How to Use Extended Match and Condition Expressions

https://campus.barracuda.com/doc/3539018/

You can use Extended Match and Condition expressions in content rules, HTTP request rewrite rules, and HTTP response rewrite rules. To learn more about these rules, all of which only apply to Layer 7 - HTTP(S) Services, see the following:

- Directing HTTP Requests based on Content Rules
- Modifying HTTP Requests and Responses

In this article:

This article documents the syntax of the extended match and condition expressions. For example:

- `Header Host co example.com` - match a request whose Host header contains example.com
- `Parameter userid ex` - match any request in which the parameter 'userid' is present
- `(Header Host eq www.example.com) && (Client-IP eq 10.0.0.0/24)` - match a request whose host header is `www.example.com` and the requesting client's IP address is in the 10.0.0.* subnet.

Quick Reference

Expressions	• Element Match • (Expression) [Join (Expression) ...]
Joins	• &&, ||
Elements	• Request Elements: Method, HTTP-Version, Client-IP, URI, URI-Path, Header • Request Parameters: Parameter, Pathinfo • Response Elements: Status-code, Response-Header
Operators	• Matching: eq, neq, req, nreq • Containing: co, nco, rco, nrco • Existence: ex, nex

Structure

The following explains the components of an Extended Match or Condition expression.

An expression consists of one or more **Element Matches**, combined using Join operators to indicate AND and OR operations to combine the Element Matches. Parentheses must be used to delimit individual Element Matches when using join operators. Parentheses can be nested.

An Element Match consists of an **Element**, an optional **Element Name**, an **Operator** followed by an optional **Value**. Some elements like “Header” require an Element Name like “User-Agent”, whereas some elements like “HTTP-Version” require no further qualification. Also, some operators like “eq” (stands for “equals”) require a value, whereas some operators like “ex” (stands for “exists”) require no value.

Tokens are delimited by space and the parenthesis characters. Double quotes (") can be used to enclose single tokens which contain parenthesis characters or spaces. The back-slash character can also be used to escape, that is, remove the special meaning of the special characters (space and parentheses).

Operators

The following are the possible operators in an Element Match. The operators are case insensitive; for example, “eq”, “Eq” and “EQ” are all treated the same.

eq	True if the operand is equal to the given value. A case insensitive string comparison is performed. Thus, a value of “01” is not the same as a value of “1”, whereas values “one” and “ONE” are treated the same.
neq	True if the operand is not equal to the given value. A case insensitive string comparison is performed.
co	True if the operand contains the given value.
nco	True if the operand does not contain the given value.
rco	True if the operand contains the given value, which is treated as a regular expression.
nrco	True if the operand does not contain the given value, which is treated as a regular expression.
req	True if the operand matches the given value, which is treated as a regular expression.
nreq	True if the operand does not match the given value, which is treated as a regular expression.
ex	True if the operand exists. A value is not required.
nex	True if the operand does not exist. A value is not required.

Elements

The following are the different Elements allowed in the expression. Elements and Element Names are

case insensitive, so “Method” and “METHOD” are treated the same.

`Method`	The HTTP Method that was received in the request. Example: (`Method eq GET`)
`HTTP-Version`	This refers to the version of the HTTP protocol of the request. Example: (`HTTP-Version eq HTTP/1.1`)
`Header`	An HTTP header in the request. An Element Name to identify which header is required to follow the word “Header”. Example: (`Header Accept co gzip`). This checks if the “Accept:” header contains the string “gzip”.
`Client-IP`	This refers to the IP address of the client sending the request. The IP address can be either host IP address or subnet IP address specified by a mask. Only “eq” and “neq” operations are possible for this element. Examples: (`client-ip eq 192.168.1.0/24`), (`Client-IP eq 192.168.1.10`)
`URI`	The URI is the Uniform Resource Identifier in the request. This includes any query parameters in the request. Example: (`URI rco /abc.*html?userid=b`)
`URI-path`	This refers to the path portion of the URI, which excludes any query parameters. Example: (`URI-path req \/.*copy%20[^/]*`)
`Pathinfo`	This refers to the portion of URL which is interpreted as `PATH_INFO` on the server. The Barracuda Load Balancer uses a set of known extensions to determine whether a portion of the URL is a Pathinfo or not. For example, if the request URL is `/twiki/view.cgi/Engineering`, then, “`/Engineering`” is considered to be the `pathinfo` rather than part of the URL. Example: (`PathInfo rco abc*`)
`Parameter`	This refers to a parameter in the query string part of the URL. the servers as a name-value pair. The special parameter “`$NONAME_PARAM`” is used to refer to the case where the parameter name is absent. Examples: (`Parameter sid eq 1234`), (`Parameter $NONAME_PARAM co abcd`)
`Status-code`	This refers to the status code of the response returned by the servers. Example: (`status-code eq 302`)
`Response-header`	This refers to the HTTP response header in the response. The term “`Response-header`” should be followed by the name of the header on which the action is to be applied. Example: (Response-Header Set-Cookie co sessionid)

Each expression may use only some of these elements. The following restrictions apply:

- The Extended Match expression in the Content Rules can use these elements: Method, HTTP-Version, Header, Client-IP, URI, URI-Path, Pathinfo, and Parameters.
- Request Rewrite Condition allows these elements: Method, HTTP-Version, Header, Client-IP, Parameter, Pathinfo and URI.

- Response Rewrite Condition allows these elements: Header, Status-code and Response-Header.

Joins

Each expression can be joined with another expression by one of the following:

||	True if either of the expressions are true.
&&	True only if both the expressions are true.

Combining

More than one Element Match can be combined together by using the join operators || and && provided the Element Matches are enclosed in parentheses. Combining Element Matches *without* parentheses *is not allowed*.

Example: `(Header cookie ex) && (URI rco .*\.html) && (Method eq GET)`

Nested sub-expressions can be created by enclosing parentheses within expressions. This makes the expression more readable as well as unambiguous.

Example: `(HTTP-Version eq HTTP/1.1) && ((Header Host eq www.example.com) || (Header Host eq website.example.com))`

Escaping

The space character and the parentheses characters are special characters since they cause the parser to split the string into tokens at these separators. In some cases, it is required to specify these characters as part of the value itself. For example, the User-Agent header typically contains both spaces and parentheses, as in:

User-Agent: Mozilla/5.0 (Linux i686; en-US; rv:1.8.1.3) Firefox/2.0.0.3

The spaces and parenthesis characters in such cases must be escaped by prefixing these characters with a back-slash (\), or the entire value can be enclosed in double-quotes (“).

Examples:

- Header User-Agent eq “Mozilla/5.0 (Linux i686; en-US; rv:1.8.1.3) Firefox/2.0.0.3”
- Header User-Agent eq Mozilla/5.0\ \(Linux\ i686;\ en-US;\ rv:1.8.1.3\)\ Firefox/2.0.0.3

To specify the double-quote character itself, it must be escaped with a back-slash. This is true inside a quoted string, or a non-quoted string. Note that the single quote character has no special meaning, and is treated as any other character.

To specify the back-slash character itself, it must be escaped as “\\”. This is true within both quoted strings and non-quoted strings.

The back-slash character escapes all characters, not just the special characters. Thus, “\c” stands for the character “c” etc. In other words, back-slash followed by any character stands for the character, whether or not that character has a special meaning in the syntax.

Macro Definitions

The Barracuda Load Balancer supports several macros to assist in configuring policies. The following table describes these macros arranged by the areas where they can be used. The URI in these cases does not include the host.

$SRC_ADDR	Inserts the source (client) IP address. You can use it for the new value (Rewrite Value parameter) when inserting or rewriting a header.
$URI	Should be specified in the new value, if you are rewriting or redirecting the URI. $URI specifies the complete request URI including the query string.
$AUTH_USER	Adds the username. (1) (2) (3)
$AUTH_PASSWD	Adds the password. (1) (2) (3)
$AUTH_GROUPS	Adds the user roles. (1) (2) (3)
URL ACLs	
$NONAME_PARAM	Inserts a parameter with no name (see No Name Parameters)

Notes:
(1) The URL is not protected, i.e. access-control or authentication is off. The value substituted for the macro is the special string NCURLNotProtected.
(2) The client has not logged in. The value substituted for the macros is the special string NCNoUserSession.
(3) The user does not belong to any groups. The value substituted for $AUTH_GROUPS is the special string NCNOUserRoles.

No Name Parameters

There might be times when you want to configure a parameter without a name. For example, consider a site that pops up an advertising window when a user lands there. A Javascript adds a query string that results in the following GET request:

```
GET /ad?xyz
```

> The Barracuda Load Balancer does not learn “no name” parameters such as query strings like "`GET /ad?0`" added by a Javascript. Workaround: Add a null value URL ACL.

The Barracuda Load Balancer treats `xyz` as the value of a parameter. In this case, you cannot create an exception rule based on the `xyz` value because there is no way to associate it with a named parameter.

To address such situations (that is, requests with parameter name-value pairs of the type `?xyz` or `?=xyz` where `xyz` is the value), you can use a special token: `$NONAME_PARAM` (case insensitive). This token allows you to create an expression for a parameter without a name as in the following examples:

```
set = parameter $NONAME_PARAM ex

set = parameter $NONAME_PARAM eq 0

set = parameter $noname_param co xyz
```