS	<p>Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."</p>
HTTP.RES.SET_COOKIE.COOKIE.VERSION	<p>"Prefixes for HTTP Headers."</p> <p>"Compound Operations for Numbers."</p>
HTTP.RES.SET_COOKIE.COOKIE("name",integer)[.PORT   PATH   DOMAIN   VERSION   EXPIRES]	<p>"Prefixes for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE.COOKIE.EXPIRES	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p>
HTTP.RES.SET_COOKIE.EXISTS("name")	<p>"Prefixes for HTTP Headers."</p> <p>"Booleans in Compound Expressions."</p>
HTTP.RES.SET_COOKIE2	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>

HTTP.RES.SET_COOKIE2.COOKIE("name")	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE2.COOKIE.[DOMAIN   PATH   PORT ]	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE2.COOKIE.EXPIRES	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p>
HTTP.RES.SET_COOKIE2.COOKIE.PATH.IGNORE_EMPTY_ELEMENTS	<p>Ignores spaces in the data. For an example, see the table <a href="#">HTTP Expression Prefixes that Return Text.</a></p>
HTTP.RES.SET_COOKIE2.COOKIE.PORT.IGNORE_EMPTY_ELEMENTS	<p>Ignores spaces in the data. For an example, see the table <a href="#">"HTTP Expression Prefixes that Return Text."</a></p> <p>See also <a href="#">"Default Syntax Expressions: Evaluating Text"</a> and <a href="#">"Compound Operations for Numbers."</a></p>
HTTP.RES.SET_COOKIE2.COOKIE("name",integer)[.PORT   PATH   DOMAIN   VERSION   EXPIRES]	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE2.COOKIE.DOMAIN	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p>
HTTP.RES.SET_COOKIE2.COOKIE.EXPIRES	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p>
HTTP.RES.SET_COOKIE2.COOKIE.VERSION	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>

	<p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p>
<code>HTTP.RES.SET_COOKIE2.EXISTS( "name" )</code>	<p>"Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p> <p>"Booleans in Compound Expressions."</p>
<code>HTTP.RES.STATUS</code>	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p> <p>"Compound Operations for Numbers."</p>
<code>HTTP.RES.STATUS_MSG</code>	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p>
<code>HTTP.RES.TRACKING</code>	<p>Returns the HTTP body tracking mechanism. See the descriptions of other <code>HTTP.REQ.TRACKING</code> prefixes in this table.</p>
<code>HTTP.RES.TRACKING.EQ( "tracking_method" )</code>	<p>Returns TRUE or FALSE. See "Booleans in Compound Expressions."</p>
<code>HTTP.RES.TXID</code>	<p>" Prefixes for HTTP Headers."</p> <p>"Operations for HTTP Headers."</p>
<code>HTTP.RES.VERSION</code>	<p>"Expression Prefixes for Text in HTTP Requests and Responses."</p>
<code>HTTP.RES.VERSION.[MAJOR   MINOR]</code>	<p>Operates on the major or minor HTTP version string. See "Expression Prefixes for Text in HTTP Requests and Responses" and "Compound Operations for Numbers."</p>
<code>SERVER</code>	<p>Designates an expression that refers to the server. This is the starting point for access into parameters such as Ether and SSL. See the other <code>SERVER</code> prefixes in this table.</p>
<code>SERVER.ETHER</code>	<p>Operates on the ethernet protocol data associated with the current packet. See the other <code>SERVER</code> prefixes in this table.</p>
<code>SERVER.ETHER.DSTMAC</code>	<p>"Prefixes for MAC Addresses."</p> <p>"Prefixes for MAC Addresses."</p>
<code>SERVER.INTERFACE</code>	<p>Designates an expression that refers to the ID of the network interface that</p>



	received the current packet of data. See the other <code>SERVER.INTERFACE</code> prefixes in this table.
<code>SERVER.INTERFACE.ID.EQ("id")</code>	<p>Returns Boolean TRUE if the interface's ID matches the ID that is passed as the argument. For example:</p> <pre>SERVER.INTERFACE.ID.EQ("LA/1")</pre> <p>See "Booleans in Compound Expressions."</p>
<code>SERVER.INTERFACE.[RXTHROUGHPUT   RXTXTHROUGHPUT   TXTHROUGHPUT]</code>	<p>"Expressions for Numeric Client and Server Data."</p> <p>"Compound Operations for Numbers."</p>
<code>SERVER.IP</code>	Operates on the IP protocol data associated with the current packet. See the other <code>SERVER.IP</code> prefixes in this table.
<code>SERVER.IP.[DST   SRC]</code>	<p>"Prefixes for IPV4 Addresses and IP Subnets."</p> <p>"Operations for IPV4 Addresses."</p> <p>"Compound Operations for Numbers."</p>
<code>SERVER.IPV6</code>	Operates on IPv6 protocol data. See the other <code>SERVER.IPV6</code> prefixes in this table.
<code>SERVER.IPV6.DST</code>	<p>"Expression Prefixes for IPv6 Addresses."</p> <p>"Operations for IPv6 Prefixes."</p>
<code>SERVER.IPV6.SRC</code>	<p>"Expression Prefixes for IPv6 Addresses."</p> <p>"Operations for IPv6 Prefixes."</p>
<code>SERVER.TCP</code>	Operates on TCP protocol data. See the other <code>CLIENT.TCP</code> prefixes in this table.
<code>SERVER.TCP.[DSTPORT   MSS   SRCPORT]</code>	<p>"Expressions for TCP, UDP, and VLAN Data."</p> <p>"Compound Operations for Numbers."</p>
<code>SERVER.VLAN</code>	Operates on the VLAN through which the current packet entered the NetScaler. See the other <code>SERVER.VLAN</code> prefixes in this table.
<code>SERVER.VLAN.ID</code>	"Expressions for TCP, UDP, and VLAN Data."

	"Compound Operations for Numbers."
SYS	Designates an expression that refers to the NetScaler itself, not to the client or server.. See the other SYS prefixes in this table.
SYS.EVAL_CLASSIC_EXPR( <i>classic_expression</i> )	"Classic Expressions in Default Syntax Expressions."  "Booleans in Compound Expressions."
SYS.HTTP_CALLOUT( <i>http_callout</i> )	"HTTP Callouts."
SYS.CHECK_LIMIT	"Rate Limiting."
SYS.TIME	"Expressions for the NetScaler System Time."  "Compound Operations for Numbers."
SYS.TIME.[BETWEEN( <i>time1</i> , <i>time2</i> )   EQ( <i>time</i> )   GE( <i>time</i> )   GT( <i>time</i> )   LE( <i>time</i> )   LT( <i>time</i> )   WITHIN( <i>time1</i> , <i>time2</i> )]	"Expressions for the NetScaler System Time."  "Booleans in Compound Expressions."  "Compound Operations for Numbers."
SYS.TIME.[DAY   HOURS   MINUTES   MONTH   RELATIVE_BOOT   RELATIVE_NOW SECONDS   WEEKDAY   YEAR]	"Expressions for the NetScaler System Time."  "Compound Operations for Numbers."
SYS.RANDOM	Returns a random number between 0 and 1, inclusive of 0 but exclusive of 1.
VPN.BASEURL.[CVPN_DECODE   CVPN_ENCODE   HOSTNAME   HOSTNAME.DOMAIN   HOSTNAME.SERVER   PATH   PATH_AND_QUERY   PROTOCOL   QUERY   SUFFIX]	"Expression Prefixes for VPNs and Clientless VPNs."
VPN.BASEURL.HOSTNAME.EQ("hostname")	"Expression Prefixes for VPNs and Clientless VPNs."  "Booleans in Compound Expressions."
VPN.BASEURL.HOSTNAME.PORT	"Expression Prefixes for VPNs and Clientless VPNs."  "Compound Operations for Numbers."
VPN.BASEURL.PATH.IGNORE_EMPTY_ELEMENTS	Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."
VPN.BASEURL.QUERY.IGNORE_EMPTY_ELEMENTS	Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."

VPN.CLIENTLESS_BASEURL	" Expression Prefixes for VPNs and Clientless VPNs."
VPN.CLIENTLESS_BASEURL.[CVPN_DECODE   CVPN_ENCODE   HOSTNAME   HOSTNAME.DOMAIN   HOSTNAME.SERVER   PATH   PATH_AND_QUERY   PROTOCOL   QUERY   SUFFIX]	"Expression Prefixes for VPNs and Clientless VPNs."
VPN.CLIENTLESS_BASEURL.HOSTNAME.EQ( "hostname" )	"Expression Prefixes for VPNs and Clientless VPNs."  "Booleans in Compound Expressions."
VPN.CLIENTLESS_BASEURL.HOSTNAME.PORT	"Expression Prefixes for VPNs and Clientless VPNs."  "Compound Operations for Numbers."
VPN.CLIENTLESS_BASEURL.PATH.IGNORE_EMPTY_ELEMENTS	Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."
VPN.CLIENTLESS_BASEURL.QUERY.IGNORE_EMPTY_ELEMENTS	Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."
VPN.CLIENTLESS_HOSTURL	" Expression Prefixes for VPNs and Clientless VPNs."
VPN.CLIENTLESS_HOSTURL.[CVPN_DECODE   CVPN_ENCODE   HOSTNAME   HOSTNAME.DOMAIN   HOSTNAME.SERVER   PATH   PATH_AND_QUERY   PROTOCOL   QUERY   SUFFIX]	"Expression Prefixes for VPNs and Clientless VPNs."
VPN.CLIENTLESS_HOSTURL.HOSTNAME.EQ( "hostname" )	"Expression Prefixes for VPNs and Clientless VPNs."  "Booleans in Compound Expressions."
VPN.CLIENTLESS_HOSTURL.HOSTNAME.PORT	"Expression Prefixes for VPNs and Clientless VPNs."  "Compound Operations for Numbers."
VPN.CLIENTLESS_HOSTURL.PATH.IGNORE_EMPTY_ELEMENTS	Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."
VPN.CLIENTLESS_HOSTURL.QUERY.IGNORE_EMPTY_ELEMENTS	Ignores spaces in the data. For an example, see the table "HTTP Expression Prefixes that Return Text."
VPN.HOST	" Expression Prefixes for VPNs and Clientless VPNs."
VPN.HOST.[DOMAIN   Server]	"Expression Prefixes for VPNs and Clientless VPNs."

VPN.HOST.EQ( "hostname" )	<p>"Expression Prefixes for VPNs and Clientless VPNs."</p> <p>"Booleans in Compound Expressions."</p>
VPN.HOST.PORT	<p>"Expression Prefixes for VPNs and Clientless VPNs."</p> <p>"Default Syntax Expressions: Evaluating Text."</p> <p>"Compound Operations for Numbers."</p>

## Expressions Reference-Classic Expressions

The subtopics listed in the table of contents on the left side of your screen contain tables listing the NetScaler classic expressions.

In the table of operators, the result type of each operator is shown at the beginning of the description. In the other tables, the level of each expression is shown at the beginning of the description. For named expressions, each expression is shown as a whole.

This document includes the following details:

- Operators
- General Expressions
- Client Security Expressions
- Network-Based Expressions
- Date/Time Expressions
- File System Expressions
- Built-In Named Expressions (General)
- Built-In Named Expressions (Anti-Virus)
- Built-In Named Expressions (Personal Firewall)
- Built-In Named Expressions (Client Security)

### Operators

Expression Element	Definition
==	Boolean.  Returns TRUE if the current expression equals the argument. For text operations, the items being compared must exactly match one another. For numeric operations, the items must evaluate to the same number.
!=	Boolean.  Returns TRUE if the current expression does not equal the argument. For text operations, the items being compared must not exactly match one another. For numeric operations, the items must not evaluate to the same number.
CONTAINS	Boolean.  Returns TRUE if the current expression contains the string that is designated in the argument.
NOTCONTAINS	Boolean.  Returns TRUE if the current expression does not contain the string that is designated in the argument.
CONTENTS	Text.  Returns the contents of the current expression.
EXISTS	Boolean.  Returns TRUE if the item designated by the current expression exists.
NOTEXISTS	Boolean.  Returns TRUE if the item designated by the current expression does not exist.

>	<p>Boolean.</p> <p>Returns TRUE if the current expression evaluates to a number that is greater than the argument.</p>
<	<p>Boolean.</p> <p>Returns TRUE if the current expression evaluates to a number that is less than the argument.</p>
>=	<p>Boolean.</p> <p>Returns TRUE if the current expression evaluates to a number that is greater than or equal to the argument.</p>
<=	<p>Boolean.</p> <p>Returns TRUE if the current expression evaluates to a number that is less than or equal to the argument.</p>

## General Expressions

Expression Element	Definition
REQ	<p>Flow Type.</p> <p>Operates on incoming (or request) packets.</p>
REQ.HTTP	<p>Protocol</p> <p>Operates on HTTP requests.</p>
REQ.HTTP.METHOD	<p>Qualifier</p> <p>Designates the HTTP method.</p>
REQ.HTTP.URL	<p>Qualifier</p> <p>Designates the URL.</p>
REQ.HTTP.URLTOKENS	<p>Qualifier</p> <p>Designates the URL token.</p>
REQ.HTTP.VERSION	<p>Qualifier</p> <p>Designates the HTTP version.</p>
REQ.HTTP.HEADER	<p>Qualifier</p> <p>Designates the HTTP header.</p>
REQ.HTTP.URLLEN	<p>Qualifier</p> <p>Designates the number of characters in the URL.</p>

REQ.HTTP.URLQUERY	Qualifier Designates the query portion of the URL.
REQ.HTTP.URLQUERYLEN	Qualifier Designates the length of the query portion of the URL.
REQ.SSL	Protocol Operates on SSL requests.
REQ.SSL.CLIENT.CERT	Qualifier Designates the entire client certificate.
REQ.SSL.CLIENT.CERT.SUBJECT	Qualifier Designates the client certificate subject.
REQ.SSL.CLIENT.CERT.ISSUER	Qualifier Designates the issuer of the client certificate.
REQ.SSL.CLIENT.CERT.SIGALGO	Qualifier Designates the validation algorithm used by the client certificate.
REQ.SSL.CLIENT.CERT.VERSION	Qualifier Designates the client certificate version.
REQ.SSL.CLIENT.CERT.VALIDFROM	Qualifier Designates the date before which the client certificate is not valid.
REQ.SSL.CLIENT.CERT.VALIDTO	Qualifier Designates the date after which the client certificate is not valid.
REQ.SSL.CLIENT.CERT.SERIALNUMBER	Qualifier Designates the serial number of the client certificate.
REQ.SSL.CLIENT.CIPHER.TYPE	Qualifier Designates the encryption protocol used by the client.
REQ.SSL.CLIENT.CIPHER.BITS	Qualifier Designates the number of bits used by the client's SSL key.
REQ.SSL.CLIENT.SSL.VERSION	Qualifier Designates the SSL version that the client is using.
REQ.TCP	Protocol

	Operates on incoming TCP packets.
REQ.TCP.SOURCEPORT	Qualifier Designates the source port of the incoming packet.
REQ.TCP.DESTPORT	Qualifier Designates the destination port of the incoming packet.
REQ.IP	Protocol Operates on incoming IP packets.
REQ.IP.SOURCEIP	Qualifier Designates the source IP of the incoming packet.
REQ.IP.DESTIP	Qualifier Designates the destination IP of the incoming packet.
RES	Flow Type Operates on outgoing (or response) packets.
RES.HTTP	Protocol Operates on HTTP responses.
RES.HTTP.VERSION	Qualifier Designates the HTTP version.
RES.HTTP.HEADER	Qualifier Designates the HTTP header.
RES.HTTP.STATUSCODE	Qualifier Designates the status code of the HTTP response.
RES.TCP	Protocol Operates on incoming TCP packets.
RES.TCP.SOURCEPORT	Qualifier Designates the source port of the outgoing packet.
RES.TCP.DESTPORT	Qualifier Designates the destination port of the outgoing packet.
RES.IP	Protocol Operates on outgoing IP packets.



RES.IP.SOURCEIP	<p>Qualifier</p> <p>Designates the source IP of the outgoing packet. This can be in IPv4 or IPv6 format. For example:</p> <pre>add expr exp3 "sourceip == 10.102.32.123 " netmask 255.255.255.0 &amp;&amp; destip == 2001:: 23/120"</pre>
RES.IP.DESTIP	<p>Qualifier</p> <p>Designates the destination IP of the outgoing packet.</p>

## Client Security Expressions

Updated: 2013-10-21

The expressions to configure client settings on the Access Gateway with the following software:

- o Antivirus
- o Personal firewall
- o Antispam
- o Internet Security

For example usage, see <http://support.citrix.com/article/CTX112599>.

Actual Expression	Definition
CLIENT.APPLICATION.AV( <NAME> . VERSION == <VERSION> )	Checks whether the client is running the designated anti-virus program and version.
CLIENT.APPLICATION.AV( <NAME> . VERSION != <VERSION> )	Checks whether the client is not running the designated anti-virus program and version.
CLIENT.APPLICATION.PF( <NAME> . VERSION == <VERSION> )	Checks whether the client is running the designated personal firewall program and version.
CLIENT.APPLICATION.PF( <NAME> . VERSION != <VERSION> )	Checks whether the client is not running the designated personal firewall program and version.
CLIENT.APPLICATION.IS( <NAME> . VERSION == <VERSION> )	Checks whether the client is running the designated internet security program and version.
CLIENT.APPLICATION.IS( <NAME> . VERSION != <VERSION> )	Checks whether the client is not running the designated internet security program and version.
CLIENT.APPLICATION.AS( <NAME> . VERSION == <VERSION> )	Checks whether the client is running the designated anti-spam program and version.
CLIENT.APPLICATION.AS( <NAME> . VERSION != <VERSION> )	Checks whether the client is not running the designated anti-spam program and version.

## Network-Based Expressions

Expression	Definition

REQ	Flow Type.  Operates on incoming, or request, packets.
REQ.VLANID	Qualifier.  Operates on the virtual LAN (VLAN) ID.
REQ.INTERFACE.ID	Qualifier.  Operates on the ID of the designated NetScaler interface.
REQ.INTERFACE.RXTHROUGHPUT	Qualifier.  Operates on the raw received packet throughput of the designated NetScaler interface.
REQ.INTERFACE.TXTHROUGHPUT	Qualifier.  Operates on the raw transmitted packet throughput of the designated NetScaler interface.
REQ.INTERFACE.RXTXTHROUGHPUT	Qualifier.  Operates on the raw received and transmitted packet throughput of the designated NetScaler interface.
REQ.ETHER.SOURCEMAC	Qualifier.  Operates on the source MAC address.
REQ.ETHER.DESTMAC	Qualifier.  Operates on the destination MAC address.
RES	Flow Type.  Operates on outgoing (or response) packets.
RES.VLANID	Qualifier.  Operates on the virtual LAN (VLAN) ID.
RES.INTERFACE.ID	Qualifier.  Operates on the ID of the designated NetScaler interface.
RES.INTERFACE.RXTHROUGHPUT	Qualifier.  Operates on the raw received packet throughput of the designated NetScaler interface.
RES.INTERFACE.TXTHROUGHPUT	Qualifier.  Operates on the raw transmitted packet throughput of the designated NetScaler interface.
RES.INTERFACE.RXTXTHROUGHPUT	Qualifier.

	Operates on the raw received and transmitted packet throughput of the designated NetScaler interface.
RES.ETHER.SOURCEMAC	Qualifier. Operates on the source MAC address.
RES.ETHER.DESTMAC	Qualifier. Operates on the destination MAC address.

## Date/Time Expressions

Expression	Definition
TIME	Qualifier. Operates on the date and time of day, GMT.
DATE	Qualifier. Operates on the date, GMT.
DAYOFWEEK	Operates on the specified day in the week, GMT.

## File System Expressions

Updated: 2013-09-30

You can specify file system expressions in authorization policies for users and groups who access file sharing through the NetScaler Gateway file transfer utility (the VPN portal). These expressions work with the NetScaler Gateway file transfer authorization feature to control user access to file servers, folders, and files. For example, you can use these expressions in authorization policies to control access based on file type and size.

Expression	Definition
FS.COMMAND	Qualifier.  Operates on a file system command. The user can issue multiple commands on a file transfer portal. (For example, ls to list files or mkdir to create a directory). This expression returns the current action that the user is taking.  Possible values: Neighbor, login, ls, get, put, rename, mkdir, rmdir, del, logout, any.  Following is an example:  <pre>Add authorization policy poll "fs.command eq login &amp;&amp; (fs.user eq administrator    fs.serverip eq 10.102.88.221 &amp;" netmask 255.255.255.252)" allow</pre>
FS.USER	Returns the user who is logged on to the file system.
FS.SERVER	Returns the host name of the target server. In the following example, the string win2k3-88-22 is the server name:  <pre>fs.server eq win2k3-88-221</pre>
FS.SERVERIP	Returns the IP address of the target server.

FS.SERVICE	<p>Returns a shared root directory on the file server. If a particular folder is exposed as shared, a user can directly log on to the specified first level folder. This first level folder is called a service. For example, in the path \\hostname\SERVICEX\ETC, SERVICEX is the service. As another example, if a user accesses the file \\hostname\service1\dir1\file1.doc, FS.SERVICE will return service1.</p> <p>Following is an example:</p> <pre>fs.service notcontains New</pre>
FS.DOMAIN	Returns the domain name of the target server.
FS.PATH	<p>Returns the complete path of the file being accessed. For example, if a user accesses the file \\hostname\service1\dir1\file1.doc, FS.PATH will return \service\dir1\file1.doc.</p> <p>Following is an example:</p> <pre>fs.path notcontains SSL</pre>
FS.FILE	Returns the name of the file being accessed. For example, if a user accesses the file \\hostname\service1\dir1\file1.doc, FS.FILE will return file1.doc.
FS.DIR	Returns the directory being accessed. For example, if a user accesses the file \\hostname\service1\dir1\file1.doc, FS.DIR will return \service\dir1.
FS.FILE.ACCESTIME	Returns the time at which the file was last accessed. This is one of several options that provide you with granular control over actions that the user performs. (See the following entries in this table.)
FS.FILE.CREATETIME	Returns the time at which the file was created.
FS.FILE.MODIFYTIME	Returns the time at which the file was edited.
FS.FILE.WRITETIME	Returns the time of the most recent change in the status of the file.
FS.FILE.SIZE	Returns the file size.
FS.DIR.ACCESTIME	Returns the time at which the directory was last accessed.
FS.DIR.CREATETIME	Returns the time at which the directory was created.
FS.DIR.MODIFYTIME	Returns the time at which the directory was last modified.
FS.DIR.WRITETIME	Returns the time at which the directory status last changed.

Note: File system expressions do not support regular expressions.

## Built-In Named Expressions (General)

Expression	Definition
ns_all_apps_ncomp	

	Tests for connections with destination ports between 0 and 65535. In other words, tests for all applications.
ns_cachecontrol_nocache	Tests for connections with an HTTP Cache-Control header that contains the value "no-cache".
ns_cachecontrol_nostore	Tests for connections with an HTTP Cache-Control header that contains the value "no-store".
ns_cmpclient	Tests the client to determine if it accepts compressed content.
ns_content_type	Tests for connections with an HTTP Content-Type header that contains "text".
ns_css	Tests for connections with an HTTP Content-Type header that contains "text/css".
ns_ext_asp	Tests for HTTP connections to any URL that contains the string ".asp" in other words, any connection to an active server page (ASP).
ns_ext_cfm	Tests for HTTP connections to any URL that contains the string ".cfm"
ns_ext_cgi	Tests for HTTP connections to any URL that contains the string ".cgi" in other words, any connection to a common gateway interface (CGI) script.
ns_ext_ex	Tests for HTTP connections to any URL that contains the string ".ex"
ns_ext_exe	Tests for HTTP connections to any URL that contains the string ".exe" in other words, any connection to a executable file.
ns_ext_htx	Tests for HTTP connections to any URL that contains the string ".htx"
ns_ext_not_gif	Tests for HTTP connections to any URL that does not contain the string ".gif" in other words, any connection to a URL that is not a GIF image.
ns_ext_not_jpeg	Tests for HTTP connections to any URL that does not contain the string ".jpeg" in other words, any connection to a URL that is not a JPEG image.
ns_ext_shtml	Tests for HTTP connections to any URL that contains the string ".shtml" in other words, any connection to a server-parsed HTML page.
ns_false	Always returns a value of FALSE.
ns_farclient	<p>Client is in a different geographical region from the NetScaler, as determined by the geographical region in the client's IP address. The following regions are predefined:</p> <p>192.0.0.0 – 193.255.255.255: Multi-regional</p> <p>194.0.0.0 – 195.255.255.255: European Union</p> <p>196.0.0.0 – 197.255.255.255: Other1</p> <p>198.0.0.0 – 199.255.255.255: North America</p>

	<p>200.0.0.0 â€“ 201.255.255.255: Central and South America</p> <p>202.0.0.0 â€“ 203.255.255.255: Pacific Rim</p> <p>204.0.0.0 â€“ 205.255.255.255: Other2</p> <p>206.0.0.0 â€“ 207.255.255.255: Other3</p>
ns_header_cookie	Tests for HTTP connections that contain a Cookie header
ns_header_pragma	Tests for HTTP connections that contain a Pragma: no-cache header.
ns_mozilla_47	Tests for HTTP connections whose User-Agent header contains the string Mozilla/4.7â€”in other words, any connection from a client using the Mozilla 4.7 Web browser.
ns_msexcel	Tests for HTTP connections whose Content-Type header contains the string application/vnd.ms-excelâ€”in other words, any connection transmitting a Microsoft Excel spreadsheet.
ns_msie	Tests for HTTP connections whose User-Agent header contains the string MSIEâ€”in other words, any connection from a client using any version of the Internet Explorer Web browser.
ns_msppt	Tests for HTTP connections whose Content-Type header contains the string application/vnd.ms-powerpointâ€”in other words, any connection transmitting a Microsoft PowerPoint file.
ns_msword	Tests for HTTP connections whose Content-Type header contains the string application/vnd.mswordâ€”in other words, any connection transmitting a Microsoft Word file.
ns_non_get	Tests for HTTP connections that use any HTTP method except for GET.
ns_slowclient	Returns TRUE if the average round trip time between the client and the NetScaler is more than 80 milliseconds.
ns_true	Returns TRUE for all traffic.
ns_url_path_bin	Tests the URL path to see if it points to the /bin/ directory.
ns_url_path_cgibin	Tests the URL path to see if it points to the CGI-BIN directory.
ns_url_path_exec	Tests the URL path to see if it points to the /exec/ directory.
ns_url_tokens	Tests for the presence of URL tokens.
ns_xmldata	Tests for the presence of XML data.

## Built-In Named Expressions (Anti-Virus)

Expression	Definition
McAfee Virus Scan 11	Tests to determine whether the client is running the latest version of McAfee VirusScan.
McAfee Antivirus	Tests to determine whether the client is running any version of McAfee Antivirus.
Symantec AntiVirus 10 (with Updated Definition File)	Tests to determine whether the client is running the most current version of Symantec AntiVirus.
Symantec AntiVirus 6.0	Tests to determine whether the client is running Symantec AntiVirus 6.0.
Symantec AntiVirus 7.5	Tests to determine whether the client is running Symantec AntiVirus 7.5.
TrendMicro OfficeScan 7.3	Tests to determine whether the client is running Trend Microsystemsâ€™ OfficeScan, version 7.3.
TrendMicro AntiVirus 11.25	Tests to determine whether the client is running Trend Microsystemsâ€™ AntiVirus, version 11.25.
Sophos Antivirus 4	Tests to determine whether the client is running Sophos Antivirus, version 4.
Sophos Antivirus 5	Tests to determine whether the client is running Sophos Antivirus, version 5.
Sophos Antivirus 6	Tests to determine whether the client is running Sophos Antivirus, version 6.

## Built-In Named Expressions (Personal Firewall)

Expression	Definition
TrendMicro OfficeScan 7.3	Tests to determine whether the client is running Trend Microsystemsâ€™ OfficeScan, version 7.3.
Sygate Personal Firewall 5.6	Tests to determine whether the client is running the Sygate Personal Firewall, version 5.6.
ZoneAlarm Personal Firewall 6.5	Tests to determine whether the client is running the ZoneAlarm Personal Firewall, version 6.5.

## Built-In Named Expressions (Client Security)

Expression	Definition
Norton Internet Security	Tests to determine whether the client is running any version of Norton Internet Security.

## Summary Examples of Default Syntax Expressions and Policies

The following table provides examples of default syntax expressions that you can use as the basis for your own default syntax expressions.

Table 1. Examples of Default Syntax Expressions

Expression Type	Sample Expressions
Look at the method used in the HTTP request.	<pre>http.req.method.eq(post)</pre> <pre>http.req.method.eq(get)</pre>
Check the Cache-Control or Pragma header value in an HTTP request (req) or response (res).	<pre>http.req.header("Cache-Control").contains("no-store")</pre> <pre>http.req.header("Cache-Control").contains("no-cache")</pre> <pre>http.req.header("Pragma").contains("no-cache")</pre> <pre>http.res.header("Cache-Control").contains("private")</pre> <pre>http.res.header("Cache-Control").contains("public")</pre> <pre>http.res.header("Cache-Control").contains("must-revalidate")</pre> <pre>http.res.header("Cache-Control").contains("proxy-revalidate")</pre> <pre>http.res.header("Cache-Control").contains("max-age")</pre>
Check for the presence of a header in a request (req) or response (res).	<pre>http.req.header("myHeader").exists</pre> <pre>http.res.header("myHeader").exists</pre>
Look for a particular file type in an HTTP request based on the file extension.	<pre>http.req.url.contains(".html")</pre> <pre>http.req.url.contains(".cgi")</pre> <pre>http.req.url.contains(".asp")</pre> <pre>http.req.url.contains(".exe")</pre> <pre>http.req.url.contains(".cfm")</pre> <pre>http.req.url.contains(".ex")</pre> <pre>http.req.url.contains(".shtml")</pre> <pre>http.req.url.contains(".htx")</pre> <pre>http.req.url.contains("/cgi-bin/")</pre> <pre>http.req.url.contains("/exec/")</pre> <pre>http.req.url.contains("/bin/")</pre>
Look for anything that is other than a particular file type in an	<pre>http.req.url.contains(".gif").not</pre> <pre>http.req.url.contains(".jpeg").not</pre>



HTTP request.	
Check the type of file that is being sent in an HTTP response based on the Content-Type header.	<pre>http.res.header("Content-Type").contains("text") http.res.header("Content-Type").contains("application/msword") http.res.header("Content-Type").contains("vnd.ms-excel") http.res.header("Content-Type").contains("application/vnd.ms-powerpoint") http.res.header("Content-Type").contains("text/css") http.res.header("Content-Type").contains("text/xml") http.res.header("Content-Type").contains("image/")</pre>
Check whether this response contains an expiration header.	<pre>http.res.header("Expires").exists</pre>
Check for a Set-Cookie header in a response.	<pre>http.res.header("Set-Cookie").exists</pre>
Check the agent that sent the response.	<pre>http.res.header("User-Agent").contains("Mozilla/4.7") http.res.header("User-Agent").contains("MSIE")</pre>
Check if the first 1024 bytes of the body of a request starts with the string "some text".	<pre>http.req.body(1024).contains("some text")</pre>

The following table shows examples of policy configurations and bindings for commonly used functions.

Table 2. Examples of Default Syntax Expressions and Policies

Purpose	Example
Use the rewrite feature to replace occurrences of http:// with https:// in the body of an HTTP response.	<pre>add rewrite action httpRewriteAction replace_all http.res.body(50000) "\"https://\"" -pattern http://  add rewrite policy demo_rep34312 "http.res.body (50000).contains(\"http://\")" httpRewriteAction</pre>
Replace all occurrences of "abcd" with "1234" in the first 1000 bytes of the HTTP body.	<pre>add rewrite action abcdTo1234Action replace_all "http.req.body(1000)" "\"1234\"" -pattern abcd  add rewrite policy abcdTo1234Policy "http.req.body (1000).contains(\"abcd\")" abcdTo1234Action  bind rewrite global abcdTo1234Policy 100 END -type REQ_OVERRIDE</pre>

Downgrade the HTTP version to 1.0 to prevent the server from chunking HTTP responses.	<pre>add rewrite action downgradeTo1.0Action replace http.req.version.minor "\0"</pre> <pre>add rewrite policy downgradeTo1.0Policy "http.req.version.minor.eq(1)" downgradeTo1.0Action</pre> <pre>bind lb vserver myLBVserver -policyName downgradeTo1.0Policy -priority 100 -gotoPriorityExpression NEXT -type REQUEST</pre>
Remove references to the HTTP or HTTPS protocol in all responses, so that if the user's connection is HTTP, the link is opened by using HTTP, and if the user's connection is HTTPS, the link is opened by using HTTPS.	<pre>add rewrite action remove_http_https replace_all "http.res.body(1000000).set_text_mode(ignorecase) "\"/\" -pattern "re~https?:// HTTPS?://~"</pre> <pre>add rewrite policy remove_http_https true remove_http_https</pre> <pre>bind lb vserver test_vsvr -policyName remove_http_https -priority 20 -gotoPriorityExpression NEXT -type RESPONSE</pre>
<p>Rewrite instances of http:// to https:// in all URLs.</p> <p>This policy uses the responder functionality.</p>	<pre>add responder action httpToHttpsAction redirect "\"https://\" + http.req.hostname + http.req.url" -bypassSafetyCheck YES</pre> <pre>add responder policy httpToHttpsPolicy "!CLIENT.SSL.IS_SSL" httpToHttpsAction</pre> <pre>bind responder global httpToHttpsPolicy 1 END -type OVERRIDE</pre>
<p>Modify a URL to redirect from URL A to URL B. In this example, file5.html is appended to the path.</p> <p>This policy uses the responder functionality.</p>	<pre>add responder action appendFile5Action redirect "\"http://\" + http.req.hostname + http.req.url + \"/file5.html\" -bypassSafetyCheck YES</pre> <pre>add responder policy appendFile5Policy "http.req.url.eq(\"/testsite\")" appendFile5Action</pre> <pre>bind responder global appendFile5Policy 1 END -type OVERRIDE</pre>
Redirect an external URL to an internal URL.	<pre>add rewrite action act_external_to_internal REPLACE 'http.req.hostname.server' 'www.my.host.com'</pre> <pre>add rewrite policy pol_external_to_internal 'http.req.hostname.server.eq("www.external.host.com")' act_external_to_internal</pre> <pre>bind rewrite global pol_external_to_internal 100 END -type REQ_OVERRIDE</pre>
Redirect requests to www.example.com that have a query string to www.Webn.example.com. The value n is derived from a server parameter in the query string, for example, server=5.	<pre>add rewrite action act_redirect_query REPLACE q#http.req.header("Host").before_str(".example.com") 'Web' + http.req.url.query.value("server")#</pre> <pre>add rewrite policy pol_redirect_query q#http.req.header("Host").eq("www.example.com") &amp;&amp; http.req.url.contains("?")' act_redirect_query#</pre>
Limit the number of requests per second from a URL.	<pre>add ns limitSelector ip_limit_selector http.req.url "client.ip.src"</pre>

	<pre> add ns limitIdentifier ip_limit_identifier - threshold 4 -timeSlice 3600 -mode request_rate - limitType smooth -selectorName ip_limit_selector  add responder action my_Web_site_redirect_action redirect "\"http://www.mycompany.com/\\""  add responder policy ip_limit_responder_policy "http. req.url.contains(\"myasp.asp\") &amp;&amp; sys.check_limit(\" ip_limit_identifier\")" my_Web_site_redirect_action  bind responder global ip_limit_responder_policy 100 END -type default </pre>
Check the client IP address but pass the request without modifying the request.	<pre> add rewrite policy check_client_ip_policy 'HTTP.REQ. HEADER("x-forwarded-for").EXISTS    HTTP.REQ.HEADER ("client-ip").EXISTS' NOREWRITE  bind rewrite global check_client_ip_policy 100 END </pre>
Remove old headers from a request and insert an NS-Client header.	<pre> add rewrite action del_x_forwarded_for delete_http_header x-forwarded-for  add rewrite action del_client_ip delete_http_header client-ip  add rewrite policy check_x_forwarded_for_policy 'HTTP.REQ.HEADER("x-forwarded-for").EXISTS' del_x_forwarded_for  add rewrite policy check_client_ip_policy 'HTTP.REQ. HEADER("client-ip").EXISTS' del_client_ip  add rewrite action insert_ns_client_header insert_http_header NS-Client 'CLIENT.IP.SRC'  add rewrite policy insert_ns_client_policy 'HTTP.REQ. HEADER("x-forwarded-for").EXISTS    HTTP.REQ.HEADER ("client-ip").EXISTS' insert_ns_client_header  bind rewrite global check_x_forwarded_for_policy 100 200  bind rewrite global check_client_ip_policy 200 300  bind rewrite global insert_ns_client_policy 300 END </pre>
<p>Remove old headers from a request, insert an NS-Client header, and then modify the <code>insert_header</code> action so that the value of the inserted header contains the client IP values from the old headers and the NetScaler appliance's connection IP address.</p> <p>Note that this example repeats the previous example, with the exception of the final <code>set rewrite action</code>.</p>	<pre> add rewrite action del_x_forwarded_for delete_http_header x-forwarded-for  add rewrite action del_client_ip delete_http_header client-ip  add rewrite policy check_x_forwarded_for_policy 'HTTP.REQ.HEADER("x-forwarded-for").EXISTS' del_x_forwarded_for  add rewrite policy check_client_ip_policy 'HTTP.REQ. HEADER("client-ip").EXISTS' del_client_ip  add rewrite action insert_ns_client_header insert_http_header NS-Client 'CLIENT.IP.SRC'  add rewrite policy insert_ns_client_policy 'HTTP.REQ. HEADER("x-forwarded-for").EXISTS    HTTP.REQ.HEADER ("client-ip").EXISTS' insert_ns_client_header </pre>

```
bind rewrite global check_x_forwarded_for_policy 100
200

bind rewrite global check_client_ip_policy 200 300

bind rewrite global insert_ns_client_policy 300 END

set rewrite action insert_ns_client_header -
stringBuilderExpr 'HTTP.REQ.HEADER("x-forwarded-
for").VALUE(0) + " " + HTTP.REQ.HEADER("client-ip").
VALUE(0) + " " + CLIENT.IP.SRC' -bypassSafetyCheck
YES
```

## Tutorial Examples of Default Syntax Policies for Rewrite

With the rewrite feature, you can modify any part of an HTTP header, and, for responses, you can modify the HTTP body. You can use this feature to accomplish a number of useful tasks, such as removing unnecessary HTTP headers, masking internal URLs, redirecting Web pages, and redirecting queries or keywords.

In the following examples, you first create a rewrite action and a rewrite policy. Then you bind the policy globally.

This document includes the following details:

- [Redirecting an External URL to an Internal URL](#)
- [Redirecting a Query](#)
- [Rewriting HTTP to HTTPS](#)
- [Removing Unwanted Headers](#)
- [Reducing Web Server Redirects](#)
- [Masking the Server Header](#)

### Redirecting an External URL to an Internal URL

Updated: 2013-10-29

This example describes how to create a rewrite action and rewrite policy that redirects an external URL to an internal URL. You create an action, called `act_external_to_internal`, that performs the rewrite. Then you create a policy called `pol_external_to_internal`.

#### To redirect an external URL to an internal URL by using the command line interface

- To create the rewrite action, at the command prompt, type:

```
add rewrite action act_external_to_internal REPLACE 'http.req.hostname.server'
"host_name_of_internal_Web_server"
```

- To create the rewrite policy, at the NetScaler command prompt, type:

```
add rewrite policy pol_external_to_internal 'http.req.hostname.server.eq
("host_name_of_external_Web_server")' act_external_to_internal
```

- Bind the policy globally.

#### To redirect an external URL to an internal URL by using the configuration utility

1. Navigate to AppExpert > Rewrite > Actions.
2. In the details pane, click Add.
3. In the Create Rewrite Action dialog box, enter the name `act_external_to_internal`.
4. To replace the HTTP server host name with the internal server name, choose Replace from the Type list box.
5. In the Header Name field, type Host.
6. In the String expression for replacement text field, type the internal host name of your Web server.
7. Click Create and then click Close.
8. In the navigation pane, click Policies.
9. In the details pane, click Add.
10. In the Name field, type `pol_external_to_internal`. This policy will detect connections to the Web server.
11. In the Action drop-down menu, choose the action `act_external_to_internal`.
12. In the Expression editor, construct the following expression:

```
HTTP.REQ.HOSTNAME.SERVER.EQ("www.example.com")
```

13. Bind your new policy globally.

### Redirecting a Query

This example describes how to create a rewrite action and rewrite policy that redirects a query to the proper URL. The example assumes that the request contains a Host header set to `www.example.com` and a GET method with the `string /query.cgi?server=5`. The redirect extracts the domain name from the host header and the number from the query string, and redirects the user's query to the server `web5.example.com`, where the rest of the user's query is processed.

Note: Although the following commands appears on multiple lines, you should enter them on a single line without line breaks.

## To redirect a query to the appropriate URL using the command line

- o To create a rewrite action named `act_redirect_query` that replaces the HTTP server host name with the internal server name, type:

```
add rewrite action act_redirect_query REPLACE q#http.req.header("Host").before_str(".example.com")'
""Web" + http.req.url.query.value("server")#
```

- o To create a rewrite policy named `pol_redirect_query`, type the following commands at the NetScaler command prompt.. This policy detects connections, to the Web server, that contain a query string. Do not apply this policy to connections that do not contain a query string:

```
add rewrite policy pol_redirect_query q#http.req.header("Host").eq("www.example.com") && http.req.url.
contains("?")' act_redirect_query#
```

- o Bind your new policy globally.

Because this rewrite policy is highly specific and should be run before any other rewrite policies, it is advisable to assign it a high priority. If you assign it a priority of 1, it will be evaluated first.

## Rewriting HTTP to HTTPS

Updated: 2014-09-17

This example describes how to rewrite Web server responses to find all URLs that begin with the string `http` and replace that string with `https`. You can use this to avoid having to update Web pages after moving a server from HTTP to HTTPS.

### To redirect HTTP URLs to HTTPS by using the command line interface

- o To create a rewrite action named `act_replace_http_with_https` that replaces all instances of the string `http` with the string `https`, enter the following command:

```
add rewrite action act_replace_http_with_https replace_all 'http.res.body(100)' 'https' -pattern http
```

- o To create a rewrite policy named `pol_replace_http_with_https` that detects connections to the Web server, enter the following command:

```
add rewrite policy pol_replace_http_with_https TRUE act_replace_http_with_https NOREWRITE
```

- o Bind your new policy globally.

To troubleshoot this rewrite operation, see ["Case Study: Rewrite Policy for Converting HTTP Links to HTTPS not Working."](#)

## Removing Unwanted Headers

Updated: 2013-09-02

This example explains how to use a Rewrite policy to remove unwanted headers. Specifically, the example shows how to remove the following headers:

- o **Accept Encoding header.** Removing the Accept Encoding header from HTTP responses prevents compression of the response.
- o **Content Location header.** Removing the Content Location header from HTTP responses prevents your server from providing a hacker with information that might allow a security breach.

To delete headers from HTTP responses, you create a rewrite action and a rewrite policy, and you bind the policy globally.

### To create the appropriate Rewrite action by using the command line interface

At the command prompt, type one of the following commands to either remove the Accept Encoding header and prevent response compression or remove the Content Location header:

- o `add rewrite action "act_remove-ae" delete_http_header "Accept-Encoding"`
- o `add rewrite action "act_remove-cl" delete_http_header "Content-Location"`

## To create the appropriate Rewrite policy by using the command line interface

At the command prompt, type one of the following commands to remove either the Accept Encoding header or the Content Location header:

- o add rewrite policy "pol\_remove-ae" true "act\_remove-ae"
- o add rewrite policy "pol\_remove-cl" true "act\_remove-cl"

## To bind the policy globally by using the command line interface

At the command prompt, type one of the following commands, as appropriate, to globally bind the policy that you have created:

- o bind rewrite global pol\_remove\_ae 100
- o bind rewrite global pol\_remove\_cl 200

## Reducing Web Server Redirects

This example explains how to use a Rewrite policy to modify connections to your home page and other URLs that end with a forward slash (/) to the default index page for your server, preventing redirects and reducing load on your server.

### To modify directory-level HTTP requests to include the default home page by using the command line

- o To create a Rewrite action named action-default-homepage that modifies URLs that end in a forward slash to include the default home page index.html, type:

```
add rewrite action "action-default-homepage" replace q#http.req.url.path "/" "/index.html"#
```

- o To create a Rewrite policy named policy-default-homepage that detects connections to your home page and applies your new action, type:

```
add rewrite policy "policy-default-homepage" q#http.req.url.path.EQ("/") "action-default-homepage"#
```

- o Globally bind your new policy to put it into effect.

## Masking the Server Header

This example explains how to use a Rewrite policy to mask the information in the Server header in HTTP responses from your Web server. That header contains information that hackers can use to compromise your Web site. While masking the header will not prevent a skilled hacker from finding out information about your server, it will make hacking your Web server more difficult and encourage hackers to choose less well protected targets.

### To mask the Server header in responses from the command line

1. To create a Rewrite action named act\_mask-server that replaces the contents of the Server header with an uninformative string, type:

```
add rewrite action "act_mask-server" replace "http.RES.HEADER(\"Server\")" "\"Web Server 1.0\""
```

2. To create a Rewrite policy named pol\_mask-server that detects all connections, type:

```
add rewrite policy "pol_mask-server" true "act_mask-server"
```

3. Globally bind your new policy to put it into effect.

## Tutorial Examples of Classic Policies

The following examples describe useful examples of classic policy configuration for certain NetScaler features, such as NetScaler Gateway, application firewall, and SSL.

This document includes the following details:

- [NetScaler Gateway Policy to Check for a Valid Client Certificate](#)
- [Application Firewall Policy to Protect a Shopping Cart Application](#)
- [Application Firewall Policy to Protect Scripted Web Pages](#)
- [DNS Policy to Drop Packets from Specific IPs](#)
- [SSL Policy to Require Valid Client Certificates](#)

### NetScaler Gateway Policy to Check for a Valid Client Certificate

Updated: 2014-09-25

The following policies enable the NetScaler to ensure that a client presents a valid certificate before establishing a connection to a company's SSL VPN.

#### To check for a valid client certificate by using the command line interface

- Add an action to perform client certificate authentication.

```
add ssl action act1 -clientAuth DOCLIENTAUTH
```

- Create an SSL policy to evaluate the client requests.

```
add ssl policy pol1 -rule "REQ.HTTP.METHOD == GET" -action act1
```

- Add a rewrite action to insert the certificate issuer details into the HTTP header of the requests being sent to web server.

```
add rewrite action act2 insert_http_header "CertDN" CLIENT.SSL.CLIENT_CERT.SUBJECT
```

- Create a rewrite policy to insert the certificate issuer details, if the client certificate exists.

```
add rewrite policy pol2 "CLIENT.SSL.CLIENT_CERT.EXISTS" act2
```

Bind these new policies to the NetScaler VIP to put them into effect.

### Application Firewall Policy to Protect a Shopping Cart Application

Updated: 2013-09-02

Shopping cart applications handle sensitive customer information, for example, credit card numbers and expiration dates, and they access back-end database servers. Many shopping cart applications also use legacy CGI scripts, which can contain security flaws that were unknown at the time they were written, but are now known to hackers and identity thieves.

A shopping cart application is particularly vulnerable to the following attacks:

- **Cookie tampering.** If a shopping cart application uses cookies, and does not perform the appropriate checks on the cookies that users return to the application, an attacker could modify a cookie and gain access to the shopping cart application under another user's credentials. Once logged on as that user, the attacker could obtain sensitive private information about the legitimate user or place orders using the legitimate user's account.
- **SQL injection.** A shopping cart application normally accesses a back-end database server. Unless the application performs the appropriate safety checks on the data users return in the form fields of its Web forms before it passes that information on to the SQL database, an attacker can use a Web form to inject unauthorized SQL commands into the database server. Attackers normally use this type of attack to obtain sensitive private information from the database or modify information in the database.

The following configuration will protect a shopping cart application against these and other attacks.



## To protect a shopping cart application by using the configuration utility

1. Navigate to Security > Application Firewall > Profiles, and then click Add.
2. In the Create Application Firewall Profile dialog box, in the Profile Name field, enter shopping\_cart.
3. In the Profile Type drop-down list, select Web Application.
4. In the Configure Select Advanced defaults.
5. Click Create and then click Close.
6. In the details view, double-click the new profile.
7. In the Configure Web Application Profile dialog box, configure your new profile as described below:
  - a. Click the Checks tab, double-click the Start URL check, and in the Modify Start URL Check dialog box, click the General tab and disable blocking, and enable learning, logging, statistics, and URL closure. Click OK and then click Close.

Note that if you are using the command line, you configure these settings by typing the following at the prompt, and pressing ENTER:

```
set appfw profile shopping_cart -startURLAction LEARN LOG STATS -startURLClosure ON
```

- b. For the Cookie Consistency check and Form Field Consistency checks, disable blocking, and enable learning, logging, statistics, using a similar method to the Modify Start URL Check configuration.

If you are using the command line, you configure these settings by typing the following commands:

```
set appfw profile shopping_cart -cookieConsistencyAction LEARN LOG STATS
```

```
set appfw profile shopping_cart -fieldConsistencyAction LEARN LOG STATS
```

- c. For the SQL Injection check, disable blocking, and enable learning, logging, statistics, and transformation of special characters in the Modify SQL Injection Check dialog box, General tab, Check Actions section.

If you are using the command line, you configure these settings by typing the following at the prompt, and pressing ENTER:

```
set appfw profile shopping_cart -SQLInjectionAction LEARN LOG STATS -  
SQLInjectionTransformSpecialChars ON
```

- d. For the Credit Card check, disable blocking; enable logging, statistics, and masking of credit card numbers; and enable protection for those credit cards you accept as forms of payment.

- If you are using the configuration utility, you configure blocking, logging, statistics, and masking (or *x-out*) in the Modify Credit Card Check dialog box, General tab, Check Actions section. You configure protection for specific credit cards in the Settings tab of the same dialog box.
- If you are using the command line, you configure these settings by typing the following at the prompt, and pressing ENTER:

```
set appfw profile shopping_cart -creditCardAction LOG STATS -creditCardXOut ON -creditCard <name>  
[<name>...]
```

For <name> you substitute the name of the credit card you want to protect. For Visa, you substitute VISA. For Master Card, you substitute MasterCard. For American Express, you substitute Amex. For Discover, you substitute Discover. For Diners Club, you substitute DinersClub. For JCB, you substitute JCB.

8. Create a policy named shopping\_cart that detects connections to your shopping cart application and applies the shopping\_cart profile to those connections.

To detect connections to the shopping cart, you examine the URL of incoming connections. If you host your shopping cart application on a separate host (a wise measure for security and other reasons), you can simply look for the presence of that host in the URL. If you host your shopping cart in a directory on a host that handles other traffic, as well, you must determine that the connection is going to the appropriate directory and/or HTML page.

The process for detecting either of these is the same; you create a policy based on the following expression, and substitute the proper host or URL for <string>.

```
REQ.HTTP.HEADER URL CONTAINS <string>
```

- If you are using the configuration utility, you navigate to the application firewall Policies page, click the Add... button to add a new policy, and follow the policy creation process described in “To create a policy with classic expressions using the configuration utility” beginning on page 201 and following.
- If you are using the command line, you type the following command at the prompt and press Enter:

```
add appfw policy shopping_cart "REQ.HTTP.HEADER URL CONTAINS <string>" shopping_cart
```

9. Globally bind your new policy to put it into effect.

Because you want to ensure that this policy will match all connections to the shopping cart, and not be preempted by another more general policy, you should assign a high priority to it. If you assign one (1) as the priority, no other policy can preempt this one.

## Application Firewall Policy to Protect Scripted Web Pages

Updated: 2013-11-14

Web pages with embedded scripts, especially legacy JavaScripts, often violate the "same origin rule," which does not allow scripts to access or modify content on any server but the server where they are located. This security vulnerability is called *cross-site scripting*. The application firewall Cross-Site Scripting rule normally filters out requests that contain cross-site scripting.

Unfortunately, this can cause Web pages with older JavaScripts to stop functioning, even when your system administrator has checked those scripts and knows that they are safe. The example below explains how to configure the application firewall to allow cross-site scripting in Web pages from trusted sources without disabling this important filter for the rest of your Web sites.

### To protect Web pages with cross-site scripting by using the command line interface

- At the command line, to create an advanced profile, type:

```
add appfw profile pr_xssokay -defaults advanced
```

- To configure the profile, type:

```
set appfw profile pr_xssokay -startURLAction NONE -startURLClosure OFF -cookieConsistencyAction LEARN LOG STATS -fieldConsistencyAction LEARN LOG STATS -crossSiteScriptingAction LEARN LOG STATS$"
```

- Create a policy that detects connections to your scripted Web pages and applies the pr\_xssokay profile, type:

```
add appfw policy pol_xssokay "REQ.HTTP.HEADER URL CONTAINS ^\.p\?$ || REQ.HTTP.HEADER URL CONTAINS ^\.js$" pr_xssokay
```

- Globally bind the policy.

### To protect Web pages with cross-site scripting by using the configuration utility

- Navigate to Security > Application Firewall > Profiles.
- In the details view, click Add.
- In the Create Application Firewall Profile dialog box, create a Web Application profile with advanced defaults and name it pr\_xssokay. Click Create and then click Close.
- In the details view, click the profile, click Open, and in the Configure Web Application Profile dialog box, configure the pr\_xssokay profile as shown below.

Start URL Check: Clear all actions.

- Cookie Consistency Check: Disable blocking.
- Form Field Consistency Check: Disable blocking.
- Cross-Site Scripting Check: Disable blocking.

This should prevent blocking of legitimate requests involving Web pages with cross-site scripting that you know are nonetheless safe.

- Click Policies, and then click Add.
- In the Create Application Firewall Policy dialog box, create a policy that detects connections to your scripted Web pages and applies the pr\_xssokay profile:

- Policy name: pol\_xssokay
- Associated profile: pr\_xssokay

Policy expression: "REQ.HTTP.HEADER URL CONTAINS ^\.p1\?\$ || REQ.HTTP.HEADER URL CONTAINS ^\.js\$"

- Globally bind your new policy to put it into effect.

## DNS Policy to Drop Packets from Specific IPs

Updated: 2013-09-02

The following example describes how to create a DNS action and DNS policy that detects connections from unwanted IPs or networks, such as those used in a DDOS attack, and drops all packets from those locations. The example shows networks within the IANA reserved IP block 192.168.0.0/16. A hostile network will normally be on publicly routable IPs.

### To drop packets from specific IPs by using the command line interface

- o To create a DNS policy named pol\_ddos\_drop that detects connections from hostile networks and drops those packets, type:

```
add dns policy pol_ddos_drop 'client.ip.src.in_subnet(192.168.253.128/25) || client.ip.src.in_subnet(192.168.254.32/27)' -drop YES'
```

For the example networks in the 192.168.0.0/16 range, you substitute the IP and netmask in ###.###.###.###/## format of each network you want to block. You can include as many networks as you want, separating each CLIENT.IP.SRC.IN\_SUBNET(###.###.###.###/##) command with the OR operator.

- o Globally bind your new policy to put it into effect.

## SSL Policy to Require Valid Client Certificates

Updated: 2013-09-02

The following example shows an SSL policy that checks the user's client certificate validity before initiating an SSL connection with a client.

### To block connections from users with expired client certificates

- o Log on to the command line interface.

If you are using the GUI, navigate to the SSL Policies page, then in the Data area, click the Actions tab.

- o Create an SSL action named act\_current\_client\_cert that requires that users have a current client certificate to establish an SSL connection with the NetScaler.

```
add ssl action act_current_client_cert-clientAuth DOCLIENTAUTH -clientCert ENABLED -certHeader "clientCertificateHeader" -clientCertNotBefore ENABLED -certNotBeforeHeader "Mon, 01 Jan 2007 00:00:00 GMT"
```

- o Create an SSL policy named pol\_current\_client\_cert that detects connections to the Web server that contain a query string.

```
add ssl policy pol_current_client_cert 'REQ.SSL.CLIENT.CERT.VALIDFROM >= "Mon, 01 Jan 2007 00:00:00 GMT"' act_block_ssl
```

- o Bind your new policy globally.

Because this SSL policy should apply to any user's SSL connection unless a more specific SSL policy applies, you may want to assign it a low priority. If you assign it a priority of one thousand (1000), that should ensure that other SSL policies are evaluated first, meaning that this policy will apply only to connections that do not match more specific policy criteria.

## Migration of Apache mod\_rewrite Rules to the Default Syntax

The Apache HTTP Server provides an engine known as mod\_rewrite for rewriting HTTP request URLs. If you migrate the mod\_rewrite rules from Apache to the NetScaler, you boost back-end server performance. In addition, because the NetScaler typically load balances multiple (sometimes thousands of) Web servers, after migrating the rules to the NetScaler you will have a single point of control for these rules.

Following are examples of mod\_rewrite functions, and translations of these functions into Rewrite and Responder policies on the NetScaler.

This document includes the following details:

- [Converting URL Variations into Canonical URLs](#)
- [Converting Host Name Variations to Canonical Host Names](#)
- [Moving a Document Root](#)
- [Moving Home Directories to a New Web Server](#)
- [Working with Structured Home Directories](#)
- [Redirecting Invalid URLs to Other Web Servers](#)
- [Rewriting a URL Based on Time](#)
- [Redirecting to a New File Name \(Invisible to the User\)](#)
- [Redirecting to New File Name \(User-Visible URL\)](#)
- [Accommodating Browser Dependent Content](#)
- [Blocking Access by Robots](#)
- [Blocking Access to Inline Images](#)
- [Creating Extensionless Links](#)
- [Redirecting a Working URI to a New Format](#)
- [Ensuring That a Secure Server Is Used for Selected Pages](#)

## Converting URL Variations into Canonical URLs

On some Web servers you can have multiple URLs for a resource. Although the canonical URLs should be used and distributed, other URLs can exist as shortcuts or internal URLs. You can make sure that users see the canonical URL regardless of the URL used to make an initial request.

In the following examples, the URL /~user is converted to /u/user.

### Apache mod\_rewrite solution for converting a URL

```
RewriteRule    ^/~([^/]+)?/(.*)    /u/$1/$2[R]
```

### NetScaler solution for converting a URL

```
add responder action act1 redirect '" /u/" + HTTP.REQ.URL.AFTER_STR("/~")' -bypassSafetyCheck y
add responder policy poll 'HTTP.REQ.URL.STARTSWITH("/~") && HTTP.REQ.URL.LENGTH.GT(2)' act1
bind responder global poll 100
```

## Converting Host Name Variations to Canonical Host Names

You can enforce the use of a particular host name for reaching a site. For example, you can enforce the use of www.example.com instead of example.com.

### Apache mod\_rewrite solution for enforcing a particular host name for sites running on a port other than 80

```
RewriteCond %{HTTP_HOST} !^www.example.com
RewriteCond %{HTTP_HOST} !^$
RewriteCond %{SERVER_PORT} !^80$
RewriteRule ^/(.*) http://www.example.com:%{SERVER_PORT}/$1 [L,R]
```

### Apache mod\_rewrite solution for enforcing a particular host name for sites running on port 80

```
RewriteCond %{HTTP_HOST} !^www.example.com
RewriteCond %{HTTP_HOST} !^$
RewriteRule ^/(.*) http://www.example.com/$1 [L,R]
```

### NetScaler solution for enforcing a particular host name for sites running on a port other than 80

```
add responder action act1 redirect '"http://www.example.com:"+CLIENT.TCP.DSTPORT+HTTP.REQ.UR
add responder policy poll '!HTTP.REQ.HOSTNAME.CONTAINS("www.example.com")&&!HTTP.REQ.HOSTNAM
bind responder global poll 100 END
```

### NetScaler solution for enforcing a particular host name for sites running on port 80

```
add responder action act1 redirect '"http://www.example.com"+HTTP.REQ.URL' -bypassSafetyChec
add responder policy poll '!HTTP.REQ.HOSTNAME.CONTAINS("www.example.com")&&!HTTP.REQ.HOSTNAM
bind responder global poll 100 END
```

## Moving a Document Root

Usually the document root of a Web server is based on the URL `http://www.example.com/`. However, the document root can be any directory. You can redirect traffic to the document root if it changes from the top-level `http://www.example.com/` directory to another directory.

In the following examples, you change the document root from `/` to `/e/www/`. The first two examples simply replace one string with another. The third example is more universal because, along with replacing the root directory, it preserves the rest of the URL (the path and query string), for example, redirecting `/example/file.html` to `/e/www/example/file.html`.

### Apache `mod_rewrite` solution for moving the document root

```
RewriteEngine on
RewriteRule ^/$ /e/www/ [R]
```

### NetScaler solution for moving the document root

```
add responder action act1 redirect '"e/www/' -bypassSafetyCheck yes
add responder policy poll 'HTTP.REQ.URL.EQ("/")' act1
bind responder global poll 100
```

### NetScaler solution for moving the document root and appending path information to the request

```
add responder action act1 redirect '"e/www"+HTTP.REQ.URL' -bypassSafetyCheck yes
add responder policy poll '!HTTP.REQ.URL.STARTSWITH("/e/www/")' act1
bind responder global poll 100 END
```

## Moving Home Directories to a New Web Server

You may want to redirect requests that are sent to home directories on a Web server to a different Web server. For example, if a new Web server is replacing an old one over time, as you migrate home directories to the new location you need to redirect requests for the migrated home directories to the new Web server.

In the following examples, the host name for the new Web server is `newserver`.

### Apache `mod_rewrite` solution for redirecting to another Web server

```
RewriteRule ^/(.+) http://newserver/$1 [R,L]
```

### NetScaler solution for redirecting to another Web server (method 1)

```
add responder action act1 redirect '"http://newserver"+HTTP.REQ.URL' -bypassSafetyCheck yes
add responder policy poll 'HTTP.REQ.URL.REGEX_MATCH(re#^/(.+)#)' act1
bind responder global poll 100 END
```

### NetScaler solution for redirecting to another Web server (method 2)

```
add responder action act1 redirect '"http://newserver"+HTTP.REQ.URL' -bypassSafetyCheck yes
add responder policy poll 'HTTP.REQ.URL.LENGTH.GT(1)' act1
bind responder global poll 100 END
```

## Working with Structured Home Directories

Typically, a site with thousands of users has a structured home directory layout. For example, each home directory may reside under a subdirectory that is named using the first character of the user name. For example, the home directory for jsmith (/~jsmith/anypath) might be /home/j/smith/.www/anypath, and the home directory for rvalveti (/~rvalveti/anypath) might be /home/r/rvalveti/.www/anypath.

The following examples redirect requests to the home directory.

### Apache mod\_rewrite solution for structured home directories

```
RewriteRule ^/~((([a-z])[a-z0-9]+)(.*)) /home/$2/$1/.www$3
```

### NetScaler solution for structured home directories

NetScaler solution for structured home directories

```
add rewrite action act1 replace 'HTTP.REQ.URL' '/'home/" + HTTP.REQ.URL.AFTER_STR("~/").PREFIX
add rewrite policy poll1 'HTTP.REQ.URL.PATH.STARTSWITH("~/")' act1
bind rewrite global poll1 100
```

## Redirecting Invalid URLs to Other Web Servers

If a URL is not valid, it should be redirected to another Web server. For example, you should redirect to another Web server if a file that is named in a URL does not exist on the server that is named in the URL.

On Apache, you can perform this check using mod\_rewrite. On the NetScaler, an HTTP callout can check for a file on a server by running a script on the server. In the following NetScaler examples, a script named file\_check.cgi processes the URL and uses this information to check for the presence of the target file on the server. The script returns TRUE or FALSE, and the NetScaler uses the value that the script returns to validate the policy.

In addition to performing the redirection, the NetScaler can add custom headers or, as in the second NetScaler example, it can add text in the response body.

### Apache mod\_rewrite solution for redirection if a URL is wrong

```
RewriteCond /your/docroot/%{REQUEST_FILENAME} !-f
RewriteRule ^(.+) http://webserverB.com/$1 [R]
```

### NetScaler solution for redirection if a URL is wrong (method 1)

```
add HTTPCallout Call
set policy httpCallout Call -IPAddress 10.102.59.101 -port 80 -hostExpr '"10.102.59.101"' -r
add responder action act1 redirect '"http://webserverB.com"+HTTP.REQ.URL' -bypassSafetyCheck
add responder policy poll1 '!HTTP.REQ.HEADER("Name").EXISTS && !SYS.HTTP_CALLOUT(call)' act1
bind responder global poll1 100
```

### NetScaler solution for redirection if a URL is wrong (method 2)

```
add HTTPCallout Call
set policy httpCallout Call -IPAddress 10.102.59.101 -port 80 -hostExpr '"10.102.59.101"' -r
add responder action act1 respondwith '"HTTP/1.1 302 Moved Temporarily\r\nLocation: http://
add responder policy poll1 '!HTTP.REQ.HEADER("Name").EXISTS && !SYS.HTTP_CALLOUT(call)' act1
bind responder global poll1 100
```

## Rewriting a URL Based on Time

You can rewrite a URL based on the time. The following examples change a request for example.html to example.day.html or example.night.html, depending on the time of day.

### Apache mod\_rewrite solution for rewriting a URL based on the time

```
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^example\.html$ example.day.html [L]
RewriteRule ^example\.html$ example.night.html
```

### NetScaler solution for rewriting a URL based on the time

```
add rewrite action act1 insert_before 'HTTP.REQ.URL.PATH.SUFFIX(\'.\',0)' '"day."'
add rewrite action act2 insert_before 'HTTP.REQ.URL.PATH.SUFFIX(\'.\',0)' '"night."'
add rewrite policy poll1 'SYS.TIME.WITHIN(LOCAL 07h 00m,LOCAL 18h 59m)' act1
add rewrite policy poll2 'true' act2
bind rewrite global poll1 101
bind rewrite global poll2 102
```

## Redirecting to a New File Name (Invisible to the User)

If you rename a Web page, you can continue to support the old URL for backward compatibility while preventing users from recognizing that the page was renamed.

In the first two of the following examples, the base directory is /~quux/. The third example accommodates any base directory and the presence of query strings in the URL.

### Apache mod\_rewrite solution for managing a file name change in a fixed location

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ bar.html
```

### NetScaler solution for managing a file name change in a fixed location

```
add rewrite action act1 replace 'HTTP.REQ.URL.AFTER_STR("/~quux").SUBSTR("foo.html")' '"bar.'
add rewrite policy poll1 'HTTP.REQ.URL.ENDSWITH("/~quux/foo.html")' act1
bind rewrite global poll1 100
```

### NetScaler solution for managing a file name change regardless of the base directory or query strings in the URL

```
add rewrite action act1 replace 'HTTP.REQ.URL.PATH.SUFFIX(\'/\'',0)' '"bar.html"'
Add rewrite policy poll1 'HTTP.REQ.URL.PATH.CONTAINS("foo.html")' act1
Bind rewrite global poll1 100
```

## Redirecting to New File Name (User-Visible URL)

If you rename a Web page, you may want to continue to support the old URL for backward compatibility and allow users to see that the page was renamed by changing the URL that is displayed in the browser.

In the first two of the following examples, redirection occurs when the base directory is /~quux/. The third example accommodates any base directory and the presence of query strings in the URL.

### Apache mod\_rewrite solution for changing the file name and the URL displayed in the browser

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^old\.html$ new.html [R]
```

### NetScaler solution for changing the file name and the URL displayed in the browser

```
add responder action act1 redirect 'HTTP.REQ.URL.BEFORE_STR("foo.html")+"new.html"' -bypassS
add responder policy poll1 'HTTP.REQ.URL.ENDSWITH("/~quux/old.html")' act1
bind responder global poll1 100
```

### NetScaler solution for changing the file name and the URL displayed in the browser regardless of the base directory or query strings in the URL

```
add responder action act1 redirect 'HTTP.REQ.URL.PATH.BEFORE_STR("old.html")+"new.html"+HTTP
add responder policy poll1 'HTTP.REQ.URL.PATH.CONTAINS("old.html")' act1
bind responder global poll1 100
```

## Accommodating Browser Dependent Content



To accommodate browser-specific limitationsâ€”at least for important top-level pagesâ€”it is sometimes necessary to set restrictions on the browser type and version. For example, you might want to set a maximum version for the latest Netscape variants, a minimum version for Lynx browsers, and an average feature version for all others.

The following examples act on the HTTP header "User-Agent", such that if this header begins with "Mozilla/3", the page MyPage.html is rewritten to MyPage.NS.html. If the browser is "Lynx" or "Mozilla" version 1 or 2, the URL becomes MyPage.20.html. All other browsers receive page MyPage.32.html.

#### Apache mod\_rewrite solution for browser-specific settings

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/3.*
RewriteRule ^MyPage\.html$ MyPage.NS.html [L]
RewriteCond %{HTTP_USER_AGENT} ^Lynx/. * [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/[12]. *
RewriteRule ^MyPage\.html$ MyPage.20.html [L]
RewriteRule ^fMyPage\.html$ MyPage.32.html [L]
NetScaler solution for browser-specific settings
add patset pat1
bind patset pat1 Mozilla/1
bind Patset pat1 Mozilla/2
bind patset pat1 Lynx
bind Patset pat1 Mozilla/3
add rewrite action act1 insert_before 'HTTP.REQ.URL.SUFFIX' ' "NS." '
add rewrite action act2 insert_before 'HTTP.REQ.URL.SUFFIX' ' "20." '
add rewrite action act3 insert_before 'HTTP.REQ.URL.SUFFIX' ' "32." '
add rewrite policy pol1 'HTTP.REQ.HEADER( "User-Agent" ).STARTSWITH_INDEX( "pat1" ).EQ(4)' act1
add rewrite policy pol2 'HTTP.REQ.HEADER( "User-Agent" ).STARTSWITH_INDEX( "pat1" ).BETWEEN(1,3)
add rewrite policy pol3 '!HTTP.REQ.HEADER( "User-Agent" ).STARTSWITH_ANY( "pat1" )' act3
bind rewrite global pol1 101 END
bind rewrite global pol2 102 END
bind rewrite global pol3 103 END
```

## Blocking Access by Robots

You can block a robot from retrieving pages from a specific directory or a set of directories to ease up the traffic to and from these directories. You can restrict access based on the specific location or you can block requests based on information in User-Agent HTTP headers.

In the following examples, the Web location to be blocked is /~quux/foo/arc/, the IP addresses to be blocked are 123.45.67.8 and 123.45.67.9, and the robotâ€™s name is NameOfBadRobot.

#### Apache mod\_rewrite solution for blocking a path and a User-Agent header

```
RewriteCond %{HTTP_USER_AGENT} ^NameOfBadRobot.*
RewriteCond %{REMOTE_ADDR} ^123\.45\.67\.[8-9]$
RewriteRule ^/~quux/foo/arc/.+ - [F]
```

#### NetScaler solution for blocking a path and a User-Agent header

```
add responder action act1 respondwith '"HTTP/1.1 403 Forbidden\r\n\r\n"'
add responder policy pol1 'HTTP.REQ.HEADER( "User-Agent" ).STARTSWITH( "NameOfBadRobot" )&&CLIEN
bind responder global pol1 100
```

## Blocking Access to Inline Images

If you find people frequently going to your server to copy inline graphics for their own use (and generating unnecessary traffic), you may want to restrict the browserâ€™s ability to send an HTTP Referer header.

In the following example, the graphics are located in <http://www.quux-corp.de/~quux/>.

#### Apache mod\_rewrite solution for blocking access to an inline image

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://www.quux-corp.de/~quux/. *$
RewriteRule .*\.gif$ - [F]
```

#### NetScaler solution for blocking access to an inline image



```
add patset pat1
bind patset pat1 .gif
bind patset pat1 .jpeg
add responder action act1 respondwith '"HTTP/1.1 403 Forbidden\r\n\r\n"'
add responder policy poll1 '!HTTP.REQ.HEADER("Referer").EQ("") && !HTTP.REQ.HEADER("Referer")'
bind responder global poll1 100
```

## Creating Extensionless Links

To prevent users from knowing application or script details on the server side, you can hide file extensions from users. To do this, you may want to support extensionless links. You can achieve this behavior by using rewrite rules to add an extension to all requests, or to selectively add extensions to requests.

The first two of the following examples show adding an extension to all request URLs. In the last example, one of two file extensions is added. Note that in the last example, the `mod_rewrite` module can easily find the file extension because this module resides on the Web server. In contrast, the NetScaler must invoke an HTTP callout to check the extension of the requested file on the Web server. Based on the callout response, the NetScaler adds the `.html` or `.php` extension to the request URL.

Note: In the second NetScaler example, an HTTP callout is used to query a script named `file_check.cgi` hosted on the server. This script checks whether the argument that is provided in the callout is a valid file name.

### Apache `mod_rewrite` solution for adding a `.php` extension to all requests

```
RewriteRule ^/?([a-z]+)$ $1.php [L]
```

### NetScaler policy for adding a `.php` extension to all requests

```
add rewrite action act1 insert_after 'HTTP.REQ.URL' '".php"'
add rewrite policy poll1 'HTTP.REQ.URL.PATH.REGEX_MATCH(re#^/?([a-z]+)$#)' act1
bind rewrite global poll1 100
```

### Apache `mod_rewrite` solution for adding either `.html` or `.php` extensions to requests

```
RewriteCond %{REQUEST_FILENAME}.php -f
RewriteRule ^/?([a-zA-Z0-9]+)$ $1.php [L]
RewriteCond %{REQUEST_FILENAME}.html -f
RewriteRule ^/?([a-zA-Z0-9]+)$ $1.html [L]
```

### NetScaler policy for adding either `.html` or `.php` extensions to requests

```
add HTTPCallout Call_html
add HTTPCallout Call_php
set policy httpCallout Call_html -IPAddress 10.102.59.101 -port 80 -hostExpr '"10.102.59.101"'
set policy httpCallout Call_php -IPAddress 10.102.59.101 -port 80 -hostExpr '"10.102.59.101"'
add patset pat1
bind patset pat1 .html
bind patset pat1 .php
bind patset pat1 .asp
bind patset pat1 .cgi
add rewrite action act1 insert_after 'HTTP.REQ.URL.PATH' '".html"'
add rewrite action act2 insert_after 'HTTP.REQ.URL.PATH' '".php"'
add rewrite policy poll1 '!HTTP.REQ.URL.CONTAINS_ANY("pat1") && SYS.HTTP_CALLOUT(Call_html)'
add rewrite policy poll2 '!HTTP.REQ.URL.CONTAINS_ANY("pat1") && SYS.HTTP_CALLOUT(Call_php)' a
bind rewrite global poll1 100 END
bind rewrite global poll2 101 END
```

## Redirecting a Working URI to a New Format

Suppose that you have a set of working URLs that resemble the following:

```
/index.php?id=nnnn
```

To change these URLs to `/nnnn` and make sure that search engines update their indexes to the new URI format, you need to do the following:

- Redirect the old URIs to the new ones so that search engines update their indexes.

- o Rewrite the new URI back to the old one so that the index.php script runs correctly.

To accomplish this, you can insert marker code into the query string (making sure that the marker code is not seen by visitors), and then removing the marker code for the index.php script.

The following examples redirect from an old link to a new format only if a marker is not present in the query string. The link that uses the new format is re-written back to the old format, and a marker is added to the query string.

### Apache mod\_rewrite solution

```
RewriteCond %{QUERY_STRING} !marker
RewriteCond %{QUERY_STRING} id=([-a-zA-Z0-9_+]+)
RewriteRule ^/?index\.php$ %1? [R,L]
RewriteRule ^/?([-a-zA-Z0-9_+]+)$ index.php?marker&id=$1 [L]

NetScaler solution
add responder action act_redirect redirect 'HTTP.REQ.URL.PATH.BEFORE_STR("index.php")+HTTP.R
add responder policy pol_redirect '!HTTP.REQ.URL.QUERY.CONTAINS("marker")&& HTTP.REQ.URL.QUE
bind responder global pol_redirect 100 END
add rewrite action act1 replace 'HTTP.REQ.URL.PATH.SUFFIX('\',0)' '"index.phpmarker&id="+H
add rewrite policy poll '!HTTP.REQ.URL.QUERY.CONTAINS("marker")' act1
bind rewrite global poll 100 END
```

## Ensuring That a Secure Server Is Used for Selected Pages

To make sure that only secure servers are used for selected Web pages, you can use the following Apache mod\_rewrite code or NetScaler Responder policies.

### Apache mod\_rewrite solution

```
RewriteCond %{SERVER_PORT} !^443$
RewriteRule ^/?(page1|page2|page3|page4|page5)$ https://www.example.com/%1 [R,L]
```

### NetScaler solution using regular expressions

```
add responder action res_redirect redirect '"https://www.example.com"+HTTP.REQ.URL' -bypass
add responder policy pol_redirect '!CLIENT.TCP.DSTPORT.EQ(443)&&HTTP.REQ.URL.REGEX_MATCH(re/
bind responder global pol_redirect 100 END
```

### NetScaler solution using pattern sets

```
add patset pat1
bind patset pat1 page1
bind patset pat1 page2
bind patset pat1 page3
bind patset pat1 page4
bind patset pat1 page5
add responder action res_redirect redirect '"https://www.example.com"+HTTP.REQ.URL' -bypass
add responder policy pol_redirect '!CLIENT.TCP.DSTPORT.EQ(443)&&HTTP.REQ.URL.CONTAINS_ANY("p
bind responder global pol_redirect 100 END
```

## Rate Limiting

The rate limiting feature enables you to define the maximum load for a given network entity or virtual entity on the Citrix NetScaler appliance. The feature enables you to configure the appliance to monitor the rate of traffic associated with the entity and take preventive action, in real time, based on the traffic rate. This feature is particularly useful when the network is under attack from a hostile client that is sending the appliance a flood of requests. You can mitigate the risks that affect the availability of resources to clients, and you can improve the reliability of the network and the resources that the appliance manages.

You can monitor and control the rate of traffic that is associated with virtual and user-defined entities, including virtual servers, URLs, domains, and combinations of URLs and domains. You can throttle the rate of traffic if it is too high, base information caching on the traffic rate, and redirect traffic to a given load balancing virtual server if the traffic rate exceeds a predefined limit. You can apply rate-based monitoring to HTTP, TCP, and DNS requests.

To monitor the rate of traffic for a given scenario, you configure a *rate limit identifier*. A rate limit identifier specifies numeric thresholds such as the maximum number of requests or connections (of a particular type) that are permitted in a specified time period called a *time slice*.

Optionally, you can configure filters, known as *stream selectors*, and associate them with rate limit identifiers when you configure the identifiers. After you configure the optional stream selector and the limit identifier, you must invoke the limit identifier from a default syntax policy. You can invoke identifiers from any feature in which the identifier may be useful, including rewrite, responder, DNS, and integrated caching.

You can globally enable and disable SNMP traps for rate limit identifiers. Each trap contains cumulative data for the rate limit identifier's configured data collection interval (time slice), unless you specified multiple traps to be generated per time slice. For more information about configuring SNMP traps and managers, see "[SNMP](#)."

## Configuring a Stream Selector

A traffic stream selector is an optional filter for identifying an entity for which you want to throttle access. The selector is applied to a request or a response and selects data points (keys ) that can be analyzed by a rate stream identifier. These data points can be based on almost any characteristic of the traffic, including IP addresses, subnets, domain names, TCP or UDP identifiers, and particular strings or extensions in URLs.

A stream selector consists of individual default syntax expressions called selectlets. Each selectlet is a non-compound default syntax expression. A traffic stream selector can contain up to five non-compound expressions called selectlets. Each selectlet is considered to be in an AND relationship with the other expressions. Following are some examples of selectlets:

```
http.req.url  
http.res.body(1000>after_str(\"car_model\").before_str(\"made_in\"))  
"client.ip.src.subnet(24)"
```

The order in which you specify parameters is significant. For example, if you configure an IP address and a domain (in that order) in one selector, and then specify the domain and the IP address (in the reverse order) in another selector, the NetScaler considers these values to be unique. This can lead to the same transaction being counted twice. Also, if multiple policies invoke the same selector, the NetScaler, again, can count the same transaction more than once.

Note: If you modify an expression in a stream selector, you may get an error if any policy that invokes it is bound to a new policy label or bind point. For example, suppose that you create a stream selector named myStreamSelector1, invoke it from myLimitID1, and invoke the identifier from a DNS policy named dnsRateLimit1. If you change the expression in myStreamSelector1, you might receive an error when binding dnsRateLimit1 to a new bind point. The workaround is to modify these expressions before creating the policies that invoke them.

## To configure a traffic stream selector by using the command line interface

At the command prompt, type:

```
add stream selector <name> <rule> ...
```

### Example

```
> add stream selector myStreamSel HTTP.REQ.URL CLIENT.IP.SRC
```

## To configure a stream selector by using the configuration utility

Navigate to AppExpert > Rate Limiting > Selectors, click Add and specify the relevant details.

## Configuring a Traffic Rate Limit Identifier

A rate limit identifier returns a Boolean TRUE if the amount of traffic exceeds a numeric limit within a particular time interval. The rate limit identifier definition can optionally include a stream selector. When you include a limit identifier in the compound default syntax expression in a policy rule, if you do not specify a stream selector, the limit identifier is applied to all the requests or responses that are identified by the compound expression.

Note: The maximum length for storing string results of selectors (for example, HTTP.REQ.URL) is 60 characters. If the string (for example, URL) is 1000 characters long, of which 50 characters are enough to uniquely identify a string, use an expression to extract only the required 50 characters.

### To configure a traffic limit identifier from the command line interface

At the command prompt, type:

```
add ns limitIdentifier <limitIdentifier> -threshold <positive_integer> -timeSlice <positive_integer> -mode <mode> -limitType ( BURSTY | SMOOTH ) -selectorName <string> -maxBandwidth <positive_integer> -trapsInTimeSlice <positive_integer>
```

#### Example

Configuring traffic rate limit identifier in BURSTY mode:

```
> add ns limitIdentifier 100_request_limit -threshold 100 -timeSlice 1000 -mode REQUEST_RATE
```

Configuring traffic rate limit identifier in SMOOTH mode:

```
> add ns limitIdentifier limit_req -mode request_rate -limitType smooth -timeslice 1000 -Thr
```

### To configure a traffic limit identifier by using the configuration utility

Navigate to AppExpert > Rate Limiting > Limit Identifiers, click Add and specify the relevant details.

## Configuring and Binding a Traffic Rate Policy

You implement rate-based application behavior by configuring a policy in an appropriate NetScaler feature. The feature must support default syntax policies. The policy expression must contain the following expression prefix to enable the feature to analyze the traffic rate:

```
sys.check_limit(<limit_identifier>)
```

Where `limit_identifier` is the name of a limit identifier.

The policy expression must be a compound expression that contains at least two components:

- An expression that identifies traffic to which the rate limit identifier is applied. For example:

```
http.req.url.contains("my_aspx.aspx").
```

- An expression that identifies a rate limit identifier, for example, `sys.check_limit("my_limit_identifier")`. This must be the last expression in the policy expression.

## To configure a rate-based policy by using the command line interface

At the command prompt, type the following command to configure a rate-based policy and verify the configuration:

```
add cache|dns|rewrite|responder policy <policy_name> -rule expression && sys.check_limit("<LimitIdentifierName>")  
[<feature-specific information>]
```

Following is a complete example of a rate-based policy rule. Note that this example assumes that you have configured the responder action, `send_direct_url`, that is associated with the policy. Note that the `sys.check_limit` parameter must be the last element of the policy expression:

```
add responder policy responder_threshold_policy "http.req.url.contains(\"myindex.html\") &&
```

For information about binding a policy globally or to a virtual server, see ["Binding Default Syntax Policies."](#)

## To configure a rate-based policy by using the configuration utility

1. In the navigation pane, expand the feature in which you want to configure a policy (for example, Integrated Caching, Rewrite, or Responder), and then click Policies.
2. In the details pane, click Add. In Name, enter a unique name for the policy.
3. Under Expression, enter the policy rule, and make sure that you include the `sys.check_limit` parameter as the final component of the expression. For example:

```
http.req.url.contains("my_aspx.aspx") && sys.check_limit("my_limit_identifier")
```

4. Enter feature-specific information about the policy.

For example, you may be required to associate the policy with an action or a profile. For more information, see the feature-specific documentation.

5. Click Create, and then click Close.
6. Click Save.

## Viewing the Traffic Rate

If traffic through one or more virtual servers matches a rate-based policy, you can view the rate of this traffic. The rate statistics are maintained in the limit identifier that you named in the rule for the rate-based policy. If more than one policy uses the same limit identifier, you can view the traffic rate as defined by hits to all of the policies that use the particular limit identifier.

### To view the traffic rate by using the command line interface

At the command prompt, type the following command to view the traffic rate:

```
show ns limitSessions <limitIdentifier>
```

#### Example

```
sh limitSession myLimitSession
```

### To view the traffic rate by using the configuration utility

1. Navigate to AppExpert > Rate Limiting > Limit Identifiers.
2. Select a limit identifier whose traffic rate you want to view.
3. Click the Show Sessions button. If traffic through one or more virtual servers has matched a rate limiting policy that uses this limit identifier (and the hits are within the configured time slice for this identifier), the Session Details dialog box appears. Otherwise, you receive a “No session exists” message.

## Testing a Rate-Based Policy

To test a rate-based policy, you can send traffic to any virtual server to which a rate-based policy is bound.

### Task overview: Testing a rate-based policy

1. Configure a stream selector (optional) and a rate limit identifier (required). For example:

```
add stream selector sel_subnet Q.URL "CLIENT.IP.SRC.SUBNET(24)"
add ns limitIdentifier k_subnet -Threshold 4 -timeSlice 3600 -mode REQUEST_RATE -limit
```

2. Configure the action that you want to associate with the policy that uses the rate limit identifier. For example:

```
add responder action resp_redirect redirect "\"http://response_site.com/\""
```

3. Configure a policy that uses the `sys.check_limit` expression prefix to call the rate limit identifier. For example, the policy can apply a rate limit identifier to all requests arriving from a particular subnet, as follows:

```
add responder policy resp_subnet "SYS.CHECK_LIMIT(\"k_subnet\")" resp_redirect
```

4. Bind the policy globally or to a virtual server. For example:

```
bind responder global resp_subnet 6 END -type DEFAULT
```

5. In a browser address bar, send a test HTTP query to a virtual server. For example:

```
http://<IP of a vserver>/testsite/test.txt
```

6. At the NetScaler command prompt, type:

```
show ns limitSessions <limitIdentifier>
```

#### Example

```
> sh limitSession k_subnet
1)      Time Remaining:      98 secs  Hits: 2                      Action Taken: (
      Total Hash:      1718618  Hash String: /test.txt
      IPs gathered:
          1) 10.217.253.0
      Active Transactions: 0
Done
>
```

7. Repeat the query and check the limit identifier statistics again to verify that the statistics are being updated correctly.



## Examples of Rate-Based Policies

The following table shows examples of rate-based policies.

Table 1. Examples of Rate-Based Policies

Purpose	Example
Limit the number of requests per second from a URL	<pre>add stream selector ipStreamSelector http.req.url "client.ip.src" add ns limitIdentifier ipLimitIdentifier -threshold 4 -timeSlice 1000     -mode request_rate -limitType smooth -selectorName ipStreamSelector  add responder action myWebSiteRedirectAction redirect     "\"http://www.mycompany.com/\""  add responder policy ipLimitResponderPolicy "http.req.url.contains(\'     &amp;&amp; sys.check_limit(\'"ipLimitIdentifier\')" myWebSiteRedirectAction  bind responder global ipLimitResponderPolicy 100 END -type default</pre>
Cache a response if the request URL rate exceeds 5 per 20000 milliseconds	<pre>add stream selector cacheStreamSelector http.req.url add ns limitIdentifier cacheRateLimitIdentifier -threshold 5 -timeSlice 20000     -selectorName cacheStreamSelector  add cache policy cacheRateLimitPolicy -rule "http.req.method.eq(get)     &amp;&amp; sys.check_limit(\'"cacheRateLimitIdentifier\')" -action cache  bind cache global cacheRateLimitPolicy -priority 10</pre>
Drop a connection on the basis of cookies received in requests from www.yourcompany.com if the requests exceed the rate limit	<pre>add stream selector reqCookieStreamSelector "http.req.cookie     .value(\'"mycookie\')" "client.ip.src.subnet(24)"  add ns limitIdentifier myLimitIdentifier -Threshold 2 -timeSlice 3000     -selectorName reqCookieStreamSelector  add responder action sendRedirectUrl redirect '\http://www.mycompany     + http.req.url' -bypassSafetyCheck YES  add responder policy rateLimitCookiePolicy     "http.req.url.contains(\'"www.yourcompany.com\')     &amp;&amp; sys.check_limit(\'"myLimitIdentifier\')" sendRedirectUrl</pre>
Drop a DNS packet if the requests from a particular client IP address and DNS domain exceed the rate limit	<pre>add stream selector dropDNSStreamSelector client.udp.dns.domain client.udp.dns.src add ns limitIdentifier dropDNSRateIdentifier -timeSlice 20000 -mode request_rate     -selectorName dropDNSStreamSelector -maxBandwidth 1 -trapsintimeslice  add dns policy dnsDropOnClientRatePolicy "sys.check_limit     (\dropDNSRateIdentifier\)" -drop yes</pre>
Limit the number of HTTP requests that arrive from the same subnet (with a subnet mask of 32) and that have the same destination IP address.	<pre>add stream selector ipv6_sel "CLIENT.IPv6.src.subnet(32)" CLIENT.IPv6.dest add ns limitIdentifier ipv6_id -timeSlice 20000 -selectorName ipv6_sel add lb vserver ipv6_vip HTTP 3ffe::209 80 -persistenceType NONE -clientTimeout add responder action redirect_page redirect "\"http://redirectpage.com/" add responder policy ipv6_resp_pol "SYS.CHECK_LIMIT(\'"ipv6_id\')" redirect_page bind responder global ipv6_resp_pol 5 END -type DEFAULT</pre>

## Sample Use Cases for Rate-Based Policies

The following scenarios describe two uses of rate-based policies in global server load balancing (GSLB):

- The first scenario describes the use of a rate-based policy that sends traffic to a new data center if the rate of DNS requests exceed 1000 per second.
- In the second scenario, if more than five DNS requests arrive for a local DNS (LDNS) client within a particular period, the additional requests are dropped.

## Redirecting Traffic on the Basis of Traffic Rate

In this scenario, you configure a proximity-based load balancing method, and a rate-limiting policy that identifies DNS requests for a particular region. In the rate-limiting policy, you specify a threshold of 1000 DNS requests per second. A DNS policy applies the rate limiting policy to DNS requests for the region "Europe.GB.17.London.UK-East.ISP-UK." In the DNS policy, DNS requests that exceed the rate limiting threshold, starting with request 1001 and continuing to the end of the one-second interval, are to be forwarded to the IP addresses that are associated with the region "North America.US.TX.Dallas.US-East.ISP-US."

The following configuration demonstrates this scenario:

```
add stream selector DNSSelector1 client.udp.dns.domain
add ns limitIdentifier DNSLimitIdentifier1 -threshold 5 -timeSlice 1000 -selectorName DNSSel
add dns policy DNSLimitPolicy1 "client.ip.src.matches_location(\"Europe.GB.17.London.*.*\")
sys.check_limit(\"DNSLimitIdentifier1\")" -preferredLocation "North America.US.TX.Dallas.*.*
bind dns global DNSLimitPolicy1 5
```

## Dropping DNS Requests on the Basis of Traffic Rate

In the following example of global server load balancing, you configure a rate limiting policy that permits a maximum of five DNS requests in a particular interval, per domain, to be directed to an LDNS client for resolution. Any requests that exceed this rate are dropped. This type of policy can help protect the NetScaler from resource exploitation. For example, in this scenario, if the time to live (TTL) for a connection is five seconds, this policy prevents the LDNS from requerying a domain. Instead, it uses data that is cached on the NetScaler.

```
add stream selector LDNSSelector1 client.udp.dns.domain client.ip.src
add ns limitIdentifier LDNSLimitIdentifier1 -threshold 5 -timeSlice 1000 -selectorName LDNSS
add dns policy LDNSPolicy1 "client.udp.dns.domain.contains(\".\") && sys.check_limit(\"LDNSL
bind dns global LDNSPolicy1 6
show gslb vserver gvip
gvip - HTTP      State: UP
Last state change was at Mon Sep  8 11:50:48 2008 (+711 ms)
Time since last state change: 1 days, 02:55:08.830
Configured Method: STATICPROXIMITY
BackupMethod: ROUNDROBIN
No. of Bound Services : 3 (Total)          3 (Active)
Persistence: NONE      Persistence ID: 100
Disable Primary Vserver on Down: DISABLED      Site Persistence: NONE
Backup Session Timeout: 0
Empty Down Response: DISABLED
Multi IP Response: DISABLED Dynamic Weights: DISABLED
Cname Flag: DISABLED
Effective State Considered: NONE
1)      sitell_svc(10.100.00.00: 80)- HTTP State: UP      Weight: 1
Dynamic Weight: 0      Cumulative Weight: 1
Effective State: UP
Threshold : BELOW
Location: Europe.GB.17.London.UK-East.ISP-UK
2)      sitel2_svc(10.101.00.100: 80)- HTTP State: UP      Weight: 1
Dynamic Weight: 0      Cumulative Weight: 1
Effective State: UP
Threshold : BELOW
Location: North America.US.TX.Dallas.US-East.ISP-US
3)      sitel3_svc(10.102.00.200: 80)- HTTP State: UP      Weight: 1
Dynamic Weight: 0      Cumulative Weight: 1
Effective State: UP
Threshold : BELOW
Location: North America.US.NJ.Salem.US-Mid.ISP-US
```

1) www.gslbindia.com TTL: 5 secn  
Cookie Timeout: 0 min Site domain TTL: 3600 sec  
Done

## Rate Limiting for Traffic Domains

You can configure rate limiting for traffic domains. The following expression in the NetScaler expressions language for identifies traffic associated with traffic domains.

- `client.traffic_domain.id`

You can configure rate limiting for traffic associated with a particular traffic domain, a set of traffic domains, or all traffic domains.

For configuring rate limiting for traffic domains, you perform the following steps on a NetScaler appliance by using the configuration utility or the NetScaler command line:

1. Configure a stream selector that uses the `client.traffic_domain.id` expression for identifying the traffic, associated with traffic domains, to be rate limited.
2. Configure a rate limit identifier that specifies parameters such as maximum threshold for traffic to be rate limited. You also associate a stream selector to the rate limiter in this step.
3. Configure an action that you want to associate with the policy that uses the rate limit identifier.
4. Configure a policy that uses the `sys.check_limit` expression prefix to call the rate limit identifier, and associate the action with this policy.
5. Bind the policy globally.

Consider an example in which two traffic domains, with IDs 10 and 20, are configured on NetScaler ADC NS1. On traffic domain 10, LB1-TD-1 is configured to load balance servers S1 and S2; LB2-TD1 is configured to load balance servers S3 and S4.

On traffic domain 20, LB1-TD-2 is configured to load balance servers S5 and S6; LB2-TD2 is configured to load balance servers S7 and S8.

The following table lists some examples of rate limiting policies for traffic domains in the example setup.

Purpose	CLI commands
Limit the number of requests to 10 per second for each of the traffic domains.	<pre>add stream selector tdratelimit-1 CLIENT.TRAFFIC_DOMAIN.ID add ns limitIdentifier limitidf-1 -threshold 10 -selectorName tdratelimit-1 -trapsInTimeSlice 0 add responder policy ratelimit-pol "sys.check_limit(\"limitidf-1\")" DROP bind responder global ratelimit-pol 1</pre>
Limit the number of requests to 5 per client per second for each of the traffic domains.	<pre>add stream selector tdandclientip CLIENT.IP.SRC,CLIENT.TRAFFIC_DOMAIN.ID add ns limitIdentifier td_limitidf -threshold 5 -selectorName tdandclientip -trapsInTimeSlice 5 add responder policy tdratelimit-pol "sys.check_limit(\"td_limitidf\")" DROP bind responder global tdratelimit-pol 2</pre>
Limit the number of requests sent for a particular traffic domain (for example traffic domain 10) to 30 requests every 3 seconds.	<pre>add stream selector tdratelimit CLIENT.TRAFFIC_DOMAIN.ID add ns limitIdentifier td10_limitidf -threshold 30 -timeSlice 3000 -selectorName tdratelimit -trapsInTimeSlice 5 add responder policy td10ratelimit "client.traffic_domain.id==10 &amp;&amp; sys.check_limit(\"td10_limitidf\")" DROP bind responder global td10ratelimit 3</pre>
Limit the number of connections to 5 per client per second for a particular traffic domain (for example traffic domain 20).	<pre>add stream selector tdandclientip CLIENT.IP.SRC CLIENT.TRAFFIC_DOMAIN.ID add ns limitIdentifier td20_limitidf -threshold 5 -mode CONNECTION -selectorName tdandclientip -trapsInTimeSlice 5 add responder policy td20_ratelimit "client.traffic_domain.id==20 &amp;&amp; sys.check_limit(\"td20_limitidf\")" DROP bind responder global td20_ratelimit 4</pre>

## Responder

Today's complex Web configurations often require different responses to HTTP requests that appear, on the surface, to be similar. When users request a Web site's home page, you may want to provide a different home page depending on where each user is located, which browser the user is using, or which language(s) the browser accepts and the order of preference. You might want to break the connection immediately if the request is coming from an IP range that has been generating DDoS attacks or initiating hacking attempts.

With the Responder feature, responses can be based on who sends the request, where it is sent from, and other criteria with security and system management implications. The feature is simple and quick to use. By avoiding the invocation of more complex features, it reduces CPU cycles and time spent in handling requests that do not require complex processing.

For handling sensitive data such as financial information, if you want to ensure that the client uses a secure connection to browse a site, you can redirect the request to secure connection by using https:// instead of http://.

To use the Responder feature, do the following;

- Enable the Responder feature on the NetScaler.
- Configure responder actions. The action can be to generate a custom response, redirect a request to a different Web page, or reset a connection.
- Configure responder policies. The policy determines the requests (traffic) on which an action has to be taken.
- Bind each policy to a bind point put it into effect. A bind point refers to an entity at which NetScaler examines the traffic to see if it matches a policy. For example, a bind point can be a load balancing virtual server.

You can specify a default action for requests that do not match any policy, and you can bypass the safety check for actions that would otherwise generate error messages.

The Rewrite feature of NetScaler helps in rewriting some information in the requests or responses handled by NetScaler. The following section shows some differences between the two features.

## Comparison between Rewrite and Responder options

The main difference between the rewrite feature and the responder feature is as follows:

Responder cannot be used for response or server-based expressions. Responder can be used only for the following scenarios depending on client parameters:

- Redirecting a http request to new Web sites or Web pages
- Responding with some custom response
- Dropping or resetting a connection at request level

In case of a responder policy, the NetScaler examines the request from the client, takes action according to the applicable policies, sends the response to the client, and closes the connection with the client.

In case of a rewrite policy, the NetScaler examines the request from the client or response from the server, takes action according to the applicable policies, and forwards the traffic to the client or the server.

In general, it is recommended to use responder if you want the NetScaler to reset or drop a connection based on a client or request-based parameter. Use responder to redirect traffic, or respond with custom messages. Use rewrite for manipulating data on HTTP requests and responses.

## Enabling the Responder Feature

To use the Responder feature, you must first enable it.

### To enable the responder feature by using the command line interface

At the command prompt, type the following commands to enable the responder feature and verify the configuration:

- `enable ns feature <feature>`
- `show ns feature`

#### Example

```
enable ns feature Responder
Done
> show ns feature
```

	Feature -----	Acronym -----	Status -----
1)	Web Logging	WL	ON
2)	Surge Protection	SP	ON
.			
.			
.			
22)	<b>Responder</b>	<b>RESPONDER</b>	<b>ON</b>
23)	HTML Injection	HTMLInjection	ON
24)	NetScaler Push	push	OFF

```
Done
>
```

### To enable the responder feature by using the configuration utility

1. In the navigation pane, expand System, and then click Settings.
2. In the details pane, under Modes and Features, click Change advanced features.
3. In the Configure Advanced Features dialog box, select the Responder check box, and then click OK.
4. In the Enable/Disable Feature(s)? dialog box, click YES. A message appears in the status bar, stating that the feature has been enabled.

## Configuring a Responder Action

After enabling the responder feature, you must configure one or more actions for handling requests. The responder supports the following types of actions:

### Respond with

Sends the response defined by the Target expression without forwarding the request to a web server. (The NetScaler appliance substitutes for and acts as a web server.) Use this type of action to manually define a simple HTML-based response. Normally the text for a Respond with action consists of a web server error code and brief HTML page.

### Respond with SQL OK

Sends the designated SQL OK response defined by the Target expression. Use this type of action to send an SQL OK response to an SQL query.

### Respond with SQL Error

Sends the designated SQL Error response defined by the Target expression. Use this type of action to send an SQL Error response to an SQL query.

### Respond with HTML page

Sends the designated HTML page as the response. You can choose from a drop-down list of HTML pages that were previously uploaded, or upload a new HTML page. Use this type of action to send an imported HTML page as the response.

### Redirect

Redirects the request to a different web page or web server. A Redirect action can redirect requests originally sent to a "dummy" web site that exists in DNS, but for which there is no actual web server, to an actual web site. It can also redirect search requests to an appropriate URL. Normally, the redirection target for a Redirect action consists of a complete URL.

## To configure a responder action by using the command line interface

At the command prompt, type the following commands to configure a responder action and verify the configuration:

- add responder action <name> <type> <target> [-bypassSafetyCheck (YES | NO) ]
- show responder action

### Example

To create a responder action that displays a "Not Found" error page for URLs that do not exist:

```
add responder action act404Error respondWith "HTTP/1.1 404 Not Found\r\n\r\n" + "HTTP.REQ.URL
Done
> show responder action

1)      Name: act404Error
        Operation: respondwith
        Target: "HTTP/1.1 404 Not Found

"+ "HTTP.REQ.URL.HTTP_URL_SAFE" + "does not exist on the web server."
        BypassSafetyCheck : NO
        Hits: 0
        Undef Hits: 0
        Action Reference Count: 0

Done
```

To create a responder action that displays a "Not Found" error page for URLs that do not exist:

```
add responder action act404Error respondWith "HTTP/1.1 404 Not Found\r\n\r\n" + "HTTP.REQ.URL
Done
> show responder action

1)      Name: act404Error
        Operation: respondwith
        Target: "HTTP/1.1 404 Not Found

"+ "HTTP.REQ.URL.HTTP_URL_SAFE" + "does not exist on the web server."
        BypassSafetyCheck : NO
        Hits: 0
```

Undef Hits: 0  
Action Reference Count: 0

Done

## To modify an existing responder action by using the command line interface

At the command prompt, type the following command to modify an existing responder action and verify the configuration:

- o set responder action <name> -target <string> [-bypassSafetyCheck ( YES | NO )]
- o show responder action

### Example

```
set responder action act404Error -target 'HTTP/1.1 404 Not Found\r\n\r\n'+ "HTTP.REQ.URL.H
Done
> show responder action

1)      Name: act404Error
        Operation: respondwith
        Target: "HTTP/1.1 404 Not Found

"+ "HTTP.REQ.URL.HTTP_URL_SAFE" + "does not exist on the web server."
        BypassSafetyCheck : NO
        Hits: 0
        Undef Hits: 0
        Action Reference Count: 0

Done
```

## To remove a responder action by using the command line interface

At the command prompt, type the following command to remove a responder action and verify the configuration:

- o rm responder action <name>
- o show responder action

### Example

```
rm responder action act404Error
Done

> show responder action
Done
```

## To configure a responder action by using the configuration utility

1. Navigate to AppExpert > Responder > Actions.
2. In the details pane, do one of the following:
  - o To create a new action, click Add.
  - o To modify an existing action, select the action, and then click Open.
3. Click Create or OK, depending on whether you are creating a new action or modifying an existing action.
4. Click Close. A message appears in the status bar, stating that the feature has been enabled.
5. To delete a responder action, select the action, and then click Remove. A message appears in the status bar, stating that the feature has been disabled.

## To add an expression by using the Add Expression dialog box

1. In the Create Responder Action or Configure Responder Action dialog box, click Add.
2. In the Add Expression dialog box, in the first list box choose the first term for your expression.

### HTTP

The HTTP protocol. Choose this if you want to examine some aspect of the request that pertains to the HTTP protocol.

### SYS

The protected web site(s). Choose this if you want to examine some aspect of the request that pertains to the recipient of the request.

### CLIENT

The computer that sent the request. Choose this if you want to examine some aspect of the sender of the request.

### ANALYTICS



- The analytics data associated with the request. Choose this if you want to examine request metadata.
- SIP      A SIP request. Choose this if you want to examine some aspect of a SIP request.

When you make your choice, the rightmost list box lists appropriate terms for the next part of your expression.

3. In the second list box, choose the second term for your expression. The choices depend upon which choice you made in the previous step, and are appropriate to the context. After you make your second choice, the Help window below the Construct Expression window (which was blank) displays help describing the purpose and use of the term you just chose.
4. Continue choosing terms from the list boxes that appear to the right of the previous list box, or typing strings or numbers in the text boxes that appear to prompt you to enter a value, until your expression is finished.

## Configuring the Global HTTP Action

You can configure the global HTTP action to invoke a responder action when an HTTP request times out. To configure this feature, you must first create the responder action that you want to invoke. Then, you configure the global HTTP timeout action to respond to a timeout with that responder action.

### To configure the global HTTP action by using the command line interface

At the command prompt, type the following command:

- o `set ns httpProfile -reqTimeoutAction <responder action name>`
- o `save ns config`

For `<responder action name>`, substitute the name of the responder action.

## Configuring a Responder Policy

After you configure a responder action, you must next configure a responder policy to select the requests to which the NetScaler appliance should respond. A responder policy is based on a rule, which consists of one or more expressions. The rule is associated with an action, which is performed if a request matches the rule.

Note: For creating and managing responder policies, the configuration utility provides assistance that is not available at the NetScaler command prompt.

### To configure a responder policy by using the command line interface

At the command prompt, type the following command to add a new responder policy and verify the configuration:

- o add responder policy <name> <expression> <action> [<undefaction>]-appFlowaction<actionName>
- o show responder policy <name>

#### Example

```
> add responder policy policyThree "CLIENT.IP.SRC.IN_SUBNET(222.222.0.0/16)" RESET
Done
> show responder policy policyThree

Name: policyThree
Rule: CLIENT.IP.SRC.IN_SUBNET(222.222.0.0/16)
Responder Action: RESET
UndefAction: Use Global
Hits: 0
Undef Hits: 0
Done
```

### To modify an existing responder policy by using the command line interface

At the command prompt, type the following command to modify an existing responder policy and verify the configuration:

- o set responder policy <name> [-rule <expression>] [-action <string>] [-undefAction <string>]
- o show responder policy <name>

### To remove a responder policy by using the command line interface

At the command prompt, type the following command to remove a responder policy and verify the configuration:

- o rm responder policy <name>
- o show responder policy

#### Example

```
>rm responder policy pol404Error
Done
> show responder policy
Done
```

### To configure a responder policy by using the configuration utility

1. Navigate to AppExpert > Responder > Policies.
2. In the details pane, do one of the following:
  - o To create a new policy, click Add.
  - o To modify an existing policy, select the policy, and then click Open.
3. Click Create or OK, depending on whether you are creating a new policy or modifying an existing policy.
4. Click Close. A message appears in the status bar, stating that the feature has been configured.

## Binding a Responder Policy

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

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

In the NetScaler operating system, policy priorities work in reverse order—the higher the number, the lower the priority. For example, if you have three policies with priorities of 10, 100, and 1000, the policy assigned a priority of 10 is performed first, then the policy assigned a priority of 100, and finally the policy assigned an order of 1000. The responder feature implements only the first policy that a request matches, not any additional policies that it might also match, so policy priority is important for getting the results you intend.

You can leave yourself plenty of room to add other policies in any order, and still set them to evaluate in the order you want, by setting priorities with intervals of 50 or 100 between each policy when you globally bind it. You can then add additional policies at any time without having to reassign the priority of an existing policy.

For additional information about binding policies on the NetScaler, see ["Policies and Expressions."](#)

Note: Responder policies cannot be bound to TCP-based virtual servers.

## To globally bind a responder policy by using the command line interface

At the command prompt, type the following command to globally bind a responder policy and verify the configuration:

- `bind responder global <policyName> <priority> [<gotoPriorityExpression [-type <type>]] [-invoke (<labelType> <labelName>)]`
- `show responder global`

### Example

```
> bind responder global poliError 100
Done
> show responder global
1)      Global bindpoint: REQ_DEFAULT
        Number of bound policies: 1

Done
```

## To bind responder policy to a specific virtual server by using the command line interface

At the command prompt, type the following command to bind responder policy to a specific virtual server and verify the configuration:

```
bind lb vserver <name> -policyname <policy_name> -priority <priority>
```

### Example

```
> bind lb vserver vs-loadbal -policyName policyTwo -priority 100
Done
> show lb vserver
1)      vs-loadbal (10.102.29.20:80) - HTTP      Type: ADDRESS
        State: OUT OF SERVICE
        Last state change was at Wed Aug 19 09:05:47 2009 (+211 ms)
        Time since last state change: 2 days, 00:58:03.260
        Effective State: DOWN
        Client Idle Timeout: 180 sec
        Down state flush: ENABLED
        Disable Primary Vserver On Down : DISABLED
        Port Rewrite : DISABLED
        No. of Bound Services : 0 (Total)        0 (Active)
        Configured Method: LEASTCONNECTION
        Mode: IP
        Persistence: NONE
        Vserver IP and Port insertion: OFF
        Push: DISABLED  Push VServer:
        Push Multi Clients: NO
```

```
2)      Push Label Rule: none
        vs-cont-sw (0.0.0.0:0) - TCP      Type: ADDRESS
        State: DOWN
        Last state change was at Wed Aug 19 10:03:46 2009 (+213 ms)
        Time since last state change: 2 days, 00:00:04.260
        Effective State: DOWN
        Client Idle Timeout: 9000 sec
        Down state flush: ENABLED
        Disable Primary Vserver On Down : DISABLED
        No. of Bound Services : 0 (Total)      0 (Active)
        Configured Method: LEASTCONNECTION
        Mode: IP
        Persistence: NONE
        Connection Failover: DISABLED
```

Done

## To globally bind a responder policy by using the configuration utility

1. Navigate to AppExpert > Responder > Policies.
2. On the Responder Policies page, select a responder policy, and then click Policy Manager.
3. In the Responder Policy Manager dialog box Bind Points menu, select Default Global.
4. Click Insert Policy to insert a new row and display a drop-down list of all unbound responder policies.
5. Click one of the policies on the list. That policy is inserted into the list of globally bound responder policies.
6. Click Apply Changes.
7. Click Close. A message appears in the status bar, stating that the configuration has been successfully completed.

## To bind a responder policy to a specific virtual server by using the configuration utility

1. Navigate to Traffic Management > Load Balancing > Virtual Servers.
2. On the Load Balancing Virtual Servers page, select the virtual server to which you want to bind the responder policy, and then click Open.
3. In the Configure Virtual Server (Load Balancing) dialog box, select the Policies tab, which displays a list of all policies configured on your NetScaler appliance.
4. Select the check box next to the name of the policy you want to bind to this virtual server.
5. Click OK. A message appears in the status bar, stating that the configuration has been successfully completed.

## Setting the Responder Default Action

The NetScaler appliance generates an undefined event (UNDEF event) when a request does not match a responder policy, and then carries out the default action assigned to undefined events. By default, that action is to forward the request to the next feature without changing it. This default behavior is normally what you want; it ensures that requests that do not require special handling by a specific responder action are sent to your Web servers and clients receive access to the content that they requested.

If the Web site(s) your NetScaler appliance protects receive a significant number of invalid or malicious requests, however, you may want to change the default action to either reset the client connection or drop the request. In this type of configuration, you would write one or more responder policies that would match any legitimate requests, and simply redirect those requests to their original destinations. Your NetScaler appliance would then block any other requests as specified by the default action you configured.

You can assign any one of the following actions to an undefined event:

### NOOP

The NOOP action aborts responder processing but does not alter the packet flow. This means that the appliance continues to process requests that do not match any responder policy, and eventually forwards them to the requested URL unless another feature intervenes and blocks or redirects the request. This action is appropriate for normal requests to your Web servers and is the default setting.

### RESET

If the undefined action is set to RESET, the appliance resets the client connection, informing the client that it must re-establish its session with the Web server. This action is appropriate for repeat requests for Web pages that do not exist, or for connections that might be attempts to hack or probe your protected Web site(s).

### DROP

If the undefined action is set to DROP, the appliance silently drops the request without responding to the client in any way. This action is appropriate for requests that appear to be part of a DDoS attack or other sustained attack on your servers.

Note: UNDEF events are triggered only for client requests. No UNDEF events are triggered for responses.

## To set the undefined action by using the command line interface

At the command prompt, type the following command to set the undefined action and verify the configuration:

- set responder param -undefAction (RESET|DROP|NOOP)
- show responder param

### Example

```
>set responder param -undefAction RESET
Done
> show responder param
    Action Name: RESET
Done
>
```

## To set the undefined action by using the configuration utility

1. Navigate to AppExpert > Responder, and then under Settings, click the Change Responder Settings link.
2. In the Set Responder Params dialog box, under Global Undefined-Result Action, select NOOP, RESET, or DROP.
3. Click OK. A message appears in the status bar, stating that the Responder Parameters have been configured.

## Responder Action and Policy Examples

Responder actions and policies are powerful and complex, but you can get started with relatively simple applications. For typical examples, see ["Example: Blocking Access from Specified IPs"](#) and ["Example: Redirecting a Client to a new URL."](#)

### Example: Blocking Access from Specified IPs

The following procedures block access to your protected Web site(s) by clients originating from the CIDR 222.222.0.0/16. The responder sends an error message stating that the client is not authorized to access the URL requested.

#### To block access by using the command line interface

At the command prompt, type the following commands to block access:

- o add responder action act\_unauthorized respondwith "HTTP/1.1 200 OK\r\n\r\n" + "Client: " + CLIENT.IP.SRC + " is not authorized to access URL:" + "HTTP.REQ.URL.HTTP\_URL\_SAFE"
- o add responder policy pol\_un "CLIENT.IP.SRC.IN\_SUBNET (222.222.0.0/16)" act\_unauthorized
- o bind responder global pol\_un 10

#### To block access by using the configuration utility

1. In the navigation pane, expand Responder, and then click Actions.
2. In the details pane, click Add.
3. In the Create Responder Action dialog box, do the following:
  - a. In the Name text box, type act\_unauthorized.
  - b. Under Type, select Respond with.
  - c. In the Target text area, type the following string: "HTTP/1.1 200 OK\r\n\r\n" + "Client: " + CLIENT.IP.SRC + " is not authorized to access URL:" + HTTP.REQ.URL.HTTP\_URL\_SAFE
  - d. Click Create, and then click Close.

The responder action you configured, named act\_unauthorized, now appears in the Responder Actions page.

4. In the navigation pane, click Policies.
5. In the details pane, click Add.
6. In the Create Responder Policy dialog box, do the following:
  - a. In the Name text box, type pol\_unauthorized.
  - b. Under Action, select act\_unauthorized.
  - c. In the Expression window, type the following rule: CLIENT.IP.SRC.IN\_SUBNET(222.222.0.0/16)
  - d. Click Create, then click Close.

The responder policy you configured, named pol\_unauthorized, now appears in the Responder Policies page.

7. Globally bind your new policy, pol\_unauthorized, as described in ["Binding a Responder Policy."](#)

### Example: Redirecting a Client to a new URL

The following procedures redirect clients who access your protected Web site(s) from within the CIDR 222.222.0.0/16 to a specified URL.

#### To redirect clients by using the command line interface

At the command prompt, type the following commands to redirect clients and verify the configuration:

- o add responder action act\_redirect redirect "'http://www.example.com/404.html'"
- o show responder action act\_redirect
- o add responder policy pol\_redirect "CLIENT.IP.SRC.IN\_SUBNET(222.222.0.0/16)" act\_redirect
- o show responder policy pol\_redirect
- o bind responder global pol\_redirect 10

#### Example

```
> add responder action act_redirect redirect ' " http ://www.example.com/404.html " '
> add responder policy pol_redirect "CLIENT.IP.SRC.IN_SUBNET(222.222.0.0/16)" act_redirect
```

#### To redirect clients by using the configuration utility

1. Navigate to AppExpert > Responder > Actions.
2. In the details pane, click Add.
3. In the Create Responder Action dialog box, do the following:

- a. In the Name text box, type `act_redirect`.
- b. Under Type, select Redirect.
- c. In the Target text area, type the following string: `"http://www.example.com/404.html"`
- d. Click Create, then click Close.

The responder action you configured, named `act_redirect`, now appears in the Responder Actions page.

4. In the navigation pane, click Policies.
5. In the details pane, click Add.
6. In the Create Responder Policy dialog box, do the following:
  - a. In the Name text box, type `pol_redirect`.
  - b. Under Action, select `act_redirect`.
  - c. In the Expression window, type the following rule: `CLIENT.IP.SRC.IN_SUBNET(222.222.0.0/16)`
  - d. Click Create, then click Close.

The responder policy you configured, named `pol_redirect`, now appears in the Responder Policies page.

7. Globally bind your new policy, `pol_redirect`, as described in "[Binding a Responder Policy](#)."

## Diameter Support for Responder

The Responder feature now supports the Diameter protocol. You can configure Responder to respond to Diameter requests as it does HTTP and TCP requests. For example, you could configure Responder to respond to requests from a specific Diameter origin with a redirect to a web page enhanced for mobile devices. A number of NetScaler expressions have been added that support examination of the Diameter header and the attribute-value pairs (AVPs). These expressions support lookup of specific AVPs by index, ID or name, examine the information in each AVP, and send an appropriate response.

### To configure Responder to respond to a Diameter request

To configure the Responder feature to send a response to a diameter request, at the command prompt, type the following commands:

- o add responder action <actname> RESPONDWITH "DIAMETER.NEW\_REDIRECT(\"aaa://host.example.com\")"  
For <actname>, substitute a name for your new action. The name can consist of from one to 127 characters in length, and can contain letters, numbers, and the hyphen (-) and underscore (\_) symbols. For aaa://host.example.com, substitute the URL of the diameter host to which you want to redirect connections.
- o add responder policy <polname> "diameter.req.avp(264).value.eq(\"host1.example.net\")<actname>  
For <polname>, substitute a name for your new policy. As with <actname>, the name can consist of from one to 127 characters in length, and can contain letters, numbers, and the hyphen (-) and underscore (\_) symbols. For host1.example.net, substitute the name of the originating host of the requests that you want to redirect. For <actname>, substitute the name of the action that you just created.
- o bind lb vserver <vservname> -policyName <polname> -priority <priority> -type REQUEST  
For <vservname>, substitute the name of the load balancing virtual server to which you want to bind the policy. For <polname>, substitute the name of the policy you just created. For <priority>, substitute a priority for the policy.

#### Example

To create a Responder action and policy to respond to Diameter requests that originate from "host1.example.net" with a redirect to "host.example.com", you could add the following action and policy, and bind the policy as shown.

```
> add responder action act_resp-dm-redirect RESPONDWITH "DIAMETER.NEW_REDIRECT(\"aaa://host.  
> add responder pol_resp-dm-redirect "diameter.req.avp(264).value.eq(\"host1.example.net\")"  
> bind lb vserver vs1 -policyName pol_resp-dm-redirect -priority 10 -type REQUEST
```

Done



## RADIUS Support for Responder

The NetScaler expressions language contains expressions that can extract information from and manipulate RADIUS requests. These expressions enable you to use the Responder feature to respond to RADIUS requests. Your responder policies and actions can use any expression that is appropriate or relevant to a RADIUS request. The available expressions enable you to identify the RADIUS message type, extract any attribute-value pair (AVP) from the connection, and send different responses on the basis of that information. You can also create policy labels that invoke all responder policies for RADIUS connections.

You can use RADIUS expressions to construct simple responses that do not require communication with the RADIUS server to which the request was sent. When a responder policy matches a connection, the NetScaler ADC constructs and sends the appropriate RADIUS response without contacting the RADIUS authentication server. For example, if the source IP address of a RADIUS request is from a subnet that is specified in the responder policy, the NetScaler ADC can reply to that request with an access-reject message, or can simply drop the request.

You can also create policy labels to route specific types of RADIUS requests through a series of policies that are appropriate to those requests.

Note: The current RADIUS expressions do not work with RADIUS IPv6 attributes.

The NetScaler documentation for expressions that support RADIUS assumes familiarity with the basic structure and purpose of RADIUS communications. If you need more information about RADIUS, see your RADIUS server documentation or search online for an introduction to the RADIUS protocol.

## Configuring Responder Policies for RADIUS

The following procedure uses the NetScaler command line to configure a responder action and policy, and bind the policy to a RADIUS-specific global bind point.

### To configure a Responder action and policy, and bind the policy

At the command prompt, type the following commands:

- add responder action <actName> <actType>
- add responder policy <polName> <rule> <actName>
- bind responder policy <polName> <priority> <nextExpr> -type <bindPoint> where <bindPoint> represents one of the RADIUS-specific global bind points.

## RADIUS Expressions for Responder

In a responder configuration, you can use the following NetScaler expressions to refer to various portions of a RADIUS request.

### Identifying the Type of Connection

RADIUS.IS\_CLIENT

Returns TRUE if the connection is a RADIUS client (request) message.

RADIUS.IS\_SERVER

Returns TRUE if the connection is a RADIUS server (response) message.

### Request Expressions

RADIUS.REQ.CODE

Returns the number that corresponds to the RADIUS request type. A derivative of the num\_at class. For example, a RADIUS access request would return 1 (one). A RADIUS accounting request would return 4.

RADIUS.REQ.LENGTH

Returns the length of the RADIUS request, including the header. A derivative of the num\_at class.

RADIUS.REQ.IDENTIFIER

Returns the RADIUS request identifier, a number assigned to each request that allows the request to be matched to the corresponding response. A derivative of the num\_at class.

RADIUS.REQ.AVP(<AVP Code No>).VALUE

Returns the value of first occurrence of this AVP as a string of type text\_t.

RADIUS.REQ.AVP(<AVP code no>).INSTANCE(instance number)

Returns the specified instance of the AVP as a string of type RAVP\_t. A specific RADIUS AVP can occur multiple times in a RADIUS message. INSTANCE (0) returns the first instance, INSTANCE (1) returns second instance, and so on, up to sixteen instances.

RADIUS.REQ.AVP(<AVP code no>).VALUE(instance number)

Returns the value of specified instance of the AVP as a string of type `text_t`.  
`RADIUS.REQ.AVP(<AVP code no>).COUNT`  
Returns the number of instances of a specific AVP in a RADIUS connection, as an integer.  
`RADIUS.REQ.AVP(<AVP code no>).EXISTS`  
Returns TRUE if the specified type of AVP exists in the message, or FALSE if it does not.

## Response Expressions

RADIUS response expressions are identical to RADIUS request expressions, except that `RES` replaces `REQ`.

## Typecasts of AVP Values

The ADC supports expressions to typecast RADIUS AVP values to the text, integer, unsigned integer, long, unsigned long, ipv4 address, ipv6 address, ipv6 prefix and time data types. The syntax is the same as for other NetScaler typecast expressions.

### Example

The ADC supports expressions to typecast RADIUS AVP values to the text, integer, unsigned integer, long, unsigned long, ipv4 address, ipv6 address, ipv6 prefix and time data types. The syntax is the same as for other NetScaler typecast expressions.

```
RADIUS.REQ.AVP(8).VALUE(0).typecast_ip_address_at
```

## AVP Type Expressions

The NetScaler ADC supports expressions to extract RADIUS AVP values by using the assigned integer codes described in RFC2865 and RFC2866. You can also use text aliases to accomplish the same task. Some examples follow.

```
RADIUS.REQ.AVP(1).VALUE or RADIUS.REQ.USERNAME.value  
  Extracts the RADIUS user-name value.  
RADIUS.REQ.AVP(4).VALUE or RADIUS.REQ.ACCT_SESSION_ID.value  
  Extracts the Acct-Session-ID AVP (code 44) from the message.  
RADIUS.REQ.AVP(26).VALUE or RADIUS.REQ.VENDOR_SPECIFIC.VALUE  
  Extracts the vendor-specific value.
```

The values of most commonly-used RADIUS AVPs can be extracted in the same manner.

## RADIUS Bind Points

Four global bind points are available for policies that contain RADIUS expressions.

```
RADIUS_REQ_OVERRIDE  
  Priority/override request policy queue.  
RADIUS_REQ_DEFAULT  
  Standard request policy queue.  
RADIUS_RES_OVERRIDE  
  Priority/override response policy queue.  
RADIUS_RES_DEFAULT  
  Standard response policy queue.
```

## RADIUS Responder-Specific Expressions

```
RADIUS_RESPONDWITH  
  Respond with the specified RADIUS response. The response is created with NetScaler expressions, both RADIUS  
  expressions and any others that are applicable.  
RADIUS.NEW_ANSWER  
  Sends a new RADIUS answer to the user.  
RADIUS.NEW_ACCESSREJECT  
  Rejects the RADIUS request.  
RADIUS.NEW_AVP  
  Adds the specified new AVP to the response.
```

## Use Cases

Following are use cases for RADIUS with responder.

## Blocking RADIUS Requests from a Specific Network

To configure the responder feature to block authentication requests from a specific network, begin by creating a responder action that rejects requests. Use the action in a policy that selects requests from the networks that you want to block. Bind the responder policy to a RADIUS-specific global bind point, specifying:

- The priority
- END as the `nextExpr` value, to ensure that policy evaluation stops when this policy is matched
- RADIUS\_REQ\_OVERRIDE as the queue to which you assign the policy, so that it is evaluated before policies assigned to the default queue

#### To configure Responder to block logons from a specific network

- add responder action `<actName> <actType>`
- add responder policy `<polName> <rule> <actName>`
- bind responder global `<polName> <priority> <nextExpr> -type <bindPoint>`

#### Example

```
add responder action rspActRadiusReject respondwith radius.new_accessreject
add responder policy rspPolRadiusReject client.ip.src.in_subnet(10.224.85.0/24) rspActRadius
bind responder global rspPolRadiusReject 1 END -type RADIUS_REQ_OVERRIDE
```

## Troubleshooting

If the responder 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 ns.conf file
- The relevant trace files from the client and the NetScaler appliance

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 Responder Issues

Updated: 2013-07-29

#### • Issue

The Responder feature is configured, but the responder action is not working.

#### Resolution

Verify that the feature is enabled.

Check the hit counters of any of the policies to see if the counters are getting incremented.

Verify that the policies and actions are configured correctly.

Verify that the actions and policies are bound appropriately.

Record the packet traces on the client and the NetScaler appliance, and analyze them to get some pointer to the issue.

Record the iehttpHeaters packet traces on the client and verify the HTTP requests and responses to get some pointer to the issue.

#### • Issue

You need to create a maintenance page.

#### Resolution

1. Configure the services and virtual Server.
2. Configure a backup virtual server with a service bound to it. This ensures that the status of the Web site is always displayed as UP.
3. Configure the primary virtual server to use the backup virtual server as a backup.
4. Create a responder action with an appropriate target. Following is an example for your reference:

```
add responder action sorry_page respondwith q{"HTTP/1.0 200 OK" +  
  \r\n\r\n" + "<html><body>Sorry, this page is not  
  available</body></html>" + "\r\n"} .
```

5. Create a responder policy and bind the action to it.
6. Bind the responder policy to the backup virtual Server.

## Rewrite

Rewrite refers to the rewriting of some information in the requests or responses handled by the NetScaler appliance. Rewriting can help in providing access to the requested content without exposing unnecessary details about the Web site's actual configuration. A few situations in which the rewrite feature is useful are described below:

- To improve security, the NetScaler can rewrite all the `http://` links to `https://` in the response body.
- In the SSL offload deployment, the insecure links in the response have to be converted into secure links. Using the rewrite option, you can rewrite all the `http://` links to `https://` for making sure that the outgoing responses from NetScaler to the client have the secured links.
- If a Web site has to show an error page, you can show a custom error page instead of the default 404 Error page. For example, if you show the home page or site map of the Web site instead of an error page, the visitor remains on the site instead of moving away from the Web site.
- If you want to launch a new Web site, but use the old URL, you can use the Rewrite option.
- When a topic in a site has a complicated URL, you can rewrite it with a simple, easy-to-remember URL (also referred to as 'cool URL').
- You can append the default page name to the URL of a Web site. For example, if the default page of a company's Web site is '`http://www.abc.com/index.php`', when the user types '`abc.com`' in the address bar of the browser, you can rewrite the URL to '`abc.com/index.php`'.

When you enable the rewrite feature, NetScaler can modify the headers and body of HTTP requests and responses.

To rewrite HTTP requests and responses, you can use protocol-aware NetScaler policy expressions in the rewrite policies you configure. The virtual servers that manage the HTTP requests and responses must be of type `HTTP` or `SSL`. In HTTP traffic, you can take the following actions:

- Modify the URL of a request
- Add, modify or delete headers
- Add, replace, or delete any specific string within the body or headers.

To rewrite TCP payloads, consider the payload as a raw stream of bytes. Each of the virtual servers that managing the TCP connections must be of type `TCP` or `SSL_TCP`. The term TCP rewrite is used to refer to the rewrite of TCP payloads that are not HTTP data. In TCP traffic, you can add, modify, or delete any part of the TCP payload.

For examples to use the rewrite feature, see "[Rewrite Action and Policy Examples](#)."

## Comparison between Rewrite and Responder options

The main difference between the rewrite feature and the responder feature is as follows:

Responder cannot be used for response or server-based expressions. Responder can be used only for the following scenarios depending on client parameters:

- Redirecting a http request to new Web sites or Web pages
- Responding with some custom response
- Dropping or resetting a connection at request level

In case of a responder policy, the NetScaler examines the request from the client, takes action according to the applicable policies, sends the response to the client, and closes the connection with the client.

In case of a rewrite policy, the NetScaler examines the request from the client or response from the server, takes action according to the applicable policies, and forwards the traffic to the client or the server.

In general, it is recommended to use responder if you want the NetScaler to reset or drop a connection based on a client or request-based parameter. Use responder to redirect traffic, or respond with custom messages. Use rewrite for manipulating data on HTTP requests and responses.

## How Rewrite Works

A rewrite policy consists of a rule and action. The rule determines the traffic on which rewrite is applied and the action determines the action to be taken by the NetScaler. You can define multiple rewrite policies. For each policy, specify the bind point and priority.

A bind point refers to a point in the traffic flow at which the NetScaler examines the traffic to verify whether any rewrite policy can be applied to it. You can bind a policy to a specific load balancing or content switching virtual server, or make the policy global if you want the policy to be applied to the entire traffic handled by the NetScaler. These policies are referred to as global policies.

In addition to the user-defined policies, the NetScaler has some default policies. You cannot modify or delete a default policy.

For evaluating the policies, NetScaler follows the order mentioned below:

- Global policies
- Policies bound to specific virtual servers
- Default policies

Note: NetScaler can apply a rewrite policy only when it is bound to a point.

NetScaler implements the rewrite feature in the following steps:

- The NetScaler appliance checks for global policies and then checks for policies at individual bind points.
- If multiple policies are bound to a bind point, the NetScaler evaluates the policies in the order of their priority. The policy with the highest priority is evaluated first. After evaluating each policy, if the policy is evaluated to TRUE (the traffic matches the rule), it adds the action associated with the policy to a list of actions to be performed. A match occurs when the characteristics specified in the policy rule match the characteristics of the request or response being evaluated.
- For any policy, in addition to the action, you can specify the policy that should be evaluated after the current policy is evaluated. This policy is referred to as the 'Go to Expression'. For any policy, if a Go to Expression (gotoPriorityExpr) is specified, the NetScaler evaluates the Go to Expression policy; it ignores policy with the next highest priority.

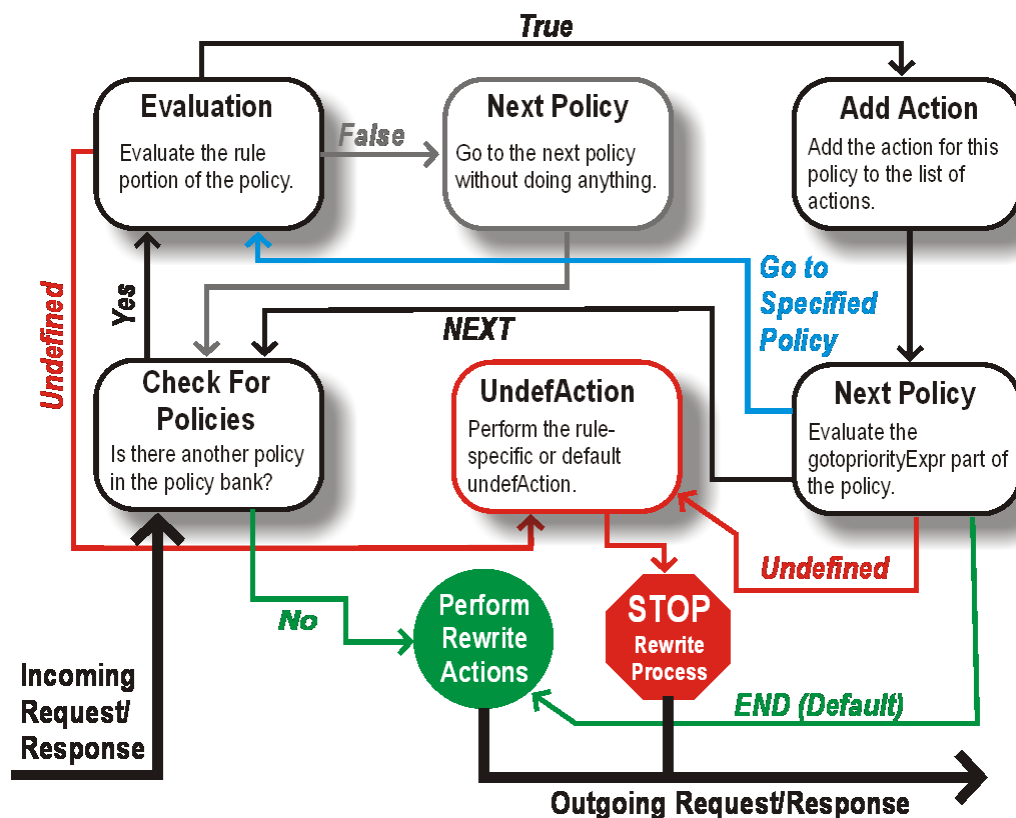
You can specify the priority of the policy to indicate the Go to Expression policy; you cannot use the name of the policy. If you want the NetScaler to stop evaluating other policies after evaluating a particular policy, you can set the Go to Expression to 'END'.

- After all the policies are evaluated or when a policy has the Go to Expression set as END, the NetScaler starts performing the actions according to the list of actions.

For more information about configuring rewrite policies, see ["Configuring a Rewrite Policy"](#) and about binding rewrite policies, see ["Binding a Rewrite Policy."](#)

The following figure illustrates how NetScaler processes a request or response when the rewrite feature is used.

Figure 1. The Rewrite Process



## Policy Evaluation

The policy with the highest priority is evaluated first. NetScaler does not stop the evaluation of rewrite policies when it finds a match; it evaluates all the rewrite policies configured on the NetScaler.

- If a policy evaluates to TRUE, the NetScaler follows the procedure below:
  - If the policy has the Go to Expression set to END, the NetScaler stops evaluating all the other policies and starts performing the rewrite.
  - The *gotoPriorityExpression* can be set to 'NEXT', 'END', some integer or 'INVOCATION\_LIST'. The value determines the policy with the next priority. The following table shows the action taken by NetScaler for each value of the expression.

Value of the expression	Action
NEXT	Policy with the next priority gets evaluated.
END	Evaluation of policies stops.
<an integer>	Policy with specified priority gets evaluated.
INVOCATION_LIST	Goto NEXT or END is applied based on the result of the invocation list.

- If a policy evaluates to FALSE, the NetScaler continues the evaluation in the order of priority.
- If a policy evaluates to UNDEFINED (cannot be evaluated on the received traffic due to an error), the NetScaler performs the action assigned to the UNDEFINED condition (referred to as undefAction) and stops further evaluation of policies.

The NetScaler starts the actual rewriting only after the evaluation is complete. It refers to the list of actions identified by policies that are evaluated to TRUE, and starts the rewriting. After implementing all the actions in the list, the NetScaler forwards the traffic as required.

Note: Ensure that the policies do not specify conflicting or overlapping actions on the same part of the HTTP header or body, or TCP payload. When such a conflict occurs, the NetScaler encounters an undefined situation and aborts the rewrite.

## Rewrite Actions

On the NetScaler appliance, specify the actions to be taken such as adding, replacing, or deleting text within the body, or adding, modifying or deleting headers, or any changes in the TCP payload as rewrite actions. For more information about rewrite actions, see "Configuring a Rewrite Action."

The following table describes the steps the NetScaler can take when a policy evaluates to TRUE.

Action	Result
Insert	The rewrite action specified for the policy is carried out.
NOREWRITE	The request or response is not rewritten. NetScaler forwards the traffic without rewriting any part of the message.
RESET	The connection is aborted at the TCP level.
DROP	The message is dropped.

Note: For any policy, you can configure the undefaction (action to be taken when the policy evaluates to UNDEFINED) as NOREWRITE, RESET, or DROP.

To use the Rewrite feature, take the following steps:

- Enable the feature on the NetScaler.
- Define rewrite actions.
- Define rewrite policies.
- Bind the policies to a bind point to bring a policy into effect.



# Enabling the Rewrite Feature

Enable the rewrite feature on the NetScaler appliance if you want to rewrite the HTTP or TCP requests or responses. If the feature is enabled, NetScaler takes rewrite action according to the specified policies. For more information, see ["How Rewrite Works."](#)

## To enable the rewrite feature by using the command line interface

At the command prompt, type the following commands to enable the rewrite feature and verify the configuration:

- enable ns feature REWRITE
- show ns feature

### Example

```
> enable ns feature REWRITE
Done
> show ns feature
```

	Feature	Acronym	Status
	-----	-----	-----
1)	Web Logging	WL	OFF
2)	Surge Protection	SP	ON
.			
.			
.			
19)	<b>Rewrite</b>	<b>REWRITE</b>	<b>ON</b>
.			
.			
24)	NetScaler Push	push	OFF
Done			

## To enable the rewrite feature by using the configuration utility

1. In the navigation pane, click System, and then click Settings.
2. In the details pane, under Modes and Features, click Configure basic features.
3. In the Configure Basic Features dialog box, select the Rewrite check box, and then click OK.
4. In the Enable/Disable Feature(s) dialog box, click Yes. A message appears in the status bar, stating that the selected feature was enabled.

## Configuring a Rewrite Action

After enabling the rewrite feature, you need to configure one or more actions unless a built-in rewrite action is sufficient. All of the built-in actions have names beginning with the string `ns_cvprn`, followed by a string of letters and underscore characters. Built-in actions perform useful and complex tasks such as decoding parts of a clientless VPN request or response or modifying JavaScript or XML data. The built-in actions can be viewed, enabled, and disabled, but cannot be modified or deleted.

Target expressions in actions for TCP rewrite must begin with one of the following expression prefixes:

- **CLIENT.TCP.PAYLOAD.** For rewriting TCP payloads in client requests. For example, `CLIENT.TCP.PAYLOAD(10000).AFTER_STR("string1")`.
- **SERVER.TCP.PAYLOAD.** For rewriting TCP payloads in server responses. For example, `SERVER.TCP.PAYLOAD(1000).B64DECODE.BETWEEN("string1","string2")`.

You can use all types of existing string manipulation functions with these prefixes to identify the strings that you want to rewrite. To configure a rewrite action, you assign it a name, specify an action type, and add one or more arguments specifying additional data. The following table describes the action types and the arguments you use with them.

Note: Action types that can be used only for HTTP rewrite are identified in the **Rewrite Action Type** column.

Table 1. Rewrite Action Types and Their Arguments

Rewrite Action Type	Argument 1	Argument 2
<b>INSERT_HTTP_HEADER:</b> Inserts the HTTP header you specify into the HTTP request or response. This is the default choice. This action type can be used only with HTTP requests and responses.	The HTTP header you want to insert.  For example, if you want to insert the client IP from which a request is sent, type <code>Client-IP</code> .	A string expression that describes the contents of the header you want to insert.  For example, if you want to insert the Client IP from which a request is sent, type <code>CLIENT.IP.SRC</code> .
<b>INSERT_BEFORE:</b> Inserts a new string before the designated string.	A string expression that describes the string before which you want to insert a new string.  For example, if you want to find the hostname <code>www.example.com</code> and insert a string before the <code>example.com</code> portion, type the following: <code>HTTP.REQ.HOSTNAME.BEFORE_STR("example.com")</code>	A string expression that describes the new string you want to insert.  For example, if you want to insert the new string <code>en.</code> before the string <code>example</code> in the hostname, type <code>en</code> followed by a period.
<b>INSERT_AFTER:</b> Inserts a new string after the designated string.	A string expression that describes the string after which you want to insert a new string.  For example, if you want to find the hostname <code>www.example.com</code> , and insert a string after the <code>www.</code> portion, type the following: <code>HTTP.REQ.HOSTNAME.AFTER_STR("www.")</code>	A string expression that describes the new string you want to insert.  For example, if you want to insert the new string <code>en.</code> after the string <code>www.</code> in the hostname, type <code>en</code> followed by a period.
<b>REPLACE:</b> Replaces the designated string with a different string.	A string expression that describes the string you want to replace with a new string.  For example, if you want to replace the entire hostname in the Host header, type <code>HTTP.REQ.HOSTNAME.SERVER</code> .	A string expression that describes the new string you want to insert.  For example, if you want to replace the current host header

		with the string web01.example.net, type web01.example.net.
<b>DELETE:</b> Deletes the designated string.	<p>A string expression that describes the string you want to delete.</p> <p>For example, if you want to find and delete the string .en in the hostname of HTTP response headers, type the following: <code>HTTP.RES.HEADER ( "Host" ).SUBSTR ( "en." )</code></p>	Ã
<b>DELETE_HTTP_HEADER:</b> Deletes the designated HTTP header, including all header contents. This action type can be used only with HTTP requests and responses.	<p>The name of the HTTP header you want to delete.</p> <p>For example, if you want to delete the cache-control header from HTTP responses, type <code>HTTP.RES.HEADER ( "Cache-Control" )</code>.</p>	Ã
<b>CORRUPT_HTTP_HEADER:</b> Replaces the name of the given HTTP header with a corrupted name so that it will not be recognized by the receiver. This action type can be used only with HTTP requests and responses.	<p>The name of the HTTP header that you want to corrupt. If the specified header occurs more than once in a request, all the occurrences are corrupted.</p> <p>For example, if you want to corrupt the Host header in an HTTP request, you can use the following rewrite action command:</p> <pre>add rewrite action corrupt_header_act CORRUPT_HTTP_HEADER Host.</pre>	Ã
<b>REPLACE_HTTP_RES:</b> Replace the http response with the value specified in the target field. This action type can be used only with HTTP requests and responses.	<p>A string expression that describes the string you want to replace the HTTP response with.</p> <p>For example, type <code>HTTP 200 OK You are not authorized to view this page</code> to replace the entire HTTP response with this warning.</p>	Ã
<b>REPLACE_ALL:</b> Will replace all occurrences of a pattern in the target text reference with the value specified in the string builder expression.	The part of either the HTTP request or response where you want to carry out the replacement.	A string expression that describes the new string you want to insert.
<b>DELETE_ALL:</b> Delete every occurrence of the pattern specified in the target text reference.	The part of either the HTTP request or response where you want the deletion to occur.	A string pattern after which the deletion should occur.
<b>INSERT_AFTER_ALL:</b> Inserts the value specified by string builder expression after each occurrence of a specified pattern in the target text reference.	The part of either the HTTP request or response where you want the insertion to occur.	A string expression that describes the new string you want to insert.
<b>INSERT_BEFORE_ALL:</b> Inserts the value you specify before each occurrence of the pattern you specify.	The part of either the HTTP request or response that you want to delete.	A string expression that describes the new string you want to insert.
<b>CLIENTLESS_VPN_ENCODE:</b> Encodes the URL you specify in clientless VPN format.	The URL you want to encode.	Ã

<b>CLIENTLESS_VPN_ENCODE_ALL:</b> Encodes all of the URLs you specify in clientless VPN format.	A pattern that matches the URLs you want to encode.	Â
<b>CLIENTLESS_VPN_DECODE:</b> Decodes the URL you specify from clientless VPN format and returns it as unencoded text.	The URL you want to decode.	Â
<b>CLIENTLESS_VPN_DECODE_ALL:</b> Decodes all of the URLs you specify from clientless VPN format and returns them as unencoded text.	A pattern that matches all of the URLs you want to decode.	Â

## To create a new rewrite action by using the command line interface

At the command prompt, type the following commands to create a new rewrite action and verify the configuration:

- add rewrite action <name> <type> <target> [<stringBuilderExpr>] [(-pattern <expression> | -patset <string>)] [-bypassSafetyCheck (YES|NO)]
- show rewrite action <name>

### Example 1: Inserting an HTTP Header With the Client IP

```
> add rewrite action insertact INSERT_HTTP_HEADER "client-IP" CLIENT.IP.SRC
Done

> show rewrite action insertact

Name: insertact
Operation: insert_http_header    Target:Client-IP
Value:CLIENT.IP.SRC
BypassSafetyCheck : NO
Hits: 0
Undef Hits: 0
Action Reference Count: 0

Done
```

### Example 2: Replacing Strings in a TCP Payload (TCP Rewrite)

```
> add rewrite action client_tcp_payload_replace_all REPLACE_ALL
'client.tcp.payload(1000)' 'new-string' -search text("old-string")
Done
> show rewrite action client_tcp_payload_replace_all

Name: client_tcp_payload_replace_all
Operation: replace_all
Target:client.tcp.payload(1000)
Value:"new-string"
Search: text("old-string")
BypassSafetyCheck : NO
Hits: 0
Undef Hits: 0
Action Reference Count: 0

Done
>
```

## To modify an existing rewrite action by using the command line interface

At the command prompt, type the following commands to modify an existing rewrite action and verify the configuration:

- set rewrite action <name> [-target <string>] [-stringBuilderExpr <string>] [(-pattern <expression> | -patset <string>)] [-bypassSafetyCheck (YES|NO)]
- show rewrite action <name>

### Example

```
> set rewrite action insertact -target "Client-IP"
Done
```

```
> show rewrite action insertact

Name: insertact
Operation: insert_http_header    Target:Client-IP
Value:CLIENT.IP.SRC
BypassSafetyCheck : NO
Hits: 0
Undef Hits: 0
Action Reference Count: 0

Done
```

## To remove a rewrite action by using the command line interface

At the command prompt, type the following commands to remove a rewrite action :

```
rm rewrite action <name>
```

### Example

```
> rm rewrite action insertact
Done
```

## To configure a rewrite action by using the configuration utility

1. Navigate to AppExpert > Rewrite > Actions.
2. In the details pane, do one of the following:
  - o To create a new action, click Add.
  - o To modify an existing action, select the action, and then click Open.
3. Click Create or OK. A message appears in the status bar, stating that the Action has been configured successfully.
4. Repeat steps 2 through 4 to create or modify as many rewrite actions as you wish.
5. Click Close.

## To add an expression by using the Add Expression dialog box

1. In the Create Rewrite Action or Configure Rewrite Action dialog box, under the text area for the type argument you want to enter, click Add.
2. In the Add Expression dialog box, in the first list box choose the first term for your expression.

### HTTP

The HTTP protocol. Choose this if you want to examine some aspect of the request that pertains to the HTTP protocol.

### SYS

The protected Web site(s). Choose this if you want to examine some aspect of the request that pertains to the recipient of the request.

### CLIENT

The computer that sent the request. Choose this if you want to examine some aspect of the sender of the request.

When you make your choice, the rightmost list box lists appropriate terms for the next part of your expression.

3. In the second list box, choose the second term for your expression. The choices depend upon which choice you made in the previous step, and are appropriate to the context. After you make your second choice, the Help window below the Construct Expression window (which was blank) displays help describing the purpose and use of the term you just chose.
4. Continue choosing terms from the list boxes that appear to the right of the previous list box, or typing strings or numbers in the text boxes that appear to prompt you to enter a value, until your expression is finished.

For more information about the PI expressions language and creating expressions for responder policies, see "[Policies and Expressions](#)."

If you want to test the effect of a rewrite action when used on sample HTTP data, you can use the Rewrite Expression Evaluator.

Note: The Rewrite Expression Evaluator is only available in the configuration utility. There is no NetScaler command line version.

## To evaluate a rewrite action by using the Rewrite Action Evaluator dialog box

1. In the Rewrite Actions details pane, select the rewrite action that you want to evaluate, and then click Evaluate.
2. In the Rewrite Expression Evaluator dialog box, specify values for the following parameters. (An asterisk indicates a required parameter.)

- Rewrite Action\*â€”If the rewrite action you want to evaluate is not already selected, select it from the drop-down list. After you select a Rewrite action, the Details section displays the details of the selected Rewrite action.
  - New\*â€”Select New to open the Create Rewrite Action dialog box and create a new rewrite action.
  - Modify\*â€”Select Modify to open the Configure Rewrite Action dialog box and modify the selected rewrite action.
  - Flow Type\*â€”Specifies whether to test the selected rewrite action with HTTP Request data or HTTP Response data. The default is Request. If you want to test with Response data, select Response.
  - HTTP Request/Response Data\*â€”Provides a space for you to provide the HTTP data that the Rewrite Action Evaluator will use for testing. You can paste the data directly into the window, or click Sample to insert some sample HTTP headers.
  - Show end-of-lineâ€”Specifies whether to show UNIX-style end-of-line characters (\n) at the end of each line of sample HTTP data.
  - Sampleâ€”Inserts sample HTTP data into the HTTP Request/Response Data window. You can choose either GET or POST data.
  - Browseâ€”Opens a local browse window so that you can choose a file containing sample HTTP data from a local or network location.
  - Clearâ€”Clears the current sample HTTP data from the HTTP Request/Response Data window.
3. Click Evaluate. The Rewrite Action Evaluator evaluates the effect of the Rewrite action on the sample data that you chose, and displays the results as modified by the selected Rewrite action in the Results window. Additions and deletions are highlighted as indicated in the legend in the lower left-hand corner of the dialog box.
  4. Continue evaluating Rewrite actions until you have determined that all of your actions have the effect that you wanted.
    - You can modify the selected rewrite action and test the modified version by clicking Modify to open the Configure Rewrite Action dialog box, making and saving your changes, and then clicking Evaluate again.
    - You can evaluate a different rewrite action using the same request or response data by selecting it from the Rewrite Action drop-down list, and then clicking Evaluate again.
  5. Click Close to close the Rewrite Expression Evaluator and return to the Rewrite Actions pane.

To delete a rewrite action, select the rewrite action you want to delete, then click Remove and, when prompted, confirm your choice by clicking OK.

## Configuring a Rewrite Policy

After you create any needed rewrite action(s), you must create at least one rewrite policy to select the requests that you want the NetScaler appliance to rewrite.

A rewrite policy consists of a rule, which itself consists of one or more expressions, and an associated action that is performed if a request or response matches the rule. Policy rules for evaluating HTTP requests and responses can be based on almost any part of a request or response.

Even though you cannot use TCP rewrite actions to rewrite data other than the TCP payload, you can base the policy rules for TCP rewrite policies on the information in the transport layer and the layers below the transport layer.

If a configured rule matches a request or response, the corresponding policy is triggered and the action associated with it is carried out.

Note: You can use either the command line interface or the configuration utility to create and configure rewrite policies. Users who are not thoroughly familiar with the command line interface and the NetScaler Policy expression language will usually find using the configuration utility much easier.

### To add a new rewrite policy by using the command line interface

At the command prompt, type the following commands to add a new rewrite policy and verify the configuration:

- o add rewrite policy <name> <expression> <action> [<undefaction>]
- o show rewrite policy <name>

#### Example 1: Rewriting HTTP Content

```
> add rewrite policy policyNew "HTTP.RES.IS_VALID" insertact NOREWRITE
Done
> show rewrite policy policyNew
    Name: policyNew
    Rule: HTTP.RES.IS_VALID
    RewriteAction: insertact
    UndefAction: NOREWRITE
    Hits: 0
    Undef Hits: 0

Done
```

#### Example 2: Rewriting a TCP Payload (TCP Rewrite)

```
> add rewrite policy client_tcp_payload_policy CLIENT.IP.SRC.EQ(172.168.12.232) client_tcp_p
Done
> show rewrite policy client_tcp_payload_policy
    Name: client_tcp_payload_policy
    Rule: CLIENT.IP.SRC.EQ(172.168.12.232)
    RewriteAction: client_tcp_payload_replace_all
    UndefAction: Use Global
    LogAction: Use Global
    Hits: 0
    Undef Hits: 0

Done
>
```

### To modify an existing rewrite policy by using the command line interface

At the command prompt, type the following commands to modify an existing rewrite policy and verify the configuration:

- o set rewrite policy <name> -rule <expression> -action <action> [<undefaction>]
- o show rewrite policy <name>

#### Example

```
> set rewrite policy policyNew -rule "HTTP.RES.IS_VALID" -action insertaction
Done
> show rewrite policy policyNew
```

```
Name: policyNew
Rule: HTTP.RES.IS_VALID
RewriteAction: insertaction
UndefAction: NOREWRITE
Hits: 0
Undef Hits: 0
```

Done

## To remove a rewrite policy by using the command line interface

At the command prompt, type the following command to remove a rewrite policy:  
`rm rewrite policy <name>`

### Example

```
> rm rewrite policy policyNew
Done
```

## To configure a rewrite policy by using the configuration utility

1. Navigate to AppExpert > Rewrite > Policies.
2. In the details pane, do one of the following:
  - To create a new policy, click Add.
  - To modify an existing policy, select the policy, and then click Open.
3. Click Create or OK. A message appears in the status bar, stating that the Policy has been configured successfully.
4. Repeat steps 2 through 4 to create or modify as many rewrite actions as you wish.
5. Click Close. To delete a rewrite policy, select the rewrite policy you want to delete, then click Remove and, when prompted, confirm your choice by clicking OK.



## Binding a Rewrite Policy

After creating a rewrite policy, you must bind it to put it into effect. You can bind your policy to Global if you want to apply it to all traffic that passes through your NetScaler, or you can bind your policy to a specific virtual server or bind point to direct only that virtual server or bind point's incoming traffic to that policy. If an incoming request matches a rewrite policy, the action associated with that policy is carried out.

Rewrite policies for evaluating HTTP requests and responses can be bound to virtual servers of type HTTP or SSL, or they can be bound to the `REQ_OVERRIDE`, `REQ_DEFAULT`, `RES_OVERRIDE`, and `RES_DEFAULT` bind points. Rewrite policies for TCP rewrite can be bound only to virtual servers of type TCP or `SSL_TCP`, or to the `OTHERTCP_REQ_OVERRIDE`, `OTHERTCP_REQ_DEFAULT`, `OTHERTCP_RES_OVERRIDE`, and `OTHERTCP_RES_DEFAULT` bind points.

**Note:** The term `OTHERTCP` is used in the context of the NetScaler appliance to refer to all TCP or `SSL_TCP` requests and responses that you want to treat as a raw stream of bytes regardless of the protocols that the TCP packets encapsulate.

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.

In the NetScaler operating system, policy priorities work in reverse order - the higher the number, the lower the priority. For example, if you have three policies with priorities of 10, 100, and 1000, the policy assigned a priority of 10 is applied first, then the policy assigned a priority of 100, and finally the policy assigned an order of 1000.

Unlike most other features in the NetScaler operating system, the rewrite feature continues to evaluate and implement policies after a request matches a policy. However, the effect of a particular action policy on a request or response will often be different depending on whether it is performed before or after another action. Priority is important to get the results you intended.

You can leave yourself plenty of room to add other policies in any order, and still set them to evaluate in the order you want, by setting priorities with intervals of 50 or 100 between each policy when you bind it. If you do this, you can add additional policies at any time without having to reassign the priority of an existing policy.

When binding a rewrite policy, you also have the option of assigning a goto expression (`gotoPriorityExpression`) to the policy. A goto expression can be any positive integer that matches the priority assigned to a different policy that has a higher priority than the policy that contains the goto expression. If you assign a goto expression to a policy, and a request or response matches the policy, the NetScaler will immediately go to the policy whose priority matches the goto expression. It will skip over any policies with priority numbers that are lower than that of the current policy, but higher than the priority number of the goto expression, and not evaluate those policies.

For more information about binding policies on the NetScaler, see ["Binding a Rewrite Policy."](#)

## To globally bind a rewrite policy by using the command line interface

At the command prompt, type the following commands to globally bind a rewrite policy and verify the configuration:

- `bind rewrite global <policyName> <priority> [<gotoPriorityExpression>] [-type <type>] [-invoke (<labelType> <labelName>)]`
- `show rewrite global`

### Example

```
>bind rewrite global policyNew 10
Done

> show rewrite global
1)      Global bindpoint: RES_DEFAULT
        Number of bound policies: 1

2)      Global bindpoint: REQ_OVERRIDE
        Number of bound policies: 1

Done
```

## To bind rewrite policy to a specific virtual server by using the command line interface

At the command prompt, type the following commands to bind rewrite policy to a specific virtual server and verify the configuration:

- o bind lb vserver <name>@ (<serviceName>@ [-weight <positive\_integer>]) | <serviceGroupName>@ | (-policyName <string>@ [-priority <positive\_integer>] [-gotoPriorityExpression <expression>] [-type (REQUEST | RESPONSE)] [-invoke (<labelType> <labelName>)] )
- o show lb vserver <name>

### Example

```
> bind lb vserver lbvip -policyName ns_cmp_msapp -priority 50
Done
>
> show lb vserver lbvip
    lbvip (8.7.6.6:80) - HTTP          Type: ADDRESS
    State: DOWN
    Last state change was at Wed Jul 15 05:54:24 2009 (+226 ms)
    Time since last state change: 28 days, 01:57:26.350
    Effective State: DOWN
    Client Idle Timeout: 180 sec
    Down state flush: ENABLED
    Disable Primary Vserver On Down : DISABLED
    Port Rewrite : DISABLED
    No. of Bound Services :    0 (Total)          0 (Active)
    Configured Method: LEASTCONNECTION
    Mode: IP
    Persistence: NONE
    Vserver IP and Port insertion: OFF
    Push: DISABLED   Push VServer:
    Push Multi Clients: NO
    Push Label Rule: none

1)      Policy : ns_cmp_msapp Priority:50
2)      Policy : cf-pol Priority:1          Inherited
Done
```

## To bind a rewrite policy to a bind point by using the configuration utility

1. Navigate to AppExpert > Rewrite > Policies.
2. In the details pane, select the rewrite policy you want to globally bind, and then click Policy Manager.
3. In the Rewrite Policy Manager dialog box, in the Bind Points menu, do one of the following:
  - a. If you want to configure bindings for HTTP rewrite policies, click HTTP, and then click either Request or Response, depending on whether you want to configure request-based rewrite policies or response-based rewrite policies.
  - b. If you want to configure bindings for TCP rewrite policies, click TCP, and then click either Client or Server, depending on whether you want to configure client-side TCP rewrite policies or server-side TCP rewrite policies.
4. Click the bind point to which you want to bind the rewrite policy. The Rewrite Policy Manager dialog box displays all the rewrite policies that are bound to the selected bind point.
5. Click Insert Policy to insert a new row and display a drop-down list with all available, unbound rewrite policies.
6. Click the policy you want to bind to the bind point. The policy is inserted into the list of rewrite policies bound to the bind point.
7. In the Priority column, you can change the priority to any positive integer. For more information about this parameter, see `priority` in "Parameters for binding a rewrite policy."
8. If you want to skip over policies and go directly to a specific policy in the event that the current policy is matched, change the value in the Goto Expression column to equal the priority of the next policy to be applied.. For more information about this parameter, see `gotoPriorityExpression` in "Parameters for binding a rewrite policy."
9. To modify a policy, click the policy, and then click Modify Policy.
10. To unbind a policy, click the policy, and then click Unbind Policy.
11. To modify an action, in the Action column, click the action you want to modify, and then click Modify Action.
12. To modify an invoke label, in the Invoke column, click the invoke label you want to modify, and then click Modify Invoke Label.
13. To regenerate the priorities of all the policies that are bound to the bind point you are currently configuring, click Regenerate Priorities. The policies retain their existing priorities relative to the other policies, but the priorities are renumbered in multiples of ten.
14. Click Apply Changes.
15. Click Close. A message appears in the status bar, stating that the Policy has been configured successfully.

## To bind a rewrite policy to a specific virtual server by using the configuration utility

1. Navigate to Traffic Management > Load Balancing > Virtual Servers.

2. In the details pane list of virtual servers, select the virtual server to which you want to bind the rewrite policy, and then click Open.
3. In the Configure Virtual Server (Load Balancing) dialog box, select the Policies tab. All policies configured on your NetScaler appear on the list.
4. Select the check box next to the name of the policy you want to bind to this virtual server.
5. Click OK. A message appears in the status bar, stating that the Policy has been configured successfully.

## Configuring Rewrite Policy Labels

If you want to build a more complex policy structure than is supported by single policies, you can create policy labels and then bind them as you would policies. A policy label is a user-defined point to which policies are bound. When a policy label is invoked, all the policies bound to it are evaluated in the order of the priority you configured. A policy label can include one or multiple policies, each of which can be assigned its own result. A match on one policy in the policy label can result in proceeding to the next policy, invoking a different policy label or appropriate resource, or an immediate end to policy evaluation and return of control to the policy that invoked the policy label.

A rewrite policy label consists of a name, a transform name that describes the type of policy included in the policy label, and a list of policies bound to the policy label. Each policy that is bound to the policy label contains all of the elements described in "[Configuring a Rewrite Policy](#)."

Note: You can use either the command line interface or the configuration utility to create and configure rewrite policy labels. Users who are not thoroughly familiar with the command line interface and the NetScaler Policy Infrastructure (PI) language will usually find using the configuration utility much easier.

### To configure a rewrite policy label by using the command line interface

To add a new rewrite policy label, at the command prompt, type the following command:

```
add rewrite policylabel <labelName> <transform>
```

For example, to add a rewrite policy label named `polLabelHTTPResponses` to group all policies that work on HTTP responses, you would type the following:

```
add rewrite policylabel polLabelHTTPResponses http_res
```

To modify an existing rewrite policy label, at the NetScaler command prompt, type the following command:

```
set rewrite policy <name> <transform>
```

Note: The `set rewrite policy` command takes the same options as the `add rewrite policy` command.

To remove a rewrite policy label, at the NetScaler command prompt, type the following command:

```
rm rewrite policy<name>
```

For example, to remove a rewrite policy label named `polLabelHTTPResponses`, you would type the following:

```
rm rewrite policy polLabelHTTPResponses
```

### To configure a rewrite policy label by using the configuration utility

1. Navigate to AppExpert > Rewrite > Policy Labels.
2. In the details pane, do one of the following:
  - To create a new policy label, click Add.
  - To modify an existing policy label, select the policy, and then click Open.
3. Add or remove policies from the list that is bound to the policy label.
  - To add a policy to the list, click Insert Policy, and choose a policy from the drop-down list. You can create a new policy and add it to the list by choosing New Policy in the list, and following the instructions in "[Configuring a Rewrite Policy](#)."
  - To remove a policy from the list, select that policy, and then click Unbind Policy.
4. Modify the priority of each policy by editing the number in the Priority column.

You can also automatically renumber policies by clicking Regenerate Priorities.

5. Click Create or OK, and then click Close.

To remove a policy label, select it, and then click Remove. To rename a policy label, select it and then click Rename. Edit the name of the policy, and then click OK to save your changes.

## Configuring the Default Rewrite Action

An undefined event is triggered when the NetScaler cannot evaluate a policy, usually because it detects a logical or other error in the policy or an error condition on the NetScaler. When the rewrite policy evaluation results in an error, the specified undefined action is carried out. Undefined actions configured at the rewrite policy level are carried out before a globally configured undefined action.

The NetScaler supports following three types of undefined actions:

### undefAction NOREWRITE

Aborts rewrite processing, but does not alter the packet flow. This means that the NetScaler continues to process requests and responses that do not match any rewrite policy, and eventually forwards them to the requested URL unless another feature intervenes and blocks or redirects the request. This action is appropriate for normal requests to your Web servers, and is the default setting.

### undefAction RESET

Resets the client connection. This means that the NetScaler tells the client that it must re-establish its session with the Web server. This action is appropriate for repeat requests for Web pages that do not exist, or for connections that might be attempts to hack or probe your protected Web site(s).

### undefAction DROP

Silently drops the request without responding to the client in any way. This means that the NetScaler simply discards the connection without responding to the client. This action is appropriate for requests that appear to be part of a DDoS attack or another sustained attack on your servers.

Note: Undefined events can be triggered for both request and response flow specific policies.

## To configure the default action by using the command line interface

At the command prompt, type the following commands to configure the default action and verify the configuration:

- o set rewrite param -undefAction ( NOREWRITE | RESET | DROP )
- o show rewrite param

### Example

```
> set rewrite param -undefAction NOREWRITE
Done
> show rewrite param
    Action Name: NOREWRITE
Done
```

## To configure the default action by using the configuration utility

1. Navigate to AppExpert > Rewrite.
2. In the details pane, under Rewrite Overview, click the Change Rewrite Settings link. The Set Rewrite Params dialog box appears.
3. Under Global Undefined-Result Action, select an option as follows:
  - o NoRewriteâ€”NOREWRITE
  - o Resetâ€”RESET
  - o Dropâ€”DROP
4. Click OK. The global undefined action is set to the value you chose.

## Bypassing the Safety Check

When you create a rewrite action, the NetScaler verifies that the expression you used to create the action is safe. Expressions created by the NetScaler from run-time data, such as URLs contained in HTTP requests, can cause unexpected errors. The NetScaler reports expressions that cause such errors as unsafe expressions.

In some cases, the expressions may be safe. For example, the NetScaler cannot validate an expression that contains a URL that does not resolve, even if the URL does not resolve because the Web server is temporarily unavailable. You can manually bypass the Safety Check to allow these expressions.

### To bypass the safety check by using the command line interface

At the command prompt, type the following commands to bypass the safety check and verify the configuration:

- set rewrite action <name> -bypassSafetyCheck YES
- show rewrite action <name>

#### Example

```
> set rewrite action insertact -bypassSafetyCheck YES
Done
> show rewrite action insertact

Name: insertact
Operation: insert_http_header    Target:Client-IP
Value:CLIENT.IP.SRC
BypassSafetyCheck : YES
Hits: 0
Undef Hits: 0
Action Reference Count: 2
Done
```

### To bypass safety check by using the configuration utility

1. Navigate to AppExpert > Rewrite > Actions.
2. In the details pane, select the rewrite action to be exempted from the safety check, and then click Open.
3. In the Configure Rewrite Action dialog box, select the Bypass Safety Check check box.
4. Click OK.

## Rewrite Action and Policy Examples

The examples in this section demonstrate how to configure rewrite to perform various useful tasks. The examples occur in the server room of Example Manufacturing Inc., a mid-sized manufacturing company that uses its Web site to manage a considerable portion of its sales, deliveries, and customer support.

Example Manufacturing has two domains: example.com for its Web site and email to customers, and example.net for its intranet. Customers use the Example Web site to place orders, request quotes, research products, and contact customer service and technical support.

As an important part of Example's revenue stream, the Web site must respond quickly and keep customer data confidential. Example therefore has several Web servers and uses Citrix NetScaler appliances to balance the Web site load and manage traffic to and from its Web servers.

The Example system administrators use the rewrite features to perform the following tasks:

### **Example 1: Delete old X-Forwarded-For and Client-IP Headers.**

Example Inc. removes old X-Forwarded-For and Client-IP HTTP headers from incoming requests.

### **Example 2: Adding a Local Client-IP Header.**

Example Inc. adds a new, local Client-IP header to incoming requests.

### **Example 3: Tagging Secure and Insecure Connections.**

Example Inc. tags incoming requests with a header that indicates whether the connection is a secure connection.

### **Example 4: Mask the HTTP Server Type.**

Example Inc. modifies the HTTP Server: header so that unauthorized users and malicious code cannot use that header to determine the HTTP server software it uses.

### **Example 5: Redirect an External URL to an Internal URL.**

Example Inc. hides information about the actual names of its Web servers and the configuration of its server room from users, to make URLs on its Web site shorter and easier to remember, and to improve security on its site.

### **Example 6: Migrating Apache Rewrite Module Rules.**

Example Inc. moved its Apache rewrite rules to a NetScaler appliance, translating the Apache PERL-based script syntax to the NetScaler rewrite rule syntax.

### **Example 7: Marketing Keyword Redirection.**

The marketing department at Example Inc. sets up simplified URLs for certain predefined keyword searches on the company's Web site.

### **Example 8: Redirect Queries to the Queried Server.**

Example Inc. redirects certain query requests to the appropriate server.

### **Example 9: Home Page Redirection.**

Example Inc. recently acquired a smaller competitor, and it now redirects requests for the acquired company's home page to a page on its own Web site.

Each of these tasks requires that the system administrators create rewrite actions and policies and bind them to a valid bind point on the NetScaler.

## Example 1: Delete Old X-Forwarded-For and Client-IP Headers

Example Inc. wants to remove old X-Forwarded-For and Client-IP HTTP headers from incoming requests, so that the only X-Forwarded-For headers that appear are the ones added by the local server. This configuration can be done through the NetScaler command line or the configuration utility. The Example Inc. system administrator is an old-school networking engineer and prefers to use a CLI where possible, but wants to be sure he understands the configuration utility interface so that he can show new system administrators on the team how to use it.

The examples below demonstrate how to perform each configuration with both the CLI and the configuration utility. The procedures are abbreviated on the assumption that users will already know the basics of creating rewrite actions, creating rewrite policies, and binding policies.

- For more detailed information about creating rewrite actions, see ["Configuring a Rewrite Action."](#)
- For more detailed information about creating rewrite policies, see ["Configuring a Rewrite Policy."](#)
- For more detailed information about binding rewrite policies, see ["Binding a Rewrite Policy."](#)

### To delete old X-Forwarded and Client-IP headers from a request by using the command line interface

At the command prompt, type the following commands in the order shown:

```
add rewrite action act_del_xfor delete_http_header x-forwarded-for
add rewrite action act_del_cip delete_http_header client-ip
add rewrite policy pol_check_xfor 'HTTP.REQ.HEADER("x-forwarded-for").EXISTS' act_del_xfor
add rewrite policy pol_check_cip 'HTTP.REQ.HEADER("client-ip").EXISTS' act_del_cip
bind rewrite global pol_check_xfor 100 200
bind rewrite global pol_check_cip 200 300
```

### To delete old X-Forwarded and Client-IP headers from a request by using the configuration utility

In the Create Rewrite Action dialog box, create two rewrite actions with the following descriptions.

Name	Type	Argument(s)
act_del_xfor	delete_http_header	x-forwarded-for
act_del_cip	delete_http_header	client-ip

In the Create Rewrite Policy dialog box, create two rewrite policies with the following descriptions.

Name	Expression	Action
pol_check_xfor	'HTTP.REQ.HEADER("x-forwarded-for").EXISTS'	act_del_xfor
pol_check_cip	'HTTP.REQ.HEADER("client-ip").EXISTS'	act_del_cip

Bind both policies to global, assigning the priorities and goto expression values shown below.

Name	Priority	Goto Expression
pol_check_xfor	100	200
pol_check_xfor	200	300

All old X-Forwarded-For and Client-IP HTTP headers are now deleted from incoming requests.



## Example 2: Adding a Local Client-IP Header

Example Inc. wants to add a local Client-IP HTTP header to incoming requests. This example contains two slightly different versions of the same basic task.

### To add a local Client-IP header by using the command line interface

At the command prompt, type the following commands in the order shown:

```
add rewrite action act_ins_client insert_http_header NS-Client 'CLIENT.IP.SRC'  
add rewrite policy pol_ins_client 'HTTP.REQ.HEADER("x-forwarded-for").EXISTS || HTTP.REQ.HEAD  
bind rewrite global pol_ins_client 300 END
```

### To add a local Client-IP header by using the configuration utility

In the Create Rewrite Action dialog box, create a rewrite action with the following description.

Name	Type	Argument(s)
act_ins_client	insert_http_header	NS-Client 'CLIENT.IP.SRC'

In the Create Rewrite Policy dialog box, create a rewrite policy with the following description.

Name	Expression	Action
pol_ins_client	'HTTP.REQ.HEADER("x-forwarded-for").EXISTS    HTTP.REQ.HEADER("client-ip").EXISTS'	act_ins_client

Bind both policies to global, assigning the priorities and goto expression values shown below.

Name	Priority	Goto Expression
pol_check_xfor	100	200
pol_check_xfor	200	300

A local Client-IP HTTP header is now added to incoming requests. You can also modify the configuration above to append all IPs from X-Forwarded-For headers to the new Client-IP header, as shown below.

## Example 3: Tagging Secure and Insecure Connections

Example Inc. wants to tag incoming requests with a header that indicates whether or not the connection is a secure connection. This helps the server keep track of secure connections after the NetScaler has decrypted the connections.

To implement this configuration, you would begin by creating rewrite actions with the values shown in the following tables. These actions label connections to port 80 as insecure connections, and connections to port 443 as secure connections.

Action Name	Type of Rewrite Action	Header Name	Value
Action-Rewrite-SSL_YES	INSERT_HTTP_HEADER	SSL	YES
Action Name	Type of Rewrite Action	Header Name	Value
Action-Rewrite-SSL_NO	INSERT_HTTP_HEADER	SSL	NO

You would then create a rewrite policy with the values shown in the following tables. These policies check incoming requests to determine which requests are directed to port 80 and which are directed to port 443. The policies then add the correct SSL header.

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-SSL_YES	Action-Rewrite-SSL_YES	NOREWRITE	CLIENT.TCP.DSTPORT.EQ(443)
Policy-Rewrite-SSL_NO	Action-Rewrite-SSL_NO	NOREWRITE	CLIENT.TCP.DSTPORT.EQ(80)

Finally, you would bind the rewrite policies to NetScaler, assigning the first policy a priority of 200, and the second a priority of 300, and setting the goto expression of both policies to END.

Each incoming connection to port 80 now has an SSL:NO HTTP header added to it and each incoming connection to port 443 has an SSL:YES HTTP header added to it.

## Example 4: Mask the HTTP Server Type

Example Inc. wants to modify the HTTP Server: header so that unauthorized users and malicious code cannot use the header to identify the software that the HTTP server uses.

To modify the HTTP Server: header, you would create a rewrite action and a rewrite policy with the values in the following tables.

Action Name	Type of Rewrite Action	Expression to choose target reference	String expression for replacement text
Action-Rewrite-Server_Mask	REPLACE	HTTP.RES.HEADER("Server")	"Web Server 1.0"

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-Server_Mask	Action-Rewrite-Server_Mask	NOREWRITE	HTTP.RES.IS_VALID

You would then globally bind the rewrite policy, assigning a priority of 100 and setting the Goto Priority Expression of the policy to END.

The HTTP Server: header is now modified to read "Web Server 1.0," masking the actual HTTP server software used by the Example Inc. Web site.

## Example 5: Redirect an External URL to an Internal URL

Example Inc. wants to hide its actual server room configuration from users to improve security on its Web servers.

To do this, you would create a rewrite action with the values as shown in the following tables. For request headers, the action in the table modifies `www.example.com` to `web.hq.example.net`. For response headers, the action does the opposite, translating `web.hq.example.net` to `www.example.com`.

Action Name	Type of Rewrite Action	Expression to choose target reference	String expression for replacement text
Action-Rewrite-Request_Server_Replace	REPLACE	HTTP.REQ.HOSTNAME.SERVER	"Web.hq.example.net"
Action-Rewrite-Response_Server_Replace	REPLACE	HTTP.RES.HEADER("Server")	"www.example.com"

Next, you would create rewrite policies using the values shown in the following tables. The first policy checks incoming requests to see if they are valid, and if they are, it performs the Action-Rewrite-Request\_Server\_Replace action. The second policy checks responses to see if they originate at the server `web.hq.example.net`. If they do, it performs the Action-Rewrite-Response\_Server\_Replace action.

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-Request_Server_Replace	Action-Rewrite-Request_Server_Replace	NOREWRITE	HTTP.REQ.HOSTNAME.SERVER
Policy-Rewrite-Response_Server_Replace	Action-Rewrite-Response_Server_Replace	NOREWRITE	HTTP.RES.HEADER("Server")

Finally, you would bind the rewrite policies, assigning each a priority of 500 because they are in different policy banks and therefore will not conflict. You should set the goto expression to NEXT for both bindings.

All instances of `www.example.com` in the request headers are now changed to `web.hq.example.net`, and all instances of `web.hq.example.net` in response headers are now changed to `www.example.com`.

## Example 6: Migrating Apache Rewrite Module Rules

Example Inc., is currently using the Apache rewrite module to process search requests sent to its Web servers and redirect those requests to the appropriate server on the basis of information in the request URL. Example Inc. wants to simplify its setup by migrating these rules onto the NetScaler platform.

Several Apache rewrite rules that Example currently uses are shown below. These rules redirect search requests to a special results page if they do not have a SiteID string or if they have a SiteID string equal to zero (0), or to the standard results page if these conditions do not apply.

The following are the current Apache rewrite rules:

- RewriteCond %{REQUEST\_FILENAME} ^/search\$ [NC]
- RewriteCond %{QUERY\_STRING} !SiteID= [OR]
- RewriteCond %{QUERY\_STRING} SiteID=0
- RewriteCond %{QUERY\_STRING} CallName=DisplayResults [NC]
- RewriteRule ^.\*\$ /results2.html [P,L]
- RewriteCond %{REQUEST\_FILENAME} ^/search\$ [NC]
- RewriteCond %{QUERY\_STRING} CallName=DisplayResults [NC]
- RewriteRule ^.\*\$ /results.html [P,L]

To implement these Apache rewrite rules on the NetScaler, you would create rewrite actions with the values in the following tables.

Action Name	Type of Rewrite Action	Expression to choose target reference	String expression for replacement text
Action-Rewrite-Display_Results_NulSiteID	REPLACE	HTTP.REQ.URL	"/results2.html"
Action-Rewrite-Display_Results	REPLACE	HTTP.REQ.URL	"/results2.html"

You would then create rewrite policies with the values as shown in the tables below.

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-Display_Results_NulSiteID	Action-Rewrite-Display_Results_NulSiteID	NOREWRITE	HTTP.REQ.URL.PA` QUERY.CONTAINS QUERY.SET_TEXT_
Policy-Rewrite-Display_Results	Action-Rewrite-Display_Results	NOREWRITE	HTTP.REQ.URL.PA` SET_TEXT_MODE(I

Finally, you would bind the rewrite policies, assigning the first a priority of 600 and the second a priority of 700, and then set the goto expression to NEXT for both bindings.

The NetScaler now handles these search requests exactly as the Web server did before the Apache rewrite module rules were migrated.

## Example 7: Marketing Keyword Redirection

The marketing department at Example Inc. wants to set up simplified URLs for certain predefined keyword searches on the company's Web site. For these keywords, it wants to redefine the URL as shown below.

- External URL: `http://www.example.com/<marketingkeyword>`
- Internal URL: `http://www.example.com/go/kwsearch.asp?keyword=<marketingkeyword>`

To set up redirection for marketing keywords, you would create a rewrite action with the values in the following table.

Action Name	Type of Rewrite Action	Expression to choose target location	String expression for replacement text
Action-Rewrite-Modify_URL	INSERT_BEFORE	HTTP.REQ.URL.PATH.GET(1)	"/go/kwsearch.aspkeyword="

You would then create a rewrite policy with the values in the following table.

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-Modify_URL	Action-Rewrite-Modify_URL	NOREWRITE	HTTP.REQ.HOSTNAME.SERVER.EQ("www.example.com")

Finally, you would bind the rewrite policy, assigning it a priority of 800. Unlike the previous rewrite policies, this policy should be the last to be applied to a request that matches its criteria. For this reason, NetScaler administrator sets its Goto Priority Expression to END.

Any request using a marketing keyword is redirected to the keyword search CGI page, whereupon a search is performed and all remaining policies are skipped.

## Example 8: Redirect Queries to the Queried Server

Example Inc. wants to redirect query requests to the appropriate server, as shown here.

- Request: GET /query.cgi?server=5HOST: www.example.com
- Redirect URL: http://web-5.example.com/

To implement this redirection, you would first create a rewrite action with the values in the following table.

Action Name	Type of Rewrite Action	Expression to choose target reference	String expression text
Action-Rewrite-Replace_Hostheader	REPLACE	HTTP.REQ.HEADER("Host").BEFORE_STR(".example.com")	"server-" + VALUE("v

You would then create a rewrite policy with the values in the following table.

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-Replace_Hostheader	Action-Rewrite-Replace_Hostheader	NOREWRITE	HTTP.REQ.HEADER("Host").EC

Finally, you would bind the rewrite policy, assigning it a priority of 900. Because this policy should be the last policy applied to a request that matches its criteria, you set the goto expression to END.

Incoming requests to any URL that begins with `http://www.example.com/query.cgi?server=` are redirected to the server number in the query.

## Example 9: Home Page Redirection

New Company, Inc. recently acquired a smaller competitor, Purchased Company, and wants to redirect the home page for Purchased Company to a new page on its own Web site, as shown here.

- Old URL: `http://www.purchasedcompany.com/*`
- New URL: `http://www.newcompany.com/products/page.htm`

To redirect requests to the Purchased Company home page, you would create rewrite actions with the values in the following table.

Action Name	Type of Rewrite Action	Expression to choose target reference	String expression for replacement text
Action-Rewrite-Replace_URLr	REPLACE	HTTP.REQ.URL.PATH_AND_QUERY	"/products/page.htm"
Action-Rewrite-Replace_Host	REPLACE	HTTP.REQ.HOSTNAME	"www.newcompany.com"

You would then create rewrite policies with the values in the following table.

Policy Name	Action Name	Undefined Action	Expression
Policy-Rewrite-Replace-None	Action-Rewrite-Replace-None	NOREWRITE	!HTTP.REQ.HOSTNAME.SERVER.EQ("www.com")
Policy-Rewrite-Replace-Host	Action-Rewrite-Replace_Host	NOREWRITE	HTTP.REQ.HOSTNAME.SERVER.EQ("www.
Policy-Rewrite-Replace-URL	Action-Rewrite-Replace_URL	NOREWRITE	HTTP.REQ.IS_VALID

Finally, you would bind the rewrite policies globally, assigning the first a priority of 100, the second a priority of 200, and the third a priority of 300. These policies should be the last policies applied to a request that matches the criteria. For this reason, set the goto expression to END for the first and third policies, and to 300 for the second policy. This ensures that all remaining requests are processed correctly.

Requests to the acquired company's old Web site are now redirected to the correct page on the New Company home page.



## URL Transformation

The URL transformation feature provides a method for modifying all URLs in designated requests from an external version seen by outside users to an internal URL seen only by your Web servers and IT staff. You can redirect user requests seamlessly, without exposing your network structure to users. You can also modify complex internal URLs that users may find difficult to remember into simpler, more easily remembered external URLs.

Note: Before you can use the URL transformation feature, you must enable the Rewrite feature. To enable the Rewrite feature, see [Enabling the Rewrite Feature](#).

To begin configuring URL transformation, you create profiles, each describing a specific transformation. Within each profile, you create one or more actions that describe the transformation in detail. Next, you create policies, each of which identifies a type of HTTP request to transform, and you associate each policy with an appropriate profile. Finally, you globally bind each policy to put it into effect.

## Configuring URL Transformation Profiles

A profile describes a specific URL transformation as a series of actions. The profile functions primarily as a container for the actions, determining the order in which the actions are performed. Most transformations transform an external hostname and optional path into a different, internal hostname and path. Most useful transformations are simple and require only a single action, but you can use multiple actions to perform complex transformations.

You cannot create actions and then add them to a profile. You must create the profile first, and then add actions to it. In the CLI, creating an action and configuring the action are separate steps. Creating a profile and configuring the profile are separate steps in both the CLI and the configuration utility.

### To create a URL transformation profile by using the NetScaler command line

At the NetScaler command prompt, type the following commands, in the order shown, to create a URL transformation profile and verify the configuration. You can then repeat the second and third commands to configure additional actions:

- o add transform profile <profileName> -type URL [-onlyTransformAbsURLInBody (ON|OFF)] [-comment <comment>]
- o add transform action <name> <profileName> <priority>
- o set transform action <name> [-priority <priority>] [-reqUrlFrom <expression>] [-reqUrlInto <expression>] [-resUrlFrom <expression>] [-resUrlInto <expression>] [-cookieDomainFrom <expression>] [-cookieDomainInto <expression>] [-state (ENABLED|DISABLED)] [-comment "<string>"]
- o show transform profile <name>

#### Example

```
> add transform profile shoppingcart -type URL
Done
> add transform action actshopping shoppingcart 1000
Done
> set transform action actshopping -priority 1000 -reqUrlFrom 'shopping.example.com' -reqUrl
Done
> show transform profile shoppingcart
    Name: shoppingcart
        Type: URL           onlyTransformAbsURLInBody: OFF
    Comment:
    Actions:

1)          Priority 1000   Name: actshopping           ENABLED
Done
```

### To modify an existing URL transformation profile or action by using the NetScaler command line

At the NetScaler command prompt, type the following commands to modify an existing URL transformation profile or action and verify the configuration:

Note: Use a set transform profile or set transform action command, respectively. The set transform profile command takes the same arguments as does the add transform profile command, and set transform action is the same command that was used for initial configuration.

- o set transform action <name> [-priority <priority>] [-reqUrlFrom <expression>] [-reqUrlInto <expression>] [-resUrlFrom <expression>] [-resUrlInto <expression>] [-cookieDomainInto <expression>] [-state (ENABLED|DISABLED)] [-comment "<string>"]
- o show transform profile <name>

#### Example

```
> set transform action actshopping -priority 1000 -reqUrlFrom 'searching.example.net' -reqUr
Done
> show transform profile shoppingcart
    Name: shoppingcart
        Type: URL           onlyTransformAbsURLInBody: OFF
    Comment:
    Actions:

1)          Priority 1000   Name: actshopping           ENABLED
Done
```

## To remove a URL transformation profile and actions by using the NetScaler command line

First remove all actions associated with that profile by typing the following command once for each action:

- o `rm transform action <name>` After you have removed all actions associated with a profile, remove the profile as shown below.
- o `rm transform profile <name>`

## To create a URL transformation profile by using the configuration utility

1. In the navigation pane, expand Rewrite, expand URL Transformation, and then click Profiles.
2. In the details pane, click Add.
3. In the Create URL Transformation Profile dialog box, type or select values for the parameters. The contents of the dialog box correspond to the parameters described in "Parameters for configuring URL transformation profiles" as follows (asterisk indicates a required parameter):
  - o Name\*â€"name
  - o Commentâ€"comment
  - o Only transform absolute URLs in response bodyâ€"onlyTransformAbsURLinBody
4. Click Create, and then click Close. A message appears in the status bar, stating that the Profile has been configured successfully.

## To configure a URL transformation profile and actions by using the configuration utility

1. In the navigation pane, expand Rewrite, expand URL Transformation, and then click Profiles.
2. In the details pane, select the profile you want to configure, and then click Open.
3. In the Configure URL Transformation Profile dialog box, do one of the following.
  - o To create a new action, click Add.
  - o To modify an existing action, select the action, and then click Open.
4. Fill in the Create URL Transformation Action or Modify URL Transformation Action dialog box by typing or selecting values for the parameters. The contents of the dialog box correspond to the parameters described in "Parameters for configuring URL transformation profiles" as follows (asterisk indicates a required parameter):
  - o Action Name\*â€"name
  - o Commentsâ€"comment
  - o Priority\*â€"priority
  - o Request URL fromâ€"reqUrlFrom
  - o Request URL intoâ€"reqUrlInto
  - o Response URL fromâ€"resUrlFrom
  - o Response URL intoâ€"resUrlInto
  - o Cookie Domain fromâ€"cookieDomainFrom
  - o Cookie Domain intoâ€"cookieDomainInto
  - o Enabledâ€"state
5. Save your changes.
  - o If you are creating a new action, click Create, and then Close.
  - o If you are modifying an existing action, click OK.

A message appears in the status bar, stating that the Profile has been configured successfully.
6. Repeat step 3 through step 5 to create or modify any additional actions.
7. To delete an action, select the action, and then click Remove. When prompted, click OK to confirm the deletion.
8. Click OK to save your changes and close the Modify URL Transformation Profile dialog box.
9. To delete a profile, in the details pane select the profile, and then click Remove. When prompted, click OK to confirm the deletion.

## Configuring URL Transformation Policies

After you create a URL transformation profile, you next create a URL transformation policy to select the requests and responses that the NetScaler should transform by using the profile. URL transformation considers each request and the response to it as a single unit, so URL transformation policies are evaluated only when a request is received. If a policy matches, the NetScaler transforms both the request and the response.

Note: The URL transformation and rewrite features cannot both operate on the same HTTP header during request processing. Because of this, if you want to apply a URL transformation to a request, you must make sure that none of the HTTP headers it will modify are manipulated by any rewrite action.

### To configure a URL transformation policy by using the NetScaler command line

You must create a new policy. On the command line, an existing policy can only be removed. At the NetScaler command prompt, type the following commands to configure a URL transformation policy and verify the configuration:

- o add transform policy <name> <rule> <profileName>
- o show transform policy <name>

#### Example

```
> add transform policy polsearch HTTP.REQ.URL.SUFFIX.EQ("Searching") prosearching
Done
> show transform policy polsearch
1)      Name: polsearch
        Rule: HTTP.REQ.URL.SUFFIX.EQ("Searching")
        Profile: prosearching
        Priority: 0
        Hits: 0
Done
```

### To remove a URL transformation policy by using the NetScaler command line

At the NetScaler command prompt, type the following command to remove a URL transformation policy:

```
rm transform policy <name>
```

#### Example

```
> rm transform policy polsearch
Done
```

### To configure a URL transformation policy by using the configuration utility

1. In the navigation pane, expand Rewrite, expand URL Transformation, and then click Policies.
2. In the details pane, do one of the following:
  - o To create a new policy, click Add.
  - o To modify an existing policy, select the policy, and then click Open.
3. In the Create URL Transformation Policy or Configure URL Transformation Policy dialog box, type or select values for the parameters. The contents of the dialog box correspond to the parameters described in "Parameters for configuring URL transformation policies" as follows (asterisk indicates a required parameter):
  - o Name\*â€"name (Cannot be changed for a previously configured policy.)
  - o Profile\*â€"profileName
  - o Expressionâ€"rule

If you want help with creating an expression for a new policy, you can either hold down the **Control** key and press the **space bar** while your cursor is in the Expression text box. To create the expression, you can type it directly as described below, or you can use the Add Expression dialog box as described in [To add an expression by using the Add Expression dialog box](#).

- a. Click Prefix, and choose the prefix for your expression.

Your choices are:

- o HTTPâ€"The HTTP protocol. Choose this if you want to examine some aspect of the request that pertains to the HTTP protocol.

- o SYSâ€”The protected Web site(s). Choose this if you want to examine some aspect of the request that pertains to the recipient of the request.
- o CLIENTâ€”The computer that sent the request. Choose this if you want to examine some aspect of the sender of the request.
- o SERVERâ€”The computer to which the request was sent. Choose this if you want to examine some aspect of the recipient of the request.
- o URLâ€”The URL of the request. Choose this if you want to examine some aspect of the URL to which the request was sent.
- o TEXTâ€”Any text string in the request. Choose this if you want to examine a text string in the request.
- o TARGETâ€”The target of the request. Choose this if you want to examine some aspect of the request target.

After you choose a prefix, the NetScaler displays a two-part prompt window that displays the possible next choices at the top, and a brief explanation of what the selected choice means at the bottom. The choices depend on which prefix you chose.

b. Select your next term.

If you chose HTTP as your prefix, your choices are REQ, which specifies HTTP requests, and RES, which specifies HTTP responses. If you chose another prefix, your choices are more varied. For help on a specific choice, click that choice once to display information about it in the lower prompt window.

When you are certain which choice you want, double-click it to insert it into the Expression window.

- c. Type a period, and then continue selecting terms from the list boxes that appear to the right of the previous list box. You type the appropriate text strings or numbers in the text boxes that appear to prompt you to enter a value, until your expression is finished.
4. Click Create or OK, depending on whether you are creating a new policy or modifying an existing policy.
  5. Click Close. A message appears in the status bar, stating that the Policy has been configured successfully.

## To add an expression by using the Add Expression dialog box

1. In the Create Responder Action or Configure Responder Action dialog box, click Add.
2. In the Add Expression dialog box, in the first list box choose the first term for your expression.

HTTP

The HTTP protocol. Choose this if you want to examine some aspect of the request that pertains to the HTTP protocol.

SYS

The protected Web site(s). Choose this if you want to examine some aspect of the request that pertains to the recipient of the request.

CLIENT

The computer that sent the request. Choose this if you want to examine some aspect of the sender of the request.

SERVER

The computer to which the request was sent. Choose this if you want to examine some aspect of the recipient of the request.

URL

The URL of the request. Choose this if you want to examine some aspect of the URL to which the request was sent.

TEXT

Any text string in the request. Choose this if you want to examine a text string in the request.

TARGET

The target of the request. Choose this if you want to examine some aspect of the request target.

When you make your choice, the rightmost list box lists appropriate terms for the next part of your expression.

3. In the second list box, choose the second term for your expression. The choices depend upon which choice you made in the previous step, and are appropriate to the context. After you make your second choice, the Help window below the Construct Expression window (which was blank) displays help describing the purpose and use of the term you just chose.
4. Continue choosing terms from the list boxes that appear to the right of the previous list box, or typing strings or numbers in the text boxes that appear to prompt you to enter a value, until your expression is finished.

## Globally Binding URL Transformation Policies

After you have configured your URL transformation policies, you bind them to Global or a bind point to put them into effect. After binding, any a request or response that matches a URL transformation policy is transformed by the profile associated with that policy.

When you bind a policy, you assign a priority to it. The priority determines the order in which the policies you define are evaluated. You can set the priority to any positive integer. In the NetScaler OS, policy priorities work in reverse order - the higher the number, the lower the priority.

Because the URL transformation feature implements only the first policy that a request matches, not any additional policies that it might also match, policy priority is important for achieving the results that you intend. If you give your first policy a low priority (such as 1000), you tell the NetScaler to perform it only if other policies with a higher priority do not match a request. If you give your first policy a high priority (such as 1), you tell the NetScaler to perform it first, and skip any other policies that might also match. You can leave yourself plenty of room to add other policies in any order, without having to reassign priorities, by setting priorities with intervals of 50 or 100 between each policy when you globally bind your policies.

Note: URL transformation policies cannot be bound to TCP-based virtual servers.

### To bind a URL transformation policy by using the NetScaler command line

At the NetScaler command prompt, type the following commands to globally bind a URL transformation policy and verify the configuration:

- bind transform global <policyName> <priority>
- show transform global

#### Example

```
> bind transform global polisearching 100
Done
> show transform global
1)      Policy Name: polisearching
        Priority: 100

Done
```

### To bind a URL transformation policy by using the configuration utility

1. In the navigation pane, expand Rewrite, then expand URL Transformation, and then click Policies.
2. In the details pane, click Policy Manager.
3. In the Transform Policy Manager dialog box, choose the bind point to which you want to bind the policy. The choices are:
  - **Override Global.** Policies that are bound to this bind point process all traffic from all interfaces on the NetScaler appliance, and are applied before any other policies.
  - **LB Virtual Server.** Policies that are bound to a load balancing virtual server are applied only to traffic that is processed by that load balancing virtual server, and are applied before any Default Global policies. After selecting LB Virtual Server, you must also select the specific load balancing virtual server to which you want to bind this policy.
  - **CS Virtual Server.** Policies that are bound to a content switching virtual server are applied only to traffic that is processed by that content switching virtual server, and are applied before any Default Global policies. After selecting CS Virtual Server, you must also select the specific content switching virtual server to which you want to bind this policy.
  - **Default Global.** Policies that are bound to this bind point process all traffic from all interfaces on the NetScaler appliance.
  - **Policy Label.** Policies that are bound to a policy label process traffic that the policy label routes to them. The policy label controls the order in which policies are applied to this traffic.
4. Select Insert Policy to insert a new row and display a drop-down list with all available, unbound URL transformation policies.
5. Select the policy you want to bind, or select New Policy to create a new policy. The policy that you selected or created is inserted into the list of globally bound URL transformation policies.
6. Make any additional adjustments to the binding.
  - To modify the policy priority, click the field to enable it, and then type a new priority. You can also select Regenerate Priorities to renumber the priorities evenly.

- To modify the policy expression, double click that field to open the Configure Transform Policy dialog box, where you can edit the policy expression.
  - To set the Goto Expression, double click field in the Goto Expression column heading to display the drop-down list, where you can choose an expression.
  - To set the Invoke option, double click field in the Invoke column heading to display the drop-down list, where you can choose an expression
7. Repeat steps 3 through 6 to add any additional URL transformation policies you want to globally bind.
  8. Click OK to save your changes. A message appears in the status bar, stating that the Policy has been configured successfully.

## RADIUS Support for the Rewrite Feature

The NetScaler expressions language includes expressions that can extract information from and manipulate RADIUS messages in requests and responses. These expressions enable you to use the rewrite feature to modify portions of a RADIUS message before sending it to its destination. Your rewrite policies and actions can use any expression that is appropriate or relevant to a RADIUS message. The available expressions enable you to identify the RADIUS message type, extract any attribute-value pair (AVP) from the connection, and modify RADIUS AVPs. You can also create policy labels for RADIUS connections.

You can use the new RADIUS expressions in Rewrite rules for a number of purposes. For example, you could:

- Remove the domain\ portion of the RADIUS user-name AVP to simplify single sign-on (SSO).
- Insert a vendor-specific AVP, such as the MSISDN field used in telephone company operations to contain subscriber information.

You can also create policy labels to route specific types of RADIUS requests through a series of policies that are appropriate to those requests.

Note: RADIUS for Rewrite has the following limitations:

- The NetScaler ADC does not re-sign rewritten RADIUS requests or responses. If the RADIUS authentication server requires signed RADIUS messages, authentication will fail.
- The currently available RADIUS expressions do not work with RADIUS IPv6 attributes.

The NetScaler documentation for expressions that support RADIUS assumes familiarity with the basic structure and purpose of RADIUS communications. If you need more information about RADIUS, see your RADIUS server documentation or search online for an introduction to the RADIUS protocol.

## Configuring Rewrite Policies for RADIUS

The following procedure uses the NetScaler command line to configure a rewrite action and policy and bind the policy to a rewrite-specific global bind point.

### To configure a Rewrite action and policy, and bind the policy

At the command prompt, type the following commands:

- add rewrite action <actName> <actType>
- add rewrite policy <polName> <rule> <actName>
- bind rewrite policy <polName> <priority> <nextExpr> -type <bindPoint>  
where <bindPoint> represents one of the rewrite-specific global bind points.

## RADIUS Expressions for Rewrite

In a rewrite configuration, you can use the following NetScaler expressions to refer to various portions of a RADIUS request or response.

### Identifying the Type of Connection

RADIUS.IS\_CLIENT

Returns TRUE if the connection is a RADIUS client (request) message.

RADIUS.IS\_SERVER

Returns TRUE if the connection is a RADIUS server (response) message.

### Request Expressions

RADIUS.REQ.CODE

Returns the number that corresponds to the RADIUS request type. A derivative of the num\_at class. For example, a RADIUS access request would return 1 (one). A RADIUS accounting request would return 4.

RADIUS.REQ.LENGTH

Returns the length of the RADIUS request, including the header. A derivative of the num\_at class.

RADIUS.REQ.IDENTIFIER

Returns the RADIUS request identifier, a number assigned to each request that allows the request to be matched to the corresponding response. A derivative of the num\_at class.

RADIUS.REQ.AVP(<AVP Code No>).VALUE



Returns the value of first occurrence of this AVP as a string of type `text_t`.

**RADIUS.REQ.AVP(<AVP code no>).INSTANCE(instance number)**  
Returns the specified instance of the AVP as a string of type `RAVP_t`. A specific RADIUS AVP can occur multiple times in a RADIUS message. **INSTANCE (0)** returns the first instance, **INSTANCE (1)** returns second instance, and so on, up to sixteen instances.

**RADIUS.REQ.AVP(<AVP code no>).VALUE(instance number)**  
Returns the value of specified instance of the AVP as a string of type `text_t`.

**RADIUS.REQ.AVP(<AVP code no>).COUNT**  
Returns the number of instances of a specific AVP in a RADIUS connection, as an integer.

**RADIUS.REQ.AVP(<AVP code no>).EXISTS**  
Returns TRUE if the specified type of AVP exists in the message, or FALSE if it does not.

## Response Expressions

RADIUS response expressions are identical to RADIUS request expressions, except that **RES** replaces **REQ**.

## Typecasts of AVP Values

The ADC supports expressions to typecast RADIUS AVP values to the text, integer, unsigned integer, long, unsigned long, ipv4 address, ipv6 address, ipv6 prefix and time data types. The syntax is the same as for other NetScaler typecast expressions.

### Example

The ADC supports expressions to typecast RADIUS AVP values to the text, integer, unsigned integer, long, unsigned long, ipv4 address, ipv6 address, ipv6 prefix and time data types. The syntax is the same as for other NetScaler typecast expressions.

```
RADIUS.REQ.AVP(8).VALUE(0).typecast_ip_address_at
```

## AVP Type Expressions

The NetScaler ADC supports expressions to extract RADIUS AVP values by using the assigned integer codes described in RFC2865 and RFC2866. You can also use text aliases to accomplish the same task. Some examples follow.

**RADIUS.REQ.AVP (1).VALUE** or **RADIUS.REQ.USERNAME.value**  
Extracts the RADIUS user-name value.

**RADIUS.REQ.AVP (4). VALUE** or **RADIUS.REQ. ACCT\_SESSION\_ID.value**  
Extracts the Acct-Session-ID AVP (code 44) from the message.

**RADIUS.REQ.AVP (26). VALUE** or **RADIUS.REQ.VENDOR\_SPECIFIC.VALUE**  
Extracts the vendor-specific value.

The values of most commonly-used RADIUS AVPs can be extracted in the same manner.

## RADIUS Bind Points

Four global bind points are available for policies that contain RADIUS expressions.

**RADIUS\_REQ\_OVERRIDE**  
Priority/override request policy queue.

**RADIUS\_REQ\_DEFAULT**  
Standard request policy queue.

**RADIUS\_RES\_OVERRIDE**  
Priority/override response policy queue.

**RADIUS\_RES\_DEFAULT**  
Standard response policy queue.

## RADIUS Rewrite-Specific Expressions

**RADIUS.NEW\_AVP**  
Returns the specified RADIUS AVP as a string.

**RADIUS.NEW\_AVP\_INTEGER32**  
Returns the specified RADIUS AVP as an integer.

**RADIUS.NEW\_AVP\_UNSIGNED32**  
Returns the specified RADIUS AVP as an unsigned integer.

**RADIUS.NEW\_VENDOR\_SPEC\_AVP(<ID>, <definition>)**  
Adds the specified extended vendor specific AVPs to the connection. For <ID>, substitute a long number. For <definition>, substitute a string that contains the data for the AVP.

**RADIUS.REQ.AVP\_START**

**Example:**

**Example:**

**Example:**

398

## Diameter Support for Rewrite

The Rewrite feature now supports the Diameter protocol. You can configure Rewrite to modify Diameter requests and response as you would HTTP or TCP requests and responses, allowing you to use Rewrite to manage the flow of Diameter requests and make necessary modifications. For example, if the "Origin-Host" value in a Diameter request is inappropriate, you can use Rewrite to replace it with a value that is acceptable to the Diameter server.

### To configure Rewrite to modify a Diameter request

To configure the Rewrite feature to replace the Origin-Host in a diameter request with a different value, at the command prompt, type the following commands:

- o add rewrite action <actname> replace "DIAMETER.REQ.AVP(264,\"netscaler.example.net\") "  
For <actname>, substitute a name for your new action. The name can consist of from one to 127 characters in length, and can contain letters, numbers, and the hyphen (-) and underscore (\_) symbols. For `netscaler.example.net`, substitute the Host-Origin that you want to use instead of the original Host-Name.
- o add rewrite policy <polname> "diameter.req.avp(264).value.eq(\"host.example.com\")" <actname>  
For <polname>, substitute a name for your new policy. As with <actname>, the name can consist of from one to 127 characters in length, and can contain letters, numbers, and the hyphen (-) and underscore (\_) symbols. For `host.example.com`, substitute the name of the Host-Origin that you want to change. For <actname>, substitute the name of the action that you just created.
- o bind lb vserver <vservname> -policyName <polname> -priority <priority> -type REQUEST  
For <vservname>, substitute the name of the load balancing virtual server to which you want to bind the policy. For <polname>, substitute the name of the policy you just created. For <priority>, substitute a priority for the policy.

#### Example

To create a Rewrite action and policy to modify all Diameter Host-Origins of "host.example.com" to "netscaler.example.net", you could add the following action and policy, and bind the policy as shown.

```
> add rewrite action rw_act_replace_avp replace "diameter.req.avp(264)" "diameter.new.avp(264)"
> add rewrite policy rw_diam_pol "diameter.req.avp(264).value.eq(\"client.realm2.net\")" rw_act_replace_avp
> bind lb vserver vs1 -policyName rw_diam_pol -priority 10 -type REQUEST
```

Done

## String Maps

You can use string maps to perform pattern matching in all NetScaler features that use the default policy syntax. A string map is a NetScaler entity that consists of key-value pairs. The keys and values are strings in either ASCII or UTF-8 format. String comparison uses two new functions, `MAP_STRING(<string_map_name>)` and `IS_STRINGMAP_KEY(<string_map_name>)`.

A policy configuration that uses string maps performs better than one that does string matching through policy expressions, and you need fewer policies to perform string matching with a large number of key-value pairs. String maps are also intuitive, simple to configure, and result in a smaller configuration.

## How String Maps Work

String maps are similar in structure to pattern sets (a pattern set defines a mapping of index values to strings; a string map defines a mapping of strings to strings) and the configuration commands for string maps (commands such as `add`, `bind`, `unbind`, `remove`, and `show`) are syntactically similar to configuration commands for pattern sets. Also, as with index values in a pattern set, each key in a string map must be unique across the map. The following table illustrates a string map called `url_string_map`, which contains URLs as keys and values.

Table 1. String Map "url\_string\_map"

Key	Value
/url_1.html	http://www.redirect_url_1.com/url_1.html
/url_2.html	http://www.redirect_url_2.com/url_2.html
/url_3.html	http://www.redirect_url_1.com/url_1.html

The following table describes the two functions that have been introduced to enable string matching with keys in a string map. String matching is always performed with the keys. Additionally, the following functions perform a comparison between the keys in the string map and the complete string that is returned by the expression prefix. The examples in the descriptions refer to the preceding example.

Table 2. String Map Functions

Function	Description
<code>&lt;TEXT&gt;.MAP_STRING(&lt;string_map_name&gt;)</code>	<p>Checks whether the value returned by the expression prefix <code>TEXT</code> keys in the string map, and returns the value that corresponds to it. If no key in the string map matches the value returned by the expression prefix, the function returns an empty string. The <code>IGNORECASE</code> and <code>NOIGNORECASE</code> functions can be used to perform case-insensitive and case-sensitive comparison, respectively.</p> <p><b>Example 1:</b> <code>HTTP.REQ.URL.MAP_STRING("url_string_map")</code> returns the value of the key in the string map <code>url_string_map</code> that matches the value returned by <code>HTTP.REQ.URL</code>. If the value returned by <code>HTTP.REQ.URL</code> is <code>/url_1.html</code>, the function returns <code>http://www.redirect_url_1.com/url_1.html</code>.</p> <p><b>Example 2:</b></p> <p><code>HTTP.REQ.URL.SET_TEXT_MODE(IGNORECASE).MAP_STRING("url_string_map")</code> checks whether the string returned by <code>HTTP.REQ.URL</code> matches any of the keys in the string map <code>url_string_map</code>. The comparison does not consider case. If the value returned by <code>HTTP.REQ.URL</code> is <code>/URL_1.html</code>, the function returns <code>http://www.redirect_url_1.com/url_1.html</code>.</p> <p><b>Parameters:</b></p> <p><code>string_map_name</code> - The string map.</p>
<code>&lt;TEXT&gt;.IS_STRINGMAP_KEY(&lt;string_map_name&gt;)</code>	<p>Returns <code>TRUE</code> if the string returned by the expression prefix <code>TEXT</code> matches any of the keys in the string map. The <code>IGNORECASE</code> and <code>NOIGNORECASE</code> functions can be used to perform case-insensitive and case-sensitive string matching, respectively.</p> <p><b>Example 1:</b></p> <p><code>HTTP.REQ.URL.IS_STRINGMAP_KEY("url_string_map")</code> returns <code>TRUE</code> if the value returned by <code>HTTP.REQ.URL</code> is one of the keys in <code>url_string_map</code>.</p>

**Example 2:** `HTTP.REQ.URL.SET_TEXT_MODE(IGNORECASE).("url_string_map")` returns TRUE if the value of `HTTP.REQ.URL` is `/url_string_map`. In this case, key lookup does not consider case. `HTTP.REQ.URL` returns TRUE even if the value of `HTTP.REQ.URL` is `/URL_3.htm`.

**Parameters:**

`string_map_name` - The string map.

## Configuring a String Map

You first create a string map and then bind key-value pairs to it. You can create a string map from the command line interface (CLI) or the configuration utility.

### To configure a string map by using the command line interface

At the command prompt, do the following:

1. Create a string map.

```
add policy stringmap <name> -comment <string>
```

2. Bind a key-value pair to the string map.

```
bind policy stringmap <name> <key> <value>
```

Example:

```
> bind policy stringmap url_string_map1 "/url_1.html" "http://www.redirect_url_1.com/ur
```

### To configure a string map by using the configuration utility

Create a string map and bind the key-value pair to the created entity.

Navigate to **AppExpert > String Maps**, click **Add** and specify the relevant details.

## Example: Responder Policy With a Redirect Action

The following use case involves a responder policy with a redirect action. In the example below, the first four commands create the string map `url_string_map` and bind the three key-value pairs used in the earlier example. After creating the map and binding the key-value pairs, you create a responder action (`act_url_redirects`) that redirects the client to the corresponding URL in the string map or to `www.default.com`. You also configure a responder policy (`pol_url_redirects`) that checks whether requested URLs match any of the keys in `url_string_map` and then performs the configured action. Finally, you bind the responder policy to the content switching virtual server that receives the client requests that are to be evaluated.

```
add stringmap url_string_map

bind stringmap url_string_map /url_1.html http://www.redirect_url_1.com/url_1.html

bind stringmap url_string_map /url_2.html http://www.redirect_url_2.com/url_2.html

bind stringmap url_string_map /url_3.html http://www.redirect_url_1.com/url_1.html

add responder action act_url_redirects redirect 'HTTP.REQ.URL.MAP_STRING
("url_string_map") ALT "www.default.com"' -bypassSafetyCheck yes

add responder policy pol_url_redirects TRUE act_url_redirects

bind cs vserver csw_redirect -policyname pol_url_redirects -priority 1 -type request
```

