BIG-IP® Local Traffic Management: Getting Started with Policies

Version 13.0

Table of Contents

Introducing Local Traffic Policies

About Local Traffic Policies

The BIG-IP® system provides Local Traffic Policies that simplify the way in which you can manage traffic associated with a virtual server. Using policies involves three basic steps: you create a draft policy, publish the policy, and then associate the published policy with a virtual server. Each policy includes a matching strategy for the specified rules, as well as conditions and actions configured within each rule, to manage traffic.

Note: *Local Traffic Policies that have been upgraded from BIG-IP software version 12.0, or earlier, appear in the Published Policies list.*

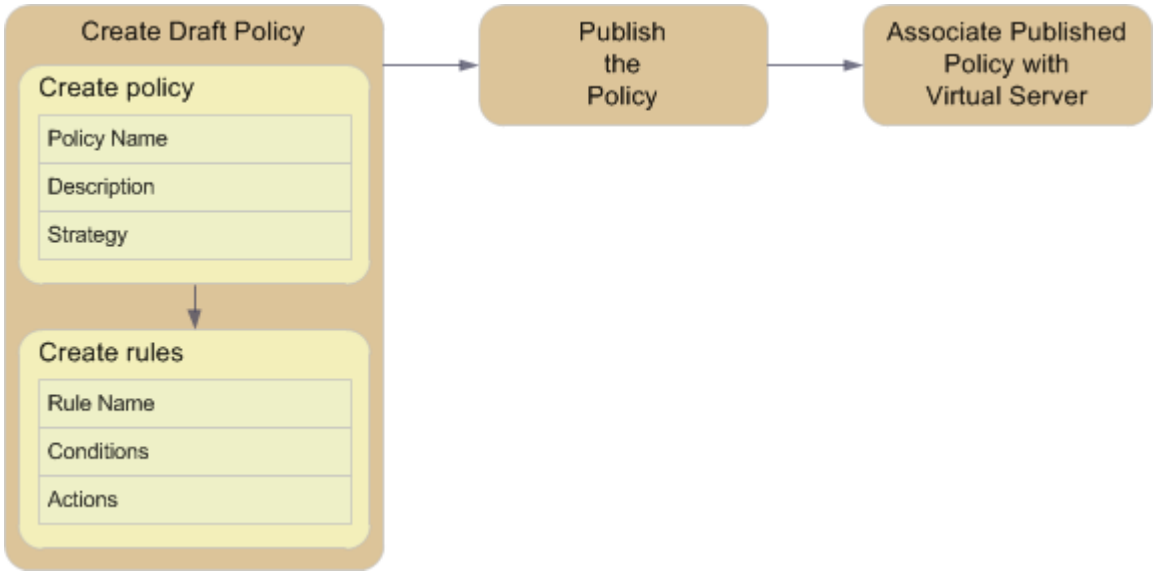

Figure 1: Basic steps for creating and using policies

About local traffic policy matching

BIG-IP® *local traffic policies* comprise a prioritized list of rules that match defined conditions and run specific actions, which you can associate with a virtual server that directs traffic accordingly. For example, you might create a policy that determines whether a client is using a mobile device, and then redirects its requests to the applicable mobile web site's URL.

About strategies for local traffic policy matching

Each BIG-IP® local traffic policy requires a matching strategy to determine which rule applies if more than one rule matches.

The BIG-IP local traffic policies provide three predefined policy matching strategies: a first-match, best-match, and all-match strategy. Each policy matching strategy prioritizes rules according to the rule's position within the Rules list.

As needed, you can create a user-defined best-match strategy to customize the precedence (order of preference) of added operands and selectors. For example, to meet your preferred operand and selector combinations, you might create a user-defined best-match strategy that changes the precedence of added operands and selectors, compared to the predefined best-match strategy.

Note: *In a best-match or first-match strategy, a rule without conditions becomes the default rule, when the rule is the last entry in the Rules list.*

Table 1: Policy matching strategies

Matching strategy	Description
all-match strategy	An *all-match strategy* starts the actions for all rules in the Rules list that match. ***Note:*** *In an all-match strategy, when multiple rules match, but specify conflicting actions, only the action of the best-match rule is implemented. A best-match rule can be the lowest ordinal, the highest priority, or the first rule that matches in the Rules list.*
best-match strategy	A *best-match strategy* selects and starts the actions of the rule in the Rules list with the best match, as determined by the following factors. **1.** A best-match strategy selects the rule with the most conditions, ignoring details about the conditions. **2.** If a rule with the most conditions is not determined, then the best-match strategy selects the rule with the highest priority condition types. The best-match strategy sorts the condition types, highest priority first, comparing one at a time until a higher priority is found. For example, a priority sequence of 0,1,3,4,6 wins over 0,1,3,5,7 because 4 is a higher priority than 5. **3.** If a rule with the highest priority condition types is not determined, then the best-match strategy selects the rule with equal match types over other match types, such as starts-with, ends-with, or contains, and processes according to condition type priority. **4.** If a rule of equal match types is not determined, then the best-match strategy uses an ordinal (the precedence of the operand). ***Note:*** *In a best-match strategy, when multiple rules match and specify an action, conflicting or otherwise, only the action of the best-match rule is implemented. A best-match rule can be the lowest ordinal, the highest priority, or the first rule that matches in the Rules list.*
first-match strategy	A *first-match strategy* starts the actions for the first rule in the Rules list that matches.

About rules for local traffic policy matching

BIG-IP® local traffic policy *rules* match defined conditions and start specific actions. You can create a policy with rules that are as simple or complex as necessary, based on the passing traffic. For example, a rule might simply determine that a client's browser is a Chrome browser that is not on an administrator network, and restrict access to certain administrative tools. Or a rule might determine that a request URL starts with `/video`, that the client is a mobile device, and that the client's subnet does not match `172.27.56.0/24`, and then that the request to a video file is designed for mobile devices on a Content Delivery Network (CDN).

About logical operators for conditions and rules

Local traffic policy rules provide you with different types of logical operators for matching conditions, which are determined by the order and configuration of the conditions within and between the rules. The different types of logical operators that you can configure are AND logical operators for conditions within a rule, and OR logical operators for values within a condition and for conditions between rules. When AND logical operators apply, then all logical operators must match the matching strategy. When OR logical operators apply, then any logical operator must match the matching strategy.

AND logical operators for conditions within a rule

When you create a rule, you can configure two or more conditions that use AND logic within that rule. For example, you can create Rule1 with two conditions, a and b, which use AND logic when Rule1 is used by the matching strategy. This means that all conditions within a rule must succeed in order to be used by a matching strategy.

OR logical operators for values within a condition and for conditions between rules

When you configure multiple values within a condition, OR logic determines if any matching value within the condition succeeds. For example, you can create a condition configured with two or more values. The matching strategy uses OR logic to determine if any configured value matches.

Similarly, when you create two or more rules, you can configure each rule with applicable conditions that use OR logic between the rules. For example, you can create Rule1 with a set of conditions, and Rule 2 with another set of conditions. The matching strategy uses OR logic to determine if a rule matches.

Examples

These examples show the logical operation of three conditions (a, b, and c) and two rules (Rule1 and Rule2).

In this first example, consider the following scenario, where you want to match condition a or b, and c ((a | b) & c). You can configure this logic by creating Rule1 to use conditions a and c (a & c), and Rule 2 to use conditions b and c (b & c). The result is when Rule1 matches the strategy, conditions a and c (a & c) are used, or when Rule 2 matches, conditions b and c (b & c) are used.

In this second example, consider the scenario where you want to match conditions a and b, or c ((a & b) | c). You can configure this logic by creating Rule1 to use conditions a and b, and Rule2 to use condition c. The result is when Rule1 matches the strategy, both conditions a and b are used, or when Rule2 matches the strategy, condition c is used.

About conditions for local traffic policy matching

The *conditions* for a local traffic policy rule define the necessary criteria that must be met in order for the rule's actions to be applied. For example, a policy might include the following condition type and settings, which, when met by a request, would allow the rule's specified actions to be applied.

Option	Setting
Condition Type	**HTTP Host**
Selector	**host**
Condition	**is**
Values	`www.siterequest.com`

You can apply one or more conditions to a rule, as needed.

Table 2: Conditions for local traffic policy matching

Condition Type	Description
Client SSL	Inspects the properties of the SSL connection on the client side of the device. • **cipher**. Specifies the cipher name. • **cipher bits**. Specifies the cipher strength by means of the number of bits. • **protocol**. Specifies the SSL protocol name.
CPU Usage	Specifies a condition that is determined by CPU usage during 15-second, 1-minute, or 5-minute intervals.
Geo. IP	Specifies a condition that is based on the properties of the geographical location of the IP address. • **continent**. Matches a two-character continent code, for example, AF (Africa), AN (Antarctica), AS (Asia), OC (Oceania), NA (North America), or SA (South America). • **country code**. Matches a two-character country code, as defined in ISO-3166-2. • **country name**. Matches the full name of a country. • **isp**. Matches the Internet Service Provider associated with the address. • **organization**. Matches the organization associated with the address. • **region code**. Matches the abbreviation of a state, province, or country-specific region. • **region name**. Matches the full name of a state, province, or country-specific region.
HTTP Basic Auth.	Inspects the username and password specified for basic authentication for the HTTP request. • **password**. Matches the basic authentication password. • **username**. Matches the basic authentication username.
HTTP Cookie	Inspects the Cookie header of an HTTP request.
HTTP Header	Matches any HTTP header.
HTTP Host	Matches the host of an HTTP request. • **host**. Matches the host name. • **port**. Matches the port number. • **full string**. Matches the full host header string.
HTTP Method	Inspects the HTTP method for the request, for example, GET, POST, or HEAD.
HTTP Referer	Inspects the HTTP Referer header or parts of the URI. • **extension**. Matches the document extension, for example, `cgi`. • **host**. Matches the DNS host name or IP address. • **path**. Matches the URI path, for example, `/path`. • **path segment**. Matches the path segment by numerical index. • **port**. Matches the numeric port number, for example, `80`. • **query parameter**. Matches the value of the query parameter by name. • **query string**. Matches the full query string, for example, `a=b&c=d`. • **scheme**. Matches the scheme, for example, http, https, or ftp.

Condition Type	Description
	• **unnamed query parameter**. Matches the value of the query parameter by numerical index. • **full string**. Matches the full URI, for example, `http://example.com/path/to/page.cgi?a-b&c=d`.
HTTP Set Cookie	Inspects the Set-Cookie header of an HTTP response. • **domain**. Matches the value of the domain specified by the named cookie. • **expiry**. Matches the time when validity of the named cookie expires in RFC 6265 format (`Wdy, DD Mon YYYY HH:MM:SS GMT`). • **path**. Matches the path of the named cookie. • **value**. Matches the value of the named cookie specified by the parameter. • **version**. Matches the version of the named cookie.
HTTP Status	Inspects the status of the HTTP response. • **code**. Matches the numeric HTTP response status code. • **text**. Matches the HTTP response status string, for example, `Authentication Required`. • **full string**. Matches the full HTTP status response, including the code and text.
HTTP URI	Inspects the URI on a request and matches parts of or the entire URI. • **extension**. Matches the file extension in the URI, for example, `html` or `cgi`. • **host**. Matches the host name in the URI. • **path**. Matches the URI path. • **path segment**. Matches a part of the URI path by a numeric index. • **port**. Matches the port number in the URI. • **query parameter**. Matches the value of the named query parameter from the query string. • **query string**. Matches the text specified in the query string. • **scheme**. Matches the scheme, for example, http, https, ftp, or file. • **unnamed query parameter**. Matches the value of a query parameter by a numeric index instead of by a name index. • **full string**. Matches a specified full text string.
HTTP User Agent	Specifies a condition that is based upon the User Agent header. • **browser type**. Matches the browser name or type. • **browser version**. Matches the browser version string. • **device make**. Matches the make of the device. • **device model**. Matches the model of the device. • **token**. Matches a string associated with a specified parameter.
HTTP Version	Inspects the version of an HTTP request or response. • **major**. Matches the numeric major part of the HTTP version. • **minor**. Matches the numeric minor part of the HTTP version. • **protocol**. Matches the HTTP protocol. • **full string**. Matches the full version string.
SSL Certificate	Inspects the properties of an SSL certificate.
SSL Extension	Inspects the SSL extensions that are negotiated during the `HELLO` phase.

Condition Type	Description
	• **alpn**. Matches the application layer protocol negotiation. • **npn**. Matches the next protocol negotiation. • **server name**. Matches the server name indication.
TCP	Inspects and matches the parameters associated with TCP connections. • **address**. Matches the specified IP address. The IP address is the address associated with the external interface remote end of the connection. • **mss**. Compares the TCP maximum segment size on the external network interface. • **port**. Matches the port number. The specified port is associated with the external interface, remote end of the connection. • **route domain**. Compares traffic with the specified route domain number on the external network interface. • **rtt**. Inspects the round-trip time on the external network interface. • **vlan**. Compares traffic with the specified vlan on the external network interface. • **vlan id**. Compares traffic with the specified vlan ID number on the external interface.
WebSocket	Specifies a condition based upon the properties of a websocket's connection. • **extension**. Matches the value of the `Sec-WebSocket-Extensions` header. • **key**. Matches the value of the `masking-key`. • **protocol**. Matches the value of the `Sec-WebSocket-Protocol` header. • **version**. Matches the value of the `Sec-WebSocket-Version` header.

About datagroup types for conditions

Conditions for a local traffic policy enable you to assign a datagroup type and value to an operand, as applicable. For example, you could configure a condition for an HTTP Referer host that ends with a datagroup value of `partner-domains`.

This table describes the datagroup types and supported comparison operators.

Table 3: Datagroup types for comparison operators

Datagroup type	Comparison operator
string	• equals • starts-with • ends-with • contains
IP address	• matches
number	• equals

About actions for a local traffic policy rule

The *actions* for a local traffic policy rule determine how traffic is handled. For example, actions for a rule could include the following ways of handling traffic.

- Blocking traffic
- Rewriting a URL
- Logging traffic
- Adding a specific header
- Redirecting traffic to a different pool member
- Selecting a specific Web Application policy

Table 4: Actions for local traffic policy matching

Action Type	Description
Enable	Enables the following actions. • **cache**. Enables caching for a connection. • **compression**. Enables compression for a connection. • **decompression**. Enables decompression for a connection. • **http**. Enables HTTP filter processing. • **request adapt**. Enables request adaptation, optionally sending traffic to a specified internal virtual server. • **response adapt**. Enables response adaptation, optionally sending traffic to a specified internal virtual server. • **server ssl**. Enables encrypted connections to backend servers. • **websocket**. Enables WebSocket processing.
Disable	Disables the following actions. • **cache**. Disables caching for a connection. • **compression**. Disables compression for a connection. • **decompression**. Disables decompression for a connection. • **http**. Disables HTTP filter processing. • **LTM policy**. Disables LTM® Policy processing on a request-by-request basis. • **persistence**. Disables persistence. • **request adapt**. Disables request adaptation, optionally sending traffic to a specified internal virtual server. • **response adapt**. Disables response adaptation, optionally sending traffic to a specified internal virtual server. • **server ssl**. Disables encrypted connections to backend servers. • **websocket**. Disables WebSocket processing.
Forward traffic	Controls where a connection is forwarded. • **node**. Forwards the connection to the specified node. • **pool**. Forwards the connection to the specified pool. • **virtual server**. Forwards the connection to the specified virtual server.
Insert	Inserts an HTTP header into the request or response. • **http cookie**. Inserts an HTTP Cookie header into the response.

Action Type	Description
	• **http header**. Inserts an HTTP header into the request or response. • **http referer**. Inserts an HTTP Referer header into the request. • **http set cookie**. Inserts an HTTP Set-Cookie header into the response.
Remove	Removes an HTTP header from the request or response. • **http cookie**. Removes an HTTP Cookie header from the response. • **http header**. Removes an HTTP header from the request or response. • **http referer**. Removes an HTTP Referer header from the request. • **http set cookie**. Removes an HTTP Set-Cookie header from the response.
Replace	• **http header**. Replaces the HTTP header in the request or response. • **http host**. Replaces the HTTP Host header in the request. • **http referer**. Replaces the HTTP Referer header in the request. • **http uri**. Replaces the HTTP URI, path, or string in the request.
Redirect	Redirects traffic to a different URL.
Reset traffic	Resets the connection.
Log	Writes messages to the local or remote system log.
Persist session	Controls how a connection is persisted. • **carp**. Persists the connection by using a Cache Array Routing Protocol algorithm. • **cookie hash**. Persists the connection by using a cookie hash method. • **cookie insert**. Persists the connection by using a cookie insert method. • **cookie passive**. Persists the connection by using a cookie passive method. • **cookie rewrite**. Persists the connection by using a cookie rewrite method. • **destination address**. Persists the connection based on the destination IP address. • **hash**. Persists the connection by using a cookie hash method. • **source address**. Persists the connection based on the source IP address. • **universal**. Persists the connection based on a user-defined key.
Set variable	Sets a Tcl variable in the runtime environment.

About Tcl command substitutions

Certain BIG-IP® local traffic policy actions support Tcl command substitutions, giving you significant flexibility in configuring policies. Tcl command substitutions provide quick, read-only access to immediately available runtime data, such as information about a current request's URI, or a header or cookie in the request or response.

Important: *Any Tcl command that requires a delay, for example, the* `after` *command, or that requires waiting for results from a request outside of the Traffic Management Microkernel (TMM), is not supported and might not succeed.*

Considerations when using Tcl command substitutions

When using Tcl command substitutions, the following guidelines apply.

- Memory and CPU capacity determine a maximum number of rules for active policies; however, excessive Tcl command substitutions can degrade performance.
- Tcl command substitutions are primarily intended for reading and returning data.

Tcl command substitution example

The following Tcl command is an example of a Tcl command substitution that can be used within a policy.

```
"tcl:[HTTP::uri]"
```

Note: *Each Tcl command must include a prefix of* `tcl:`*. If the* `tcl:` *prefix is omitted, the command is interpreted as a plain string.*

About options for conditions and actions

You can apply options to conditions and actions, as determined by the selected condition or action type. For example, you might want to constrain a condition based on the case-sensitivity of a string, which is easily applied by selecting the applicable option for that condition.

Table 5: Options for conditions

Condition Type	Options
Client SSL	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
CPU Usage	Not applicable.
Geo. IP	• **Apply to traffic on remote/local side of external/internal interface.** • **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Basic Auth.	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Cookie	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Header	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Host	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Method	• **Use case sensitive string comparison.**
HTTP Referer	• **Use normalized URI. The normalization applied includes a lower case scheme and URI, including host names, removing a trailing period (.) character, and percent encoding. For percent encoding, bytes not allowed in a URI are normalized to their percent encoded representation.** • **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Set Cookie	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**

Condition Type	Options
HTTP Status	Not applicable.
HTTP URI	• **Use normalized URI. The normalization applied includes a lower case scheme and URI, including host names, removing a trailing period (.) character, and percent encoding. For percent encoding, bytes not allowed in a URI are normalized to their percent encoded representation.** • **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP User Agent	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
HTTP Version	• **Use case sensitive string comparison.** ***Note:*** *Applies only to* ***protocol*** *and* ***full string*** *settings.*
IP Reputation	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
SSL Certificate	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
SSL Extension	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**
TCP	• **Apply to traffic on remote/local side of external/internal interface.** • **Use case sensitive string comparison.**
WebSocket	• **Skip this condition if it is missing from the request.** • **Use case sensitive string comparison.**

Table 6: Options for actions

Action Type	Options
Enable	Not applicable.
Disable	Not applicable.
Forward traffic	Override default forward action using: • **SNAT** and **disable**/**automap**. • **SNAT Pool** and pool name. • VLAN: **(none)** • VLAN: VLAN Name
Insert	Not applicable.
Remove	Not applicable.
Replace	Not applicable.
Redirect	Not applicable.

Action Type	Options
Reset traffic	Not applicable.
Log	Facility: • **local0** (default) • **local1** • **local2** • **local3** • **local4** • **local7** • **syslog** • **security** • **autopriv** • **daemon** • **kern** • **cron** • **ftp** • **lpr** • **mail** • **news** • **uucp** Priority: • **debug** • **info** (default) • **notice** • **warning** • **error** **Send log message to remote server:** • **Host** • **Port**
Persist session	Not applicable.
Set variable	Not applicable.

Common tmsh commands for local traffic policies

You can use `tmsh` commands with policies, as necessary. Common commands include those in the table.

Table 7: Common tmsh commands for local traffic policies

Description	tmsh Command
Create a draft policy.	`(tmos)# create ltm policy /Common/Drafts/policy_name strategy first-match`
Publish a draft policy.	`(tmos)# publish ltm policy /Common/Drafts/policy_name`

Description	tmsh Command
List all draft policies.	`(tmos)# show ltm policy /Common/Drafts/*`
List all published policies.	`(tmos)# show ltm policy`
List configuration details for a draft policy.	`(tmos)# list ltm policy /Common/Drafts/policy_name`
List configuration details for a published policy.	`(tmos)# list ltm policy policy_name`

Creating a draft local traffic policy

You can use BIG-IP® local traffic policy matching to direct traffic in accordance with rules, which are applied as determined by the specified strategy, conditions, and actions.

Note: *Local traffic policies that have been upgraded from BIG-IP software version 12.0, or earlier, appear in the Published Policies list.*

1. On the Main tab, click **Local Traffic** > **Policies** > **Policy List**.
   The Policy List Page screen opens.
2. Click **Create**.
   The New Policy screen opens.
3. In the **Policy Name** field, type a unique name for the policy.
4. In the **Description** field, type a description for the policy.
5. From the **Strategy** list, select a matching strategy.
6. Click **Create Policy**.
   The policy is created and the Rules area appears.
7. In the Rules area, click **Create**.
8. In the Match all of the following conditions area, click +.
9. From the **Client SSL** list, select a condition type, and configure the applicable settings and available options.
10. In the Match all of the following conditions area, click + to add an additional condition, as necessary, and configure the applicable settings and available options.
11. In the Do the following when the traffic is matched area, click +.
12. From the **Enable** list, select an action type, and configure the applicable settings and available options.
13. In the Do the following when the traffic is matched area, click + to add an additional action, as necessary, and configure the applicable settings and available options.
14. Click **Save**.

The policy appears in the Draft Policies list.

Publishing a local traffic policy

Before you can publish a local traffic policy, a draft policy must be available.

After you create a draft local traffic policy, you need to publish the policy, and then associate the published policy with a virtual server.

Note: *Local traffic policies that have been upgraded from BIG-IP software version 12.0, or earlier, appear in the Published Policies list.*

1. On the Main tab, click **Local Traffic** > **Policies** > **Policy List**.
   The Policy List Page screen opens.
2. Select the check box of the draft policy to publish.
3. Click **Publish**.

   The draft policy is removed from the Draft Policies list, and the modified published policy appears in the Published Policies list.

The draft local traffic policy is published and available to assign to a virtual server.

Modifying a published local traffic policy

You must have a published local traffic policy available, before you can modify its settings.

You can modify a published local traffic policy, by creating a draft policy from the published policy, making any necessary changes, and then publishing the modified draft policy. You cannot modify a published policy directly; you can only modify a draft policy. If the published local traffic policy is associated with a virtual server, the modified policy settings are updated in the associated virtual server.

1. On the Main tab, click **Local Traffic** > **Policies** > **Policy List**.
   The Policy List Page screen opens.
2. Click the name of a published policy.
3. Click **Create Draft**.

   A draft policy of the same name appears in the Draft Policies list.

   Note: *When you publish a policy, the draft policy is removed from the Draft Policies list.*
4. Click the name of the draft policy.
5. Modify the applicable settings.
6. Click **Save**.
7. Click **Save Draft**.
   The Policy List Page screen opens.
8. Select the check box of the draft policy to publish.
9. Click **Publish**.

   The draft policy is removed from the Draft Policies list, and the modified published policy appears in the Published Policies list.

The published local traffic policy is updated.

Reordering local traffic draft policy rules

Before you can reorder local traffic policy rules, there must be a draft policy with multiple rules available.

Note: *You cannot reorder rules in a published policy.*

You can reorder rules within a draft policy, as needed.

1. On the Main tab, click **Local Traffic** > **Policies** > **Policy List**.
   The Policy List Page screen opens.
2. Click the name of a draft policy.
3. Click and drag the rule or rules that you want to reorder into the preferred sequence.
4. Click **Save Draft**.
   The Policy List Page screen opens.

Rules in the draft local traffic policy appear in the sequence order that you configured.

Associating a published local traffic policy with a virtual server

After you publish a local traffic policy, you associate that published policy with the virtual server created to handle application traffic.

1. On the Main tab, click **Local Traffic** > **Virtual Servers**.
   The Virtual Server List screen opens.
2. Click the name of the virtual server you want to modify.
3. On the menu bar, click **Resources**.
4. In the Policies area, click the **Manage** button.
5. For the **Policies** setting, select the local traffic policy you created from the **Available** list and move it to the **Enabled** list.
6. Click **Finished**.

The published policy is associated with the virtual server.

Cloning a local traffic policy

You can clone (copy) either a draft or published BIG-IP® local traffic policy to create a different draft policy with the same settings of the original policy. After you clone the local traffic policy, you can modify it as necessary, publish it, and associate it with a virtual server.

1. On the Main tab, click **Local Traffic** > **Policies** > **Policy List**.
   The Policy List Page screen opens.
2. Click the name of a policy.
3. Click **Clone**.
   The **Policy Name** field becomes cleared.
4. In the **Policy Name** field, type a unique name for the policy.
5. Click **Create Policy**.
   The Draft Policy screen opens.
6. In the **Description** field, type a description for the policy.
7. From the **Strategy** list, select a matching strategy.
8. In the Rules area, click **Create**.
9. In the Match all of the following conditions area, click +.
10. From the **Client SSL** list, select a condition type, and configure the applicable settings and available options.
11. In the Match all of the following conditions area, click + to add an additional condition, as necessary, and configure the applicable settings and available options.
12. In the Do the following when the traffic is matched area, click +.

13. From the **Enable** list, select an action type, and configure the applicable settings and available options.
14. In the Do the following when the traffic is matched area, click **+** to add an additional action, as necessary, and configure the applicable settings and available options.

The policy appears in the Draft Policies list.

Creating a user-defined local traffic policy matching strategy

You can create a new local traffic policy matching strategy, based on a best-match policy matching strategy type. A user-defined best-match strategy can customize the precedence (order of preference) of added operands and selectors, compared to the predefined best-match policy.

1. On the Main tab, click **Local Traffic** > **Policies** > **Strategy List**.
   The Strategy List screen opens.
2. Click **Create**.
   The New Strategy screen opens.
3. In the **Name** field, type a unique name for the strategy.
4. From the **Operands** list, select an operand, configure the applicable settings, and click **Add**.
5. Click **Finished**.

The new user-defined best-match policy matching strategy appears in the Strategy List screen.

Deleting a local traffic policy

You can delete BIG-IP® local traffic policies when they become obsolete or are no longer used.

1. On the Main tab, click **Local Traffic** > **Policies** > **Policy List**.
   The Policy List Page screen opens.
2. Select the check box for each policy that you want to delete.
3. Click **Delete**.
   The Confirm delete? popup screen opens.
4. Click **OK**.

The system deletes the policies that you selected.

Example: Preventing a Nimda worm attack

You can create a local traffic policy that prevents the Nimda worm attack. If the URL contains certain strings that are known to be associated with the Nimda worm, then the local traffic policy can use a forwarding action that resets the connection.

Examples

Creating a policy to prevent a Nimda worm attack: video example

You can associate a BIG-IP® local traffic policy to prevent a Nimda worm attack. The policy forwards a URL containing strings associated with the Nimda worm, and resets the connection. Watch the following video for an example of creating a local traffic policy and associating it with a virtual server.

Watch how to create a policy to prevent a Nimda worm attack

You can also visit our DevCentral™ YouTube channel to see this video. Use any of these ways:

- Click this URL: `https://www.youtube.com/watch?v=0qe-iyt4Bwg`.
- Copy and paste the above URL into your browser window.
- Use your browser to search for this video using the title *F5: Creating a local traffic policy to prevent a Nimda worm attack*.

Preventing a Nimda worm attack: tmsh example

This topic provides a `tmsh` command to list the configured settings for a Nimda policy. The policy directs that if certain strings are known to be associated with the Nimda worm, the local traffic policy uses a forwarding action that resets the connection. This topic also provides a `tmsh` command to list the configured virtual server settings.

```
(tmos)# list ltm policy Stop_Nimda
ltm policy StopNimda{
    controls { forwarding }
    description "This policy blocks the Nimda worm."
    last-modified 2016-03-02:11:46:00
    requires { http }
    rules {
        ClobberNimda {
            actions {
                0 {
                    forward
                    reset
                }
            }
            conditions {
                0 {
                    http-uri
                    query-string
                    values { root.exe admin.dll cmd.exe }
                }
            }
```

```
        }
    }
    status published
    strategy first-match
}

(tmos.ltm.virtual)# list HTTP-VS1
ltm.virtual.HTTP-VS1{
    destination 10.10.0.21:http
    ip-protocol tcp
    mask 255.255.255.255
    policies {
        StopNimda { }
    }
    profiles {
        http { }
        tcp { }
    }
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vs-index 2
}
```

Preventing a Nimda worm attack: iRules example

This topic provides an example of iRules® code that is equivalent to a policy that protects against a Nimda worm attack. The iRule directs that if certain strings are known to be associated with the Nimda worm, the local traffic policy uses a forwarding action that resets the connection.

```
when HTTP_REQUEST {
set uri [string tolower [HTTP::uri]]
if { ($uri contains "cmd.exe") or ($uri contains "root.exe") or ($uri contains
"admin.dll") } {
discard
}
}
```

Example: Using selective compression

You can create a local traffic policy to support selective compression for types of content that can benefit from compression. For example, common text types (HTML, XML, and CSS style sheets) can realize performance improvements, especially across slow connections, if you compress them.

Examples

Creating a policy to support selective compression: video example

Selective compression: tmsh example

Selective compression: iRules example

Creating a policy to support selective compression: video example

You can associate a BIG-IP® local traffic policy with a virtual server to support selective compression for types of content that can benefit from compression. For example, common text types (HTML, XML, and CSS style sheets) can realize performance improvements, especially across slow connections, if you compress them. Watch the following video for an example of creating a local traffic policy and associating it with a virtual server.

Watch how to create a policy to support selective compression

You can also visit our DevCentral™ YouTube channel to see this video. Use any of these ways:

- Click this URL: `https://youtu.be/d85swKvXS1w`.
- Copy and paste the above URL into your browser window.
- Use your browser to search for this video using the title *F5: Using Selective Compression*.

Example: Using selective compression

Selective compression: tmsh example

This topic provides a `tmsh` command to list the configured settings for a Selective Compression policy, for types of content that can benefit from compression. For example, common text types (HTML, XML, and CSS style sheets) can realize performance improvements, especially across slow connections. This topic also provides a `tmsh` command to list the configured virtual server settings.

```
(tmos)# list ltm policy SelectiveCompression
ltm policy SelectiveCompression{
    controls { compression }
    description "This policy compresses specified file types."
    last-modified 2016-03-02:11:46:00
    requires { http }
    rules {
        CompressFiles {
            actions {
                0 {
                    compress
                    response
                    enable
                }
            }
            conditions {
                0 {
                    http-header
```

```
                    name Content-Type
                    starts-with
                    values { text/ }
                }
                1 {
                    cpu-usage
                    last-1min
                    less-or-equal
                    values {5}
                }
            }
        }
    }
    status published
    strategy first-match
}

(tmos.ltm.virtual)# list ltm virtual HTTP-VS2
ltm.virtual.HTTP-VS2{
    destination 10.10.0.31:http
    ip-protocol tcp
    mask 255.255.255.255
    policies {
        SelectCompression { }
    }
    profiles {
        http { }
        httpcompression { }
        tcp { }
    }
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vs-index 3
}
```

Example: Using selective compression

Selective compression: iRules example

This topic provides an example of iRules® code that is equivalent to a policy to support selective compression for types of content that can benefit from compression.

```
when HTTP_REQUEST {
   COMPRESS::disable
   if { [HTTP::header Content-Type] contains "text" } {
      log "Enabling compression for this request"
      COMPRESS::enable
   }
}
```

Example: Using selective compression

Example: Preventing a spoof of an x-forwarded-for request

You can create a local traffic policy to prevent a spoof of an x-forwarded-for request. This is a security issue where attackers might attempt to thwart security by falsifying the IP address in a header, and pass it through the BIG-IP® system.

Examples

Creating a policy to prevent a spoof of an x-forwarded-for request: video example

Preventing a spoof of an x-forwarded-for request: tmsh example

Preventing a spoof of an x-forwarded-for request: iRules example

Creating a policy to prevent a spoof of an x-forwarded-for request: video example

You can associate a BIG-IP® local traffic policy with a virtual server to prevent a spoof of an x-forwarded-for request. This is a security issue where attackers might attempt to thwart security by falsifying the IP address in a header, and pass it through the BIG-IP system. Watch the following video for an example of creating a local traffic policy and associating it with a virtual server.

Watch how to create a policy to prevent a spoof of an x-forwarded-for request

You can also visit our DevCentral™ YouTube channel to see this video. Use any of these ways:

- Click this URL: `https://youtu.be/QrQxjt4-e4k`.
- Copy and paste the above URL into your browser window.
- Use your browser to search for this video using the title *F5: Creating a local traffic policy to prevent a spoof of an x-forwarded-for request*.

When you have completed the task shown in the video, the policy is associated with a virtual server.

Example: Preventing a spoof of an x-forwarded-for request

Preventing a spoof of an x-forwarded-for request: tmsh example

This topic provides a `tmsh` command to list the configured settings for a policy to prevent a spoof of an x-forwarded-for request. This is a request where attackers might attempt to thwart security by falsifying the IP address in a header, and pass it through the BIG-IP® system. This topic also provides a `tmsh` command to list the configured virtual server settings.

```
(tmos)# list ltm policy PreventSpoofOfXFF
ltm policy SelectiveCompression{
    controls { compression }
    description "This policy prevents a spoof of an x-forwarded-for request."
    last-modified 2016-03-02:11:46:00
    requires { http }
    rules {
        StopSpoof {
            actions {
                0 {
                    http-header
                    replace
                    name X-foRWardED-for
```

```
                    value tcl:[IP::client_addr]
                }
            }
        }
    }
    status published
    strategy first-match
}

(tmos.ltm.virtual)# list ltm virtual HTTP-VS3
ltm.virtual.HTTP-VS3{
    destination 10.10.0.41:http
    ip-protocol tcp
    mask 255.255.255.255
    policies {
        PreventSpoofOfXFF { }
    }
    profiles {
        http { }
        tcp { }
    }
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vs-index 4
}
```

Example: Preventing a spoof of an x-forwarded-for request

Preventing a spoof of an x-forwarded-for request: iRules example

This topic provides an example of `iRules` code that is equivalent to a policy that prevents a spoof of an x-forwarded-for request. This is a situation where attackers might attempt to thwart security by falsifying the IP address in a header, and pass it through the BIG-IP® system. This example replaces a request that includes an x-forwarded-for header with the actual client IP address.

```
when HTTP_REQUEST {
    set xff 0
    foreach x [HTTP::header names] {
        if { [string tolower $x] equals "x-forwarded-for" } {
            set xff 1
            HTTP::header remove $x
            HTTP::header insert X-FORWARDED-FOR [IP::client_addr]
        }
    }
    if { $xff == 0 } {
        HTTP::header insert X-FORWARDED-FOR [IP::client_addr]
    }
}
```

Example: Preventing a spoof of an x-forwarded-for request

Example: Mitigating shellshock

You can create a local traffic policy to mitigate shellshock. In *shellshock*, an Internet service misuses `bash` shell functionality to process requests that execute arbitrary commands, potentially giving an attacker unauthorized access. This example policy examines requests for an uncommon pattern of "`() {`" in the URI, to minimize the possibility of false-positive matches.

Examples

Creating a policy to mitigate a shellshock attack: video example

A shellshock attack refers to a class of exploits that misuse the `bash` shell through a specifically crafted URL. You can associate a BIG-IP® local traffic policy with a virtual server to mitigate a shellshock attack, where the policy examines requests for a pattern of "`() {`" in the URI. Watch the following video for an example of creating a local traffic policy and associating it with a virtual server.

Watch how to create a policy to mitigate a shellshock attack

You can also visit our DevCentral™ YouTube channel to see this video. Use any of these ways:

- Click this URL: *https://youtu.be/qL98Xn1zB5U*.
- Copy and paste the above URL into your browser window.
- Use your browser to search for this video using the title *F5: Creating a local traffic policy to mitigate a shellshock attack*.

Example: Mitigating shellshock

Mitigating shellshock: tmsh example

This topic provides a `tmsh` command to list the configured settings for a Mitigating Shellshock policy. During this type of attack, a class of exploits misuse the `bash` shell through a specifically crafted URL. This topic also provides a `tmsh` command to list the configured virtual server settings.

```
(tmos)# list ltm policy MitigatingShellshock
ltm policy MitigatingShellshock{
    controls { forwarding }
    description "This policy mitigates shellshock."
    last-modified 2016-03-02:11:46:00
    requires { http }
    rules {
        StopShellshock {
            actions {
                0 {
                    log
                    write
                    facility local0
                    message "tcl:Shellshock detected from [IP::client_addr], blocked"
                    priority info
                }
                1 {
                    forward
```

```
                    reset
                }
            }
            conditions {
                0 {
                    http-uri
                    query string
                    contains
                    values { "() {" }
                }
            }
        }
    }
    status published
    strategy first-match
}

(tmos)# list ltm virtual HTTP-VS4
ltm.virtual.HTTP-VS4{
    destination 10.10.0.51:http
    ip-protocol tcp
    mask 255.255.255.255
    policies {
        MitigatingShellshock { }
    }
    profiles {
        http { }
        tcp { }
    }
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vs-index 5
}
```

Example: Mitigating shellshock

Mitigating shellshock: iRules example

This topic provides an example of iRules® code that is equivalent to a policy to mitigate shellshock, where the policy examines requests for a pattern of `"() {"` in the URI.

```
when HTTP_REQUEST {
  set pattern "*() \{*";
  if { [string match $pattern [HTTP::uri]] } {
    log local0. "Detected CVE-2014-6271 attack from '[IP::client_addr]' in URI '[HTTP::uri]'
";
    reject;
  } else {
    foreach header_name [HTTP::header names] {
      foreach header_value [HTTP::header values $header_name] {
        if { [string match $pattern $header_value] } {
          log local0. "Detected CVE-2014-6271 attack from '[IP::client_addr]' in HTTP Header
 $header_
          reject;
          break;
        }
      }
    }
  }
}

when HTTP_REQUEST {
  if { [string match "*() \{*" [HTTP::request]] } {
    log local0. "Detected CVE-2014-6271 attack from '[IP::client_addr]'; URI = '[HTTP::uri]'
";
```

```
    reject;
  }
}
```

Example: Mitigating shellshock

Legal Notices

Legal notices

Publication Date

This document was published on December 27, 2017.

Publication Number

MAN-0612-01

Copyright

Trademarks

For a current list of F5 trademarks and service marks, see *http://www.f5.com/about/guidelines-policies/trademarks*.

All other product and company names herein may be trademarks of their respective owners.

Patents

This product may be protected by one or more patents indicated at: *https://f5.com/about-us/policies/patents*.

Export Regulation Notice

This product may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this product from the United States.

Index

T

U

V

X