BIG-IP® Local Traffic Management: Profiles Reference

Version 12.1

Table of Contents

Introduction to Local Traffic Profiles

Introduction to profiles

Profiles are a configuration tool that you can use to affect the behavior of certain types of network traffic. More specifically, a *profile* is an object that contains settings with values, for controlling the behavior of a particular type of network traffic, such as HTTP requests and responses.

- You can use the default profiles, which means that you do not need to actively configure any profile settings. The BIG-IP system uses them to automatically direct the corresponding traffic types according to the values specified in the those profiles.
- You can create a custom profile, using the default profile as the parent profileA *custom profile* is a profile derived from a default profile and contains values that you specify.
- You can create a custom profile to use as a parent profile for other custom profiles.

After configuring a profile, you associate the profile with a virtual server. The virtual server then processes traffic according to the values specified in the profile. Using profiles enhances your control over managing network traffic, and makes traffic-management tasks easier and more efficient.

You can associate multiple profiles with a single virtual server. For example, you can associate a TCP profile, an SSL profile, and an HTTP profile with the same virtual server.

Profile types

The BIG-IP® system provides several types of profiles. While some profile types correspond to specific application services, such as HTTP, SSL, and FTP, other profiles pertain to traffic behaviors applicable to Layer 4 protocols such as TCP and UDP, and authentication protocols such as LDAP, and RADIUS. Also included are profiles specifically for different types of session persistence.

Default profiles

The BIG-IP® system includes one or more default profiles for each profile type. A *default profile* is a system-supplied profile that contains default values for its settings. An example of a default profile is the `http` profile. You can use a default profile in several ways:

- You can use a default profile as is. You simply configure your virtual server to reference the default profile.
- You can modify the default profile settings (not recommended). When you modify a default profile, you lose the original default profile settings. Thus, any custom profiles you create in the future that are based on that default profile inherit the modified settings.
- You can create a custom profile, based on the default profile (recommended). This allows you to preserve the default profile, and instead configure personalized settings in the custom profile. Custom profiles inherit some of the setting values of a parent profile that you specify. After creating a custom profile, you can configure your virtual server to reference the custom profile instead of the default profile.

Note: *You can modify a default profile, but you cannot create or delete a default profile.*

The BIG-IP system provides a default profile that you can use as is for each type of traffic. A *default profile* includes default values for any of the properties and settings related to managing that type of traffic. To implement a default profile, you simply assign the profile to a virtual server. You are not required to configure the setting values.

Custom and parent profiles

A *custom profile* is a profile that is derived from a parent profile that you specify. A *parent profile* is a profile from which your custom profile inherits its settings and their default values.

When creating a custom profile, you have the option of changing one or more setting values that the profile inherited from the parent profile. In this way, you can pick and choose which setting values you would like to change and which ones you would like to retain. An advantage to creating a custom profile is that by doing so, you preserve the setting values of the parent profile.

Note: *If you do not specify a parent profile when you create a custom profile, the BIG-IP® system automatically assigns a related default profile as the parent profile. For example, if you create a custom HTTP type of profile, the default parent profile is the default profile* `http`*.*

If you do not want to use a default profile as is or change its settings, you can create a custom profile. Creating a custom profile and associating it with a virtual server allows you to implement your own specific set of traffic-management policies.

When you create a custom profile, the profile is a child profile and automatically inherits the setting values of a parent profile that you specify. However, you can change any of the values in the child profile to better suit your needs.

If you do not specify a parent profile, the BIG-IP system uses the default profile that matches the type of profile you are creating.

Important: *When you create a custom profile, the BIG-IP system places the profile into your current administrative partition.*

Important: *Within the BIG-IP Configuration utility, each profile creation screen contains a check box to the right of each profile setting. When you check a box for a setting and then specify a value for that setting, the profile then retains that value, even if you change the corresponding value in the parent profile later. Thus, checking the box for a setting ensures that the parent profile never overwrites that value through inheritance.*

Once you have created a custom profile, you can adjust the settings of your custom profile later if necessary. If you have already associated the profile with a virtual server, you do not need to perform that task again.

The default profile as the parent profile

A typical profile that you can specify as a parent profile when you create a custom profile is a default profile. For example, if you create a custom TCP-type profile called `my_tcp_profile`, you can use the default profile `tcp` as the parent profile. In this case, Local Traffic Manager™ automatically creates the profile `my_tcp_profile` so that it contains the same settings and default values as the default profile `tcp`. The

new custom profile thus inherits its settings and values from its parent profile. You can then retain or change the inherited setting values in the custom profile to suit your needs.

The custom profile as the parent profile

When creating a custom profile, you can specify another custom profile, rather than the default profile, as the parent profile. The only restriction is that the custom profile that you specify as the parent must be of the same profile type as the profile you are deriving from the parent. Once you have created the new custom profile, its settings and default values are automatically inherited from the custom profile that you specified as the parent.

For example, if you create a profile called `my_tcp_profile2`, you can specify the custom profile `my_tcp_profile` as its parent. The result is that the default setting values of profile `my_tcp_profile2` are those of its parent profile `my_tcp_profile`.

If you subsequently modify the settings of the parent profile (`my_tcp_profile`), the BIG-IP® system automatically propagates those changes to the new custom profile.

For example, if you create the custom profile `my_tcp_profile` and use it as a parent profile to create the custom profile `my_tcp_profile2`, any changes you make later to the parent profile `my_tcp_profile` are automatically propagated to profile `my_tcp_profile2`. Conversely, if you modify any of the settings in the new custom profile (in our example, `my_tcp_profile2`), the new custom profile does not inherit values from the parent profile for those particular settings that you modified.

Profiles and virtual servers

Once you have created a profile for a specific type of traffic, you implement the profile by associating that profile with one or more virtual servers.

You associate a profile with a virtual server by configuring the virtual server to reference the profile. Whenever the virtual server receives that type of traffic, the BIG-IP® system applies the profile settings to that traffic, thereby controlling its behavior. Thus, profiles not only define capabilities per network traffic type, but also ensure that those capabilities are available for a virtual server.

Because certain kinds of traffic use multiple protocols and services, users often create multiple profiles and associate them with a single virtual server.

For example, a client application might use the TCP, SSL, and HTTP protocols and services to send a request. This type of traffic would therefore require three profiles, based on the three profile types TCP, Client SSL, and HTTP.

Each virtual server lists the names of the profiles currently associated with that virtual server. You can add or remove profiles from the profile list, using the BIG-IP Configuration utility. Note that the BIG-IP system has specific requirements regarding the combinations of profile types allowed for a given virtual server.

In directing traffic, if a virtual server requires a specific type of profile that does not appear in its profile list, the BIG-IP system uses the relevant default profile, automatically adding the profile to the profile list. For example, if a client application sends traffic over TCP, SSL, and HTTP, and you have assigned SSL and HTTP profiles only, LTM automatically adds the default profile tcp to its profile list.

At a minimum, a virtual server must reference a profile, and that profile must be associated with a UDP, FastL4, Fast HTTP, or TCP profile type. Thus, if you have not associated a profile with the virtual server, the BIG-IP system adds a `udp`, `fastl4`, `fasthttp`, or `tcp` default profile to the profile list.

The default profile that the BIG-IP system chooses depends on the configuration of the virtual server's protocol setting. For example, if the protocol setting is set to UDP, Local Traffic Manager adds the `udp` profile to its profile list.

Creating a local traffic profile

You perform this task to create a specific type of profile that you can then assign to a virtual server. You create a profile when the values configured in the default profile do not suit your needs.

1. On the Main tab, click **Local Traffic** > **Profiles**.
2. On the menu bar, expand or click a profile category and choose the type of profile you want to create.
3. Click **Create**.
4. In the **Name** field, type a unique name for the profile.
5. From the **Parent Profile** list, retain the default value or select another existing profile of the same type.
6. Select the **Custom** check box.
7. Configure all other profile settings as needed.
8. Click **Finished**.

After you perform this task, the new profile appears in the list of profiles on the system.

Services Profiles

Overview of Services profiles

The BIG-IP® system offers several features that you can use to intelligently control your application-layer traffic. These features are available through various configuration profiles.

A *profile* is a group of settings, with values, that correspond to a specific type of traffic, such as FTP traffic. A profile defines the way that you want the BIG-IP system to manage that traffic type. After you configure the type of profile you need, you assign it to a virtual server.

In addition to Services profiles, the BIG-IP system includes other features to help you manage your application traffic, such as health monitors for checking the health of an FTP service, and iRules® for querying or manipulating header or content data. Additional profiles may be available with other modules.

About HTTP profiles

The BIG-IP® system offers several features that you can use to intelligently control your application layer traffic. Examples of these features are the insertion of headers into HTTP requests and the compression of HTTP server responses.

These features are available through various configuration profiles. A *profile* is a group of settings, with values, that correspond to HTTP traffic. A profile defines the way that you want the BIG-IP system to manage HTTP traffic.

You can configure an HTTP profile to ensure that HTTP traffic management suits your specific needs. You can configure the profile settings either when you create a profile or after you create the profile by modifying the profile's settings. For all profile settings, you can specify values where none exist, or modify any default values to suit your needs. The BIG-IP system also includes default profiles that you can use as is, if you do not want to create a custom profile.

To manage HTTP traffic, you can use any of these profile types:

- HTTP (Hypertext Transfer Protocol)
- HTTP Compression
- Web Acceleration

In addition to the HTTP profiles, the BIG-IP system includes other features to help you manage your application traffic, such as health monitors for checking the health of HTTP and HTTPS services, and iRules® for querying or manipulating header or content data.

General HTTP properties

There are a few general settings that you can configure to create a basic HTTP type of profile that uses most of the default settings.

Proxy mode

The HTTP profile provides three proxy modes: Reverse, Explicit, and Transparent. You can configure a custom HTTP profile that uses a specific proxy mode, and assign the custom HTTP profile to a virtual server to manage proxying of HTTP traffic, as necessary.

Proxy Mode	Description
Reverse	Default. You can specify the Reverse Proxy Mode to enable the BIG-IP® system to manage responses from multiple servers.
Explicit	The Explicit Proxy Mode enables the BIG-IP system to handle HTTP proxy requests and function as a gateway. By configuring browser traffic to use the proxy, you can control whether to allow or deny a requested connection, based on configured policies. The Explicit Proxy Mode requires a DNS resolver, specified in the Explicit Proxy area of the screen.
Transparent	The Transparent Proxy Mode enables the BIG-IP system to forward invalid HTTP traffic to a specified server, instead of dropping the connection. By configuring an HTTP profile to forward invalid HTTP traffic, you can manage various atypical service provider scenarios, such as HTTP traffic from non-browser clients that function as web browsers.

Parent profile

Every profile that you create is derived from a parent profile. You can use the default `http` profile as the parent profile, or you can use another HTTP profile that you have already created.

HTTP settings

There are several general settings that you can configure to create an HTTP type of profile.

Basic Auth Realm

The Basic Auth Realm setting provides a quoted string for the basic authentication realm. The BIG-IP® system sends this string to a client whenever authorization fails.

Fallback host

Another feature that you can configure within an HTTP profile is HTTP redirection. HTTP redirection allows you to redirect HTTP traffic to another protocol identifier, host name, port number, or URI path.

Redirection to a fallback host occurs if all members of the targeted pool are unavailable, or if a selected pool member is unavailable. (The term *unavailable* refers to a member being disabled, marked as `down`, or having exceeded its connection limit.) When one or more pool members are unavailable, the BIG-IP® system can redirect the HTTP request to the fallback host, with the HTTP reply `Status Code 302 Found`.

Although HTTP redirection often occurs when the system generates an `LB_FAILED` iRule event, redirection can also occur without the occurrence of this event, such as when:

- The selected node sends an `RST` after a `TCP 3WHS` has completed, but before the node has sent at least a full response header.

- The BIG-IP system finds the selected node to be unreachable while receiving the body portion of a request or a pipelined request.

When configuring the BIG-IP system to redirect HTTP traffic to a fallback host, you can specify an IP address or a fully-qualified domain name (FQDN). The value that you specify becomes the value of the `Location` header that the server sends in the response. For example, you can specify a redirection as `http://redirector.siterequest.com`.

Fallback error codes

In addition to redirecting traffic when a target server becomes unavailable, you can also specify the HTTP error codes from server responses that should trigger a redirection to the fallback host. Typical error codes to specify are `500`, `501`, and `502`.

Headers in HTTP requests

You can insert headers into HTTP requests. The HTTP header being inserted can include a client IP address. Including a client IP address in an HTTP header is useful when a connection goes through a secure network address translation (SNAT) and you need to preserve the original client IP address.

The format of the header insertion that you specify is generally a quoted string. Alternatively, however, you can insert a Tools Command Language (Tcl) expression into a header that dynamically resolves to the preferred value. When you assign the configured HTTP profile to a virtual server, the BIG-IP® system then inserts the header specified in the profile into any HTTP request that the BIG-IP system sends to a pool or pool member.

Note: *In addition to inserting a string such as a client IP address into an HTTP request, you can configure the BIG-IP system to insert SSL-related headers into HTTP requests. Examples are: client certificates, cipher specifications, and client session IDs. To insert these types of headers, you must create an iRule.*

Content erasure from HTTP headers

You can configure a profile to erase the contents of a header from an HTTP request that is being sent from a client to a server. With this feature, you can erase header content from HTTP requests before forwarding the requests over the network. Such headers might contain sensitive information, such as user IDs or telephone numbers, that must be erased before the information is forwarded.

When you use this setting, the BIG-IP® system erases the contents of the specified header and replaces that content with blank spaces. The header itself is retained.

Note: *This feature does not apply to HTTP responses being sent from a server to a client.*

The client header with the contents to be erased must be specified as a quoted string.

Headers in an HTTP response

You can specify any headers within an HTTP response that you want the BIG-IP® system to allow. If you are specifying more than one header, separate the headers with a blank space. For example, if you type the string `Content-Type Set-Cookie Location`, the BIG-IP system then allows the headers `Content-Type`, `Set-Cookie`, and `Location`.

Response chunking

Sometimes, you might want to inspect and/or modify HTTP application data, such as compressing the content of an HTTP response. Such inspections or modifications require that the response be *unchunked*, that is, not in chunked encoding. Using the Response Chunking feature, the BIG-IP® system can unchunk a chunked response before performing an action on that response.

Possible chunking behaviors of the BIG-IP system

The BIG-IP® system takes specific action on a response depending on whether the response is chunked or unchunked.

Action	Original response is chunked	Original response is unchunked
Unchunk	the BIG-IP® system unchunks the response and processes the HTTP content, and passes the response on as unchunked. The connection closes when all data is sent to the client as indicated by the Connection: Close header.	The BIG-IP system processes the HTTP content and passes the response on untouched.
Rechunk	The BIG-IP system unchunks the response, processes the HTTP content, re-adds the chunk trailer headers, and then passes the response on as chunked. Any chunk extensions are lost.	The BIG-IP system adds transfer encoding and chunking headers on egress.
Selective	Same as Rechunk.	The BIG-IP system processes the HTTP content and then passes the response on untouched.
Preserve	The BIG-IP system leaves the response chunked, processes the HTTP content, and passes the response on untouched. Note that if HTTP compression is enabled, Local Traffic Manager does not compress the response.	The BIG-IP system processes the HTTP content and then passes the response on untouched.

OneConnect transformations

You can enable or disable part of the OneConnect™ feature, for HTTP/1.0 connections only. When this setting is enabled and a OneConnect profile is assigned to the virtual server, the setting performs `Connection` header transformations, for the purpose of keeping a client connection open. More specifically:

1. A client sends an HTTP/1.0 request.
2. The server sends a response, which initially includes a `Connection: Close` header.
3. the BIG-IP® system transforms the `Connection: Close` header to `Connection: Keep-Alive`.
4. Through use of the OneConnect profile, the server-side connection detaches, goes into the pool of available server-side connections used for servicing other requests, and eventually closes. This process is hidden from the client.
5. The client-side connection remains open, operating under the assumption that the server-side connection is still open and therefore able to accept additional requests from that client.

Note: *For this feature to take effect, you must also configure a OneConnect™ profile, which enables connection pooling.*

Rewrites of HTTP redirections

Sometimes, a client request is redirected from the HTTPS protocol to the HTTP protocol, which is a non-secure channel. If you want to ensure that the request remains on a secure channel, you can cause the redirection to be rewritten so that it is redirected back to the HTTPS protocol.

To enable the BIG-IP® system to rewrite HTTP redirections, you use the Rewrite Redirections setting to specify the way that you want the system to handle URIs during the rewrite.

Note that the rewriting of any redirection takes place only in the HTTP `Location` header of the redirection response, and not in any content of the redirection.

Possible values

When configuring the BIG-IP system to rewrite HTTP redirections, you specify one of these values:

None
The system does not rewrite any redirections. This is the default value.

All
The system rewrites the URI in all HTTP redirect responses. In this case, the system rewrites those URIs as if they matched the originally-requested URIs.

Matching
The system rewrites the URI in any HTTP redirect responses that match the request URI (minus an optional trailing slash).

Nodes
The system rewrites the hidden node IP address to a virtual server address, and rewrites the port number. You choose this value when the virtual server is not configured with a Client SSL profile (that is, when the virtual server is configured to process plain HTTP traffic only).

Note: *For values All, Matching, and Nodes, the system always hides the node IP address. Also, the system hides the node IP address independently of the protocol rewrite, with no regard to the protocol in the original redirection.*

Examples of rewriting HTTP redirections with the system listening on port 443

This table shows examples of how redirections of client requests are transformed when the BIG-IP system is listening on port 443, and the Rewrite Redirections setting is enabled.

Original Redirection	Rewrite of Redirection
http://www.myweb.com/myapp/	https://www.myweb.com/myapp/
http://www.myweb.com:8080/myapp/	https://www.myweb.com/myapp/

Examples of rewriting HTTP redirections with the system listening on port 4443

This table shows examples of how redirections of client requests are transformed when the BIG-IP system is listening on port 443, and the Rewrite Redirections setting is enabled.

Original Redirection	Rewrite of Redirection
http://www.myweb.com/myapp/	https://www.myweb.com:4443/myapp/
http://www.myweb.com:8080/myapp/	https://www.myweb.com:4443/myapp/

Cookie encryption and decryption

You can use the BIG-IP Configuration utility to encrypt one or more cookies that the BIG-IP® system sends to a client system. When the client sends the encrypted cookie back to the BIG-IP system, the system decrypts the cookie.

X-Forwarded-For header insertion

When using connection pooling, which allows clients to make use of existing server-side connections, you can insert the `XForwarded For` header into a request. When you configure the BIG-IP® system to insert this header, the target server can identify the request as coming from a client other than the client that initiated the connection. The default setting is `Disabled`.

Maximum columns for linear white space

You can specify the maximum number of columns allowed for a header that is inserted into an HTTP request.

Linear white space separators

You can specify the separator that the BIG-IP® system should use between HTTP headers when a header exceeds the maximum width specified by the LWS Maximum Columns feature.

Maximum number of requests

You can specify the maximum number of requests that the system allows for a single Keep-Alive connection. When the specified limit is reached, the final response contains a `Connection: close` header, which is followed by the closing of the connection. The default setting is `0`, which in this case means that the system allows an infinite number of requests per Keep-Alive connection.

Proxy Via headers

You can configure the BIG-IP® system to remove, preserve, or append `Via` headers in HTTP client requests, HTTP server responses, or both.

Overview: Using Via headers

Via headers provide useful information about intermediate routers that can be used in network analysis and troubleshooting.

About using Via headers in requests and responses

The `Via` header, configured in an HTTP profile, provides information about each intermediate router that forwards a message. Intermediate routers between a client and an origin web server use the `Via` header to indicate intermediate protocols and recipients. This information can be used for the following tasks:

- Identifying the intermediate routers that forward messages.
- Identifying the protocols for intermediate routers.

About identifying intermediate routers with a Via header

The `Via` header, configured in an HTTP profile, concatenates information for each router in a response or request, separated by commas. For example, the following `Via` header includes two routers, with each router comprising the required protocol and address:

```
Via: 1.1 wa.www.siterequest1.com, 1.1 wa.www.siterequest2.com
```

When a client initiates a request with a `Via` header to an origin web server, the origin web server returns a response with a `Via` header often following a similar path. For example, a `Via` header router sequence for the request would be 1, 2, 3, and the router sequence for the client's response would be 3, 2, 1.

The inverse is true when an origin web server initiates a response with a `Via` header to a client. For example, a `Via` header router sequence for a response would be 1, 2, 3, and the router sequence for the client's request would be 3, 2, 1.

About identifying protocols for intermediate routers with a Via header

You can identify specific protocols and versions of protocols for intermediate routers by using a `Via` header, configured in an HTTP profile. When a client sends a request to an origin web server, the header information is concatenated for each intermediate router, including the protocol type (if different from HTTP) and version.

The `Via` header includes both required and optional protocol information about each router, as follows:

- The HTTP protocol name is optional; however, other protocol names are required.
- The protocol version of the message is required, which for HTTP is 1.0, 1.1, and so on.
- The host name is required. For privacy purposes, however, an alias can replace the actual host name.
- The port number associated with the host name is optional. When the port number is omitted, the default port applies.
- A comment describing the router is optional, and includes whatever string you specify in the **Send Proxy Via Header Host Name** field, by selecting **Append** in the list for **Send Proxy Via Header In Request** or **Send Proxy Via Header In Response**.

Note: *If you prefer to replace the host name with another string, instead of appending a string to the* `Via` *header, you must use an iRule or the command line.*

Because the `Via` header includes the protocol name and version, applications are able to acquire this information for the various intermediate routers and use it, as necessary.

Via Header settings

This table describes controls and strings for **Via Header** settings in an HTTP profile.

Control	Default	Description
Send Proxy Via Header In Request	**Remove**	Specifies whether to **Remove**, **Preserve**, or **Append** `Via` headers included in a client request to an origin web server. • **Remove**. The BIG-IP® system deletes the `Via` header from the client request. • **Preserve**. The BIG-IP system includes the `Via` header in the client request to the origin web server. • **Append**. The BIG-IP system appends the string specified in the **Send Proxy Via Header In Host Name** field to the **Via** header in the client request to the origin web server.
Send Proxy Via Header In Response	**Remove**	Specifies whether to **Remove**, **Preserve**, or **Append** `Via` headers included in an origin web server response to a client. • **Remove**. The BIG-IP system deletes the `Via` header from the origin web server response. • **Preserve**. The BIG-IP system includes the `Via` header in the origin web server response to the client.

Control	Default	Description
		• **Append**. The BIG-IP system appends the string specified in the **Send Proxy Via Header In Host Name** field to the **Via** header in the origin web server response to the client.
Send Proxy Via Header Host Name	None	Specifies a string to append as a comment when sending a `Via` header in a request to an origin web server or in a response to a client. ***Note:*** *If you prefer to replace the host name with another string, instead of appending a string to the* `Via` *header, you must use an iRule or the command line.*

X-Forwarded-For header acceptance

This setting enables or disables trusting the client IP address, and statistics from the client IP address, based on the request's X-Forwarded-For (XFF) headers, if they exist.

Alternate X-Forwarded-For headers

Specifies alternative XFF headers instead of the default X-Forwarded-For header. If you are specifying more than one alternative XFF header, separate the alternative XFF headers with a blank space, such as `client1 proxyserver 129.78.138.66`.

Server agent name

When you create an HTTP profile, you can specify the string used as the server name in traffic generated by the BIG-IP® system. The default value is `BigIP`.

Enforcement settings

There are some settings related to enforcement that you can configure to create an HTTP type of profile.

Allow truncated redirects

The Allow Truncated Redirect setting determines the way in which the BIG-IP® system passes through traffic, when a redirect that lacks the trailing carriage-return and line-feed pair at the end of the headers is parsed. The default is Disabled, which silently drops the invalid HTTP request.

Maximum header size

This setting specifies the maximum size in bytes that the BIG-IP® system allows for all HTTP request headers combined, including the request line. If the combined headers length in bytes in a client request exceeds this value, the system stops parsing the headers and resets the TCP connection. The default value is `32,768` bytes.

Oversize client headers

The **Oversize Client Headers** setting determines the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Size** value is exceeded by the client. The default is disabled, which rejects the connection.

Note: *This feature is only available on the HTTP profile when you set the proxy mode feature to* ***Transparent****.*

Oversize server headers

The **Oversize Server Headers** setting determines the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Size** value is exceeded by the server. The default is disabled, which rejects the connection.

Note: *This feature is only available on the HTTP profile when you set the proxy mode feature to* ***Transparent****.*

Maximum header count

The Maximum Header Count setting determines the maximum number of headers in an HTTP request or response that the BIG-IP® system accepts. If a client or server sends a request or response with the number of headers exceeding the specified value, then the connection is dropped. The default value is 64.

Excess client headers

The **Excess Client Headers** setting specifies the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Count** value is exceeded by the client. The default is disabled, which rejects the connection.

Note: *This feature is only available on the HTTP profile when you set the proxy mode feature to* ***Transparent****.*

Excess server headers

The **Excess Server Headers** setting specifies the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Count** value is exceeded by the server. The default is disabled, which rejects the connection.

Note: *This feature is only available on the HTTP profile when you set the proxy mode feature to* ***Transparent****.*

Support for pipelining

Normally, a client cannot initiate a request until the previous request has received a response. HTTP/1.1 pipelining allows clients to initiate multiple requests even when prior requests have not received a response. Note, however, that each initiated request is still processed sequentially; that is, a request in the queue is not processed until the previous request has received a response.

By enabling support for pipelining on the BIG-IP® system, you remove the need to enable pipelining on the destination server itself. By default, this feature is enabled.

Unknown methods

The **Unknown Method** setting determines the way in which the BIG-IP® system manages HTTP traffic when an unknown HTTP method is parsed. You can configure the **Unknown Method** setting to allow, reject, or pass through the HTTP traffic. The default is to allow unknown methods.

Known methods

In the **Known Methods** setting, the **Enabled Methods** list determines the way in which the BIG-IP® system manages HTTP traffic when known HTTP methods are parsed. You configure the **Known Methods Enabled Methods** list to allow the BIG-IP system to manage specified known methods with optimum performance.

The default **Enabled Methods** list includes the following HTTP/1.1 methods.

- **CONNECT**
- **DELETE**
- **GET**
- **HEAD**
- **LOCK**
- **OPTIONS**
- **POST**
- **PROPFIND**
- **PUT**
- **TRACE**
- **UNLOCK**

If you delete a known method from the **Enabled Methods** list, then the BIG-IP system applies the **Unknown Method** setting to manage that traffic.

Important: *Removing a standard method, such as **HEAD** or **CONNECT**, causes BIG-IP functionality that depends on detecting that method to fail to work correctly.*

You can add a user-defined method to the **Enabled Methods** list by typing the method in the **Add user defined method** field, and then clicking **Add**.

Explicit proxy settings

When you set the proxy mode to **Explicit**, you must also configure the settings in the Explicit Proxy area of the HTTP profile.

Important: *Explicit proxy mode is not compatible with connection mirroring. Ensure connection mirroring is not enabled for virtual servers using HTTP profiles in Explicit proxy mode. This also applies to secondary virtual servers that listen on the tunnels used by HTTP explicit proxy profiles.*

DNS Resolver

The **DNS Resolver** setting specifies the DNS resolver to use for DNS inquiries handled by the virtual servers associated with the HTTP explicit forward proxy profile you are creating.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****, in which case the setting is required. If no DNS resolver exists on the system, you can create one at* ***Network*** *>* ***DNS Resolvers*** *>* ***DNS Resolvers List*** *>* ***Create****.*

Route Domain

You can configure an HTTP profile to specify the route domain that is used for outbound connect requests for the explicit forward proxy feature. The default route domain is 0.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

Tunnel Name

The **Tunnel Name** setting specifies the tunnel that is used for outbound connect requests when the explicit forward proxy feature is used. Specifying a tunnel enables other virtual servers to receive connections initiated by the proxy service.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

Host Names

The **Host Name** setting specifies the name of hosts that should not be proxied when an explicit forward proxy is used.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

Default Connect Handling

The **Default Connect Handling** setting specifies the behavior of the forward explicit proxy service when handling outbound requests. By default, this setting is disabled.

- Enabled (checked) indicates that outbound requests are delivered directly, regardless of the presence of listening virtual servers.
- Disabled (check box cleared) indicates that outbound requests are delivered only if another virtual server is listening on the tunnel for the requested outbound connection. With this setting, virtual servers are required, and the system processes the outbound traffic before it leaves the device.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

Connection Failed Message

You can configure an http explicit forward proxy profile to specify the message that appears when a connection failure occurs. You can include TCL expressions.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

DNS Lookup Failed Message

You can configure an http explicit forward proxy profile to specify the message that appears when a DNS lookup failure occurs. You can include TCL expressions.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

Bad Request Message

You can configure an http explicit forward proxy profile to specify the message that appears when a bad request occurs. You can include TCL expressions.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

Bad Response Message

You can configure an http explicit forward proxy profile to specify the message that appears when a bad response occurs. You can include TCL expressions.

Note: *This setting is available on the HTTP profile only when you set the proxy mode feature to* ***Explicit****.*

sFlow settings

You can configure the HTTP profile to use sFlow technology to monitor traffic passing through the BIG-IP system.

Polling intervals

You can configure an HTTP profile to specify the maximum interval in seconds between two pollings. The default value is **Default**, which represents the value set on the System :: sFlow :: Global Settings :: http :: Properties screen. The initial default value is 10 seconds.

Sampling rates

You can configure an HTTP profile to specify the ratio of packets observed to the samples generated. For example, a sampling rate of 2000 specifies that the system randomly generates 1 sample for every 2000 packets observed. The default value is **Default**, which represents the value set on the System :: sFlow :: Global Settings :: http :: Properties screen. The initial default value is 1024 packets.

HSTS settings

An HTTP profile provides HTTP Strict Transport Security (HSTS) settings that insert a `Strict-Transport-Security` header into HTTP responses. When enabled, HSTS functionality requests that clients only use HTTPS connections (TLS or SSL) to the current host, and optionally to any subdomains of the current host's domain name, for a specified period of time.

Mode

The Mode setting enables and disables HSTS functionality within the HTTP profile. The default is cleared (disabled).

Maximum Age

The Maximum Age value specifies the length of time, in seconds, that HSTS functionality requests that clients only use HTTPS to connect to the current host and any subdomains of the current host's domain name. The default is `16070400` seconds (about six months). A value of `0` re-enables plaintext HTTP access.

Include Subdomains

The Include Subdomains setting applies the HSTS policy to the HSTS host and its subdomains. The default is selected (enabled).

About HTTP compression profiles

HTTP compression reduces the amount of data to be transmitted, thereby significantly reducing bandwidth usage. All of the tasks needed to configure HTTP compression on the BIG-IP® system, as well as the compression software itself, are centralized on the BIG-IP system. The tasks needed to configure HTTP compression for objects in an Application Acceleration Manager module policy node are available in the Application Acceleration Manager, but an HTTP compression profile must be enabled for them to function.

When configuring the BIG-IP system to compress data, you can:

- Configure the system to include or exclude certain types of data.
- Specify the levels of compression quality and speed that you want.

You can enable the HTTP compression option by setting the **URI Compression** or the **Content Compression** setting of the **HTTP Compression** profile to **URI List** or **Content List**, respectively. This causes the BIG-IP system to compress HTTP content for any responses in which the values that you specify in the **URI List** or **Content List** settings of an HTTP profile match the values of the `Request-URI` or `Content-Type` response headers.

Exclusion is useful because some URI or file types might already be compressed. Using CPU resources to compress already-compressed data is not recommended because the cost of compressing the data usually outweighs the benefits. Examples of regular expressions that you might want to specify for exclusion are `.*\.pdf`, `.*\.gif`, or `.*\.html`.

Note: *The string that you specify in the **URI List** or the **Content List** setting can be either a pattern string or a regular expression. List types are case-sensitive for pattern strings. For example, the system treats the pattern string* `www.f5.com` *differently from the pattern string* `www.F5.com`*. You can override this case-sensitivity by using the Linux* `regexp` *command.*

HTTP Compression profile options

You can use an HTTP Compression profile alone, or with the BIG-IP® Application Acceleration Manager, to reduce the amount of data to be transmitted, thereby significantly reducing bandwidth usage. The tasks

needed to configure HTTP compression for objects in an Application Acceleration Manager policy node are available in the Application Acceleration Manager, but an HTTP Compression profile must be enabled for them to function.

URI compression

If you enable compression, you probably do not want the BIG-IP® system to compress every kind of server response. Therefore, you can instruct the BIG-IP system to include in compression, or exclude from compression, certain responses that are specified in the URIs of client requests.

More specifically, you can type regular expressions to specify the types of server responses that you want the BIG-IP system to include in, or exclude from, compression. For example, you can specify that you want the system to compress all `.htm` responses by typing the regular expression `.*.htm`. The system then compares that response type to the URI specified within each client request, and if the system finds a match, takes some action.

Note: *The string that you specify can be either a pattern string or a regular expression. Note that list types are case-sensitive for pattern strings. For example, the system treats the pattern string* `www.f5.com` *differently from the pattern string* `www.F5.com`*. You can override this case-sensitivity by using the Linux* `regexp` *command.*

Content compression

If you enable compression, you probably do not want the BIG-IP® system to compress every kind of server response. Therefore, you can instruct the BIG-IP system to include in compression, or exclude from compression, certain responses that are specified in the `Content-Type` header of server responses.

More specifically, you can type regular expressions to specify the types of server responses that you want the BIG-IP system to include in, or exclude from, compression. For example, you can specify that you want the system to compress all `.htm` responses by typing the regular expression **.*.htm**. The BIG-IP system then compares that response type to the value of the `Content-Type` header specified within each server response, and if the system finds a match, takes some action.

Note: *The string that you specify can be either a pattern string or a regular expression. Note that list types are case-sensitive for pattern strings. You can override this case-sensitivity by using the Linux* `regexp` *command.*

Preferred compression methods

You can specify the compression method that you want the BIG-IP® system to use when compressing responses. The two possible compression methods are gzip and deflate.

Minimum content length for compression

When compression is enabled, you can specify the minimum length of a server response in uncompressed bytes that the BIG-IP® system requires for compressing that response. The BIG-IP system finds the content length of a server response in the `Content-Length` header of the server response. Thus, if the content length specified in the response header is below the value that you assign for minimum content length, LTM does not compress the response. The length in bytes applies to content length only, not headers.

For example, using the default value of `1024`, the BIG-IP system compresses only those responses with HTTP content containing at least 1024 bytes.

Sometimes the `Content-Length` header does not indicate the content length of the response. In such cases, the system compresses the response, regardless of size.

Compression buffer size

When compression is enabled, you can specify the maximum number of compressed bytes that the BIG-IP® system buffers before deciding whether or not to preserve a `Keep-Alive` connection and rewrite the `Content-Length` header.

For example, using the default value of `4096`, the BIG-IP system buffers up to 4096 bytes of compressed data before deciding whether or not to preserve the connection and rewrite the `Content-Length` header.

The BIG-IP system decision to rewrite the `Content-Length` header depends on whether response chunking is enabled (using the **Response Chunking** profile setting).

About the Vary header

When compression is enabled, the **Vary Header** setting inserts the `Vary: Accept-Encoding` header into a compressed server response. If the `Vary` header already exists in the response, Local Traffic Manager™ appends the value `Accept-Encoding` to that header.

The reason for inserting the `Vary: Accept-Encoding` header into a server response is to follow a recommendation by RFC2616, which states that the `Vary` header should be inserted into any cacheable response that is subject to server-driven negotiation. Server responses that are subject to HTTP compression fall into this category.

If the **Vary Header** setting is disabled, the BIG-IP system does not insert the `Vary` header into a server response.

To disable the `Vary` header, locate the **Vary Header** setting and clear the **Enabled** box.

Compression for HTTP/1.0 requests

The **HTTP/1.0 Requests** setting is included for backward compatibility, allowing HTTP compression for responses to HTTP/1.0 client requests. The default value for this setting is Disabled.

If this setting is set to Enabled, the BIG-IP® system only compresses responses in either of the following cases:

- When the server responds with a `Connection: close` header
- When the response content is no greater than the value of the **Compression Buffer Size** setting

To enable compression for HTTP/1.0 requests, locate the **HTTP/1.0 Requests** setting and select the check box.

About the Accept-Encoding header

Normally, when you enable HTTP compression, the BIG-IP® system strips out the `Accept-Encoding` header from the HTTP request. This causes the BIG-IP system, instead of the target server, to perform the HTTP compression.

By default, the **Keep Accept Encoding** setting is disabled. If you want to allow the target server instead of the BIG-IP system to perform the HTTP compression, simply enable this setting.

Browser workarounds

When you enable the **Browser Workarounds** setting, the system uses built-in workarounds for several common browser issues that occur when compressing content. The default setting is disabled (cleared). More specifically, enabling this setting prevents the system from compressing server responses when any of these conditions exists:

- The client browser is Netscape® version 4.0x.
- The client browser is Netscape version 4.x (that is, versions 4.10 and higher), and the `Content-Type` header of the server response is not set to **text/html** or **text/plain**.
- The client browser is Microsoft® Internet Explorer® (any version), the `Content-Type` header of the server response is set to either **text/css** or **application/x-javascript**, and the client connection uses SSL.
- The client browser is Microsoft® Internet Explorer® (any version), the `Content-Type` header of the server response is set to either **text/css** or **application/x-javascript**, and the `Cache-Control` header of the server response is set to **no-cache**.

About Web Acceleration profiles

When used by the BIG-IP® system without other provisioned modules, the Web Acceleration profile uses basic default acceleration.

Web Acceleration profile settings

This table describes the Web Acceleration profile configuration settings and default values.

Setting	Value	Description
Name	No default	Specifies the name of the profile.
Parent Profile	Selected predefined or user-defined profile	Specifies the selected predefined or user-defined profile.
Partition / Path	`Common`	Specifies the partition and path to the folder for the profile objects.
Cache Size	`100`	This setting specifies the maximum size in megabytes (MB) reserved for the cache. When the cache reaches the maximum size, the system starts removing the oldest entries.
Maximum Entries	`10000`	Specifies the maximum number of entries that can be in the cache.
Maximum Age	`3600`	Specifies how long in seconds that the system considers the cached content to be valid.
Minimum Object Size	500	Specifies the smallest object in bytes that the system considers eligible for caching.

Setting	Value	Description
Maximum Object Size	50000	Specifies the largest object in bytes that the system considers eligible for caching.
URI Caching	Not Configured	Specifies whether the system retains or excludes certain Uniform Resource Identifiers (URIs) in the cache. The process forces the system either to cache URIs that typically are ineligible for caching, or to not cache URIs that typically are eligible for caching.
URI List	No default value	Specifies the URIs that the system either includes in or excludes from caching. • Pin List. Lists the URIs for responses that you want the system to store indefinitely in the cache. • Include List. Lists the URIs that are typically ineligible for caching, but the system caches them. Determines if a request should be evaluated normally according to caching rules. • Exclude List. Lists the URIs that are typically eligible for caching, but the system does not cache them. • Include Override List. Lists URIs to cache, though typically, they would not be cached due to defined constraints, for example, the Maximum Object Size setting. The default value is none. URIs in the Include Override List list are cacheable even if they are not specified in the Include List. ***Note:*** *You can use regular expressions to specify URIs in accordance with BIG-IP supported meta characters.*
Ignore Headers	All	Specifies how the system processes client-side `Cache-Control` headers when caching is enabled. • None. Specifies that the system honors all `Cache-Control` headers. • Cache-Control:max-age. Specifies that the system disregards a `Cache-Control:max-age` request header that has a value of `max-age=0`. • All. Specifies that the system disregards all `Cache-Control` headers.
Insert Age Header	Enabled	Specifies, when enabled, that the system inserts `Date` and `Age` headers in the cached entry. The `Date` header contains the current date and time on the BIG-IP® system. The `Age` header contains the length of time that the content has been in the cache.
Aging Rate	9	Specifies how quickly the system ages a cache entry. The aging rate ranges from 0 (slowest aging) to 10 (fastest aging).
AM Applications	No default	Lists enabled Application Acceleration Manager applications in the Enabled field and available applications in the Available field.

Web Acceleration Profile statistics description

This topic provides a description of Web Acceleration Profile statistics produced in tmsh.

Viewing Web Acceleration profile statistics

Statistics for the Web Acceleration Profile can be viewed in tmsh by using the following command.

```
tmsh show /ltm profile web-acceleration <profile_name>
```

Each statistic is described in the following tables.

Table 1: Virtual server statistics

Statistic	Description
Virtual Server	The name of the associated virtual server.

Table 2: Cache statistics

Statistic	Description
Cache Size (in Bytes)	The cache size for all objects.
Total Cached Items	The total number of objects cached in the local cache for each TMM.
Total Evicted Items	The total number of objects evicted from cache.
Inter-Stripe Size (in Bytes)	The inter-stripe cache size.
Inter-Stripe Cached Items	The total number of objects in the inter-stripe caches for each TMM.
Inter-Stripe Evicted Items	The total number of objects evicted from the caches for each TMM.

Table 3: Cache Hits/Misses statistics

Statistic	Description
Hits	The total number of cache hits.
Misses (Cacheable)	The number of cache misses for objects that can otherwise be cached.
Misses (Total)	The number of cache misses for all objects.
Inter-Stripe Hits	The number of inter-stripe cache hits for each TMM.
Inter-Stripe Misses	The number of inter-stripe cache misses for each TMM.
Remote Hits	The number of cache hits for owner TMMs.
Remote Misses	The number of cache misses for owner TMMs.

About FTP profiles

The BIG-IP® system includes a profile type that you can use to manage File Transfer Protocol (FTP) traffic. You can tailor FTP profile settings to your specific needs. For those settings that have default values, you

can retain those default settings or modify them. You can modify any settings either when you create the profile, or at any time after you have created it.

The Translate Extended value

Because IP version 6 addresses are not limited to 32 bits (unlike IP version 4 addresses), compatibility issues can arise when using FTP in mixed IP-version configurations.

By default,the BIG-IP system automatically translates FTP commands when a client-server configuration contains both IP version 4 (IPv4) and IP version 6 (IPv6) systems. For example, if a client system running IPv4 sends the FTP `PASV` command to a server running IPv6, the BIG-IP system automatically translates the `PASV` command to the equivalent FTP command for IPv6 systems, `EPSV`.

The BIG-IP system translates the FTP commands `EPRV` and `PORT` in the same way.

Inherit Parent Profile

When you configure the BIG-IP® system to process FTP traffic, the FTP virtual server fully proxies the control channel, allowing you to use the optimization settings of the client-side and server-side TCP profiles assigned to the virtual server.

However, the profile settings of the FTP control channel are not passed down to the FTP data channel by default. Instead, the FTP data channel uses a Fast L4 flow, which is fully accelerated by Packet Velocity ASIC to maximize performance (on applicable hardware platforms). A data channel using Fast L4 cannot use the same full-proxy TCP optimizations that exist for the control channel.

To take advantage of these optimizations for the FTP data channel, you can enable the Inherit Parent Profile setting of the FTP profile. Enabling this setting disables Fast L4 for the FTP data channel, and instead allows the data channel to use the same TCP profile settings that the control channel uses.

Data Port

The Data Port setting allows the FTP service to run on an alternate port. The default data port is 20.

Security for FTP traffic

When the BIG-IP system includes a license for the BIG-IP® Application Security Manager™, you can enable a security scan for FTP traffic.

About DNS profiles

You can create a custom DNS profile to enable various features such as converting IPv6-formatted addresses to IPv4 format, enabling DNS Express™, and enabling DNSSEC.

About RTSP profiles

The BIG-IP system® includes a profile type that you can use to manage Real Time Streaming Protocol (RTSP) traffic. *Real Time Streaming Protocol (RTSP)* is a protocol used for streaming-media presentations. Using RTSP, a client system can control a remote streaming-media server and allow time-based access to files on a server.

The RTSP profile in the BIG-IP system supports these features:

- The setup of streaming media over UDP. In this case, the control connection opens the required ports to allow data to flow through the BIG-IP® system.
- Interleaved data over the control connection, essentially streaming media over TCP.
- Real Networks tunneling of RTSP over HTTP, through the RTSP port (554).

A common configuration for the RTSP profile is one that includes RTSP clients and media servers, as well as RTSP proxies to manage accounting and authentication tasks. In this proxied configuration, you most likely want the streaming media from the servers to pass directly to the client, bypassing the RTSP proxy servers.

To implement this configuration, you configure the BIG-IP system by creating two virtual servers, one for processing traffic to and from the external network, and one for processing traffic to and from the internal network. For each virtual server, you assign a separate RTSP profile.

With this configuration:

- The RTSP profile on the external virtual server passes client IP address information to the RTSP profile on the internal virtual server.
- The RTSP profile on the internal virtual server extracts the client IP address information from the request, processes the media server's response, and opens the specified ports on the BIG-IP system. Opening these ports allows the streaming media to bypass the RTSP proxy servers as the data travels from the server to the client.

The client IP address information is stored in the Proxy Header setting that you specify in the RTSP profile.

About ICAP profiles

You can configure one or more Internet Content Adaptation Protocol (ICAP) profiles when you want to use the BIG-IP® content adaptation feature for adapting HTTP requests and responses. This feature allows a BIG-IP virtual server to conditionally forward HTTP requests and HTTP responses to a pool of ICAP servers for modification, before sending a request to a web server or returning a response to the client system.

In a typical configuration, you create two ICAP profiles:

- You assign one of the profiles to a virtual server of type Internal that sends HTTP requests to a pool of ICAP servers.
- You assign the other profile to a virtual server of type Internal that sends HTTP responses to a pool of ICAP servers.

For more information on content adaptation for HTTP traffic, see the guide titled *BIG-IP® Local Traffic Manager: Implementations*, available on the AskF5™ knowledge base at `http://support.f5.com`.

About Request Adapt and Response Adapt profiles

You can configure a Request Adapt or Response Adapt profile when you want to use the BIG-IP® content adaptation feature for adapting HTTP requests and responses. A Request Adapt or Response Adapt profile instructs an HTTP virtual server to send a request or response to a named virtual server of type Internal, for possible modification by an Internet Content Adaptation Protocol (ICAP) server.

For more information on content adaptation for HTTP traffic, see the guide titled *BIG-IP® Local Traffic Manager: Implementations*, available on the AskF5™ knowledge base at `http://support.f5.com`.

About RADIUS profiles

The BIG-IP® system includes a profile type that you can use to load balance Remote Authentication Dial-In User Service (RADIUS) traffic.

When you configure a RADIUS type of profile, the BIG-IP system can send client-initiated RADIUS messages to load balancing servers. The BIG-IP system can also ensure that those messages are persisted on the servers.

About SMTP profiles

You can create an SMTP profile to secure SMTP traffic coming into the BIG-IP system. When you create an SMTP profile, BIG-IP® Protocol Security Manager™ provides several security checks for requests sent to a protected SMTP server:

- Verifies SMTP protocol compliance as defined in RFC 2821.
- Validates incoming mail using several criteria.
- Inspects email and attachments for viruses.
- Applies rate limits to the number of messages.
- Validates DNS SPF records.
- Prevents directory harvesting attacks.
- Disallows or allows some of the SMTP methods, such as VRFY, EXPN, and ETRN, that spam senders typically use to attack mail servers.
- Rejects the first message from a sender, because legitimate senders retry sending the message, and spam senders typically do not. This process is known as *greylisting*. The system does not reject subsequent messages from the same sender to the same recipient.

With an SMTP profile configured, the system either generates an alarm for, or blocks, any requests that trigger the security check.

Note: *The SMTP profile is only available for BIG-IP systems that are licensed for BIG-IP® Protocol Security Manager™.*

About SMTPS profiles

The SMTPS profile provides a way to add SSL encryption to SMTP traffic quickly and easily. *SMTPS* is a method for securing Simple Mail Transport Protocol (SMTP) connections at the transport layer.

Normally, SMTP traffic between SMTP servers and clients is unencrypted. This creates a privacy issue because SMTP traffic often passes through routers that the servers and clients do not trust, resulting in a third party potentially changing the communications between the server and client. Also, two SMTP systems do not normally authenticate each other. A more secure SMTP server might only allow communications from other known SMTP systems, or the server might act differently with unknown systems.

To mitigate these problems, the BIG-IP system includes an SMTPS profile that you can configure. When you configure an SMTPS profile, you can activate support for the industry-standard STARTTLS extension to the SMTP protocol, by instructing the BIG-IP system to either allow, disallow, or require STARTTLS

activation for SMTP traffic. The STARTTLS extension effectively upgrades a plain-text connection to an encrypted connection on the same port, instead of using a separate port for encrypted communication.

This illustration shows a basic configuration of a BIG-IP system that uses SMTPS to secure SMTP traffic between the BIG-IP system and an SMTP mail server.

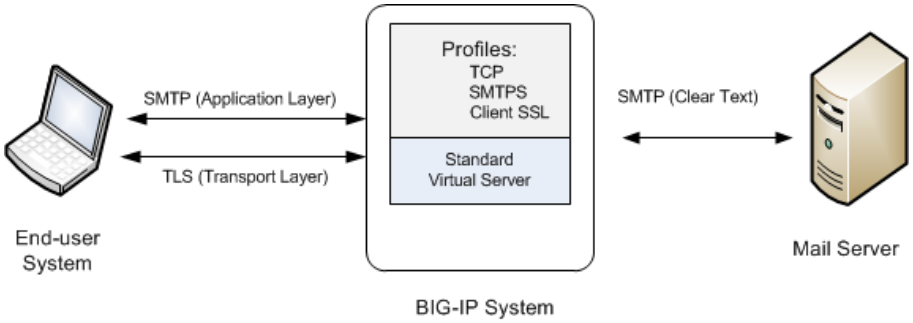

Figure 1: Sample BIG-IP configuration for SMTP traffic with STARTTLS activation

About Client LDAP and Server LDAP profiles

You can implement STARTTLS encryption for Lightweight Directory Access Protocol (LDAP) traffic passing through the BIG-IP® system. *LDAP* is an industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. You configure the BIG-IP system for STARTTLS encryption by activating the STARTTLS communication protocol for any client or server traffic that allows or requires STARTTLS encryption.

Normally, LDAP traffic between LDAP servers and clients is unencrypted. This creates a privacy issue because LDAP traffic often passes through routers that the servers and clients do not trust, resulting in a third party potentially changing the communications between the server and client. Also, two LDAP systems do not normally authenticate each other. A more secure LDAP server might only allow communications from other known LDAP systems, or the server might act differently with unknown systems.

To mitigate these problems, the BIG-IP system includes two LDAP profiles that you can configure. When you configure a Client LDAP or Server LDAP profile, you can instruct the BIG-IP system to activate the STARTTLS communication protocol for any client or server traffic that allows or requires STARTTLS encryption. The *STARTTLS* protocol effectively upgrades a plain-text connection to an encrypted connection on the same port (port 389), instead of using a separate port for encrypted communication.

This illustration shows a basic configuration of a BIG-IP system that activates STARTTLS to secure LDAP traffic between a client system and the BIG-IP system, and between the BIG-IP system and an LDAP authentication server.

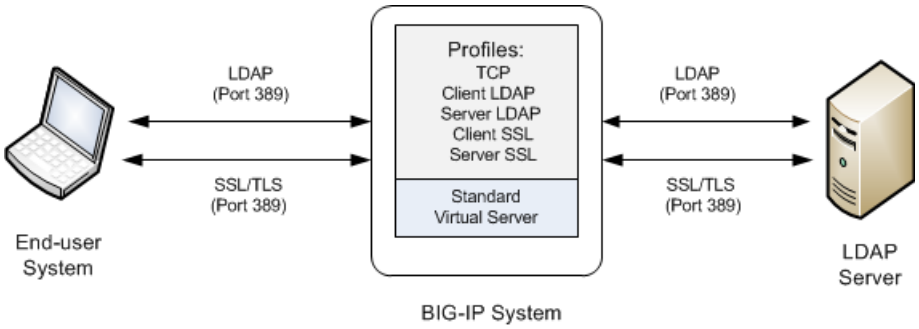

Figure 2: Sample BIG-IP configuration for LDAP traffic with STARTTLS activation

About iSession profiles

The iSession™ profile tells the system how to optimize traffic. Symmetric optimization requires an iSession profile at both ends of the iSession connection. The system-supplied parent iSession profile `isession`, is appropriate for all application traffic, and other iSession profiles have been pre-configured for specific applications. The name of each pre-configured iSession profile indicates the application for which it was configured, such as `isession-cifs`.

When you configure the iSession local endpoint on the Quick Start screen, the system automatically associates the system-supplied iSession profile `isession` with the iSession listener `isession-virtual` it creates for inbound traffic.

You must associate an iSession profile with any virtual server you create for a custom optimized application for outbound traffic, and with any iSession listener you create for inbound traffic.

Screen capture showing compression settings

The following screen capture shows the pertinent compression settings.

Note: *If adaptive compression is disabled, you must manually select a compression codec for iSession™ traffic. If you leave the other codecs enabled, the BIG-IP® system selects the bzip2 compression algorithm by default, and that might not be the algorithm you want.*

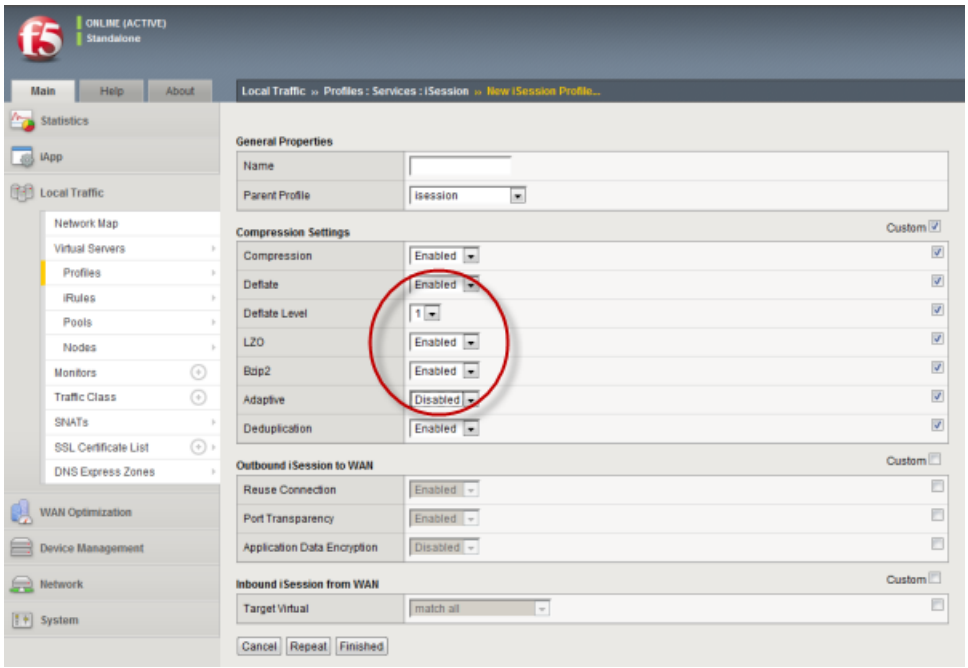

Figure 3: iSession profile screen with compression settings emphasized

About Rewrite profiles

For environments that use web servers, you might want your websites to appear differently on the external network than on the internal network. For example, you might want the BIG-IP® system to send traffic destined for `www.company.com/usa/` to the internal server `usa.company.com` instead. Normally, this translation could cause some issues, such as the web server expecting to see a certain host name (such as for name-based virtual hosting) or the web server using the internal host name when sending a redirect to client systems.

You can solve these problems by configuring a *Rewrite profile*, which causes the BIG-IP system to act as a reverse proxy server. As a *reverse proxy server*, the BIG-IP system offloads the URI translation function from web servers enabled with features such as Apache's ProxyPass module. With a Rewrite profile, the BIG-IP system can perform URI scheme, host, port, and path modifications as HTTP traffic passes through the system. The feature also provides reverse translation for the `Location`, `Content-Location`, and URI headers in the server response to the client.

Important: *The BIG-IP reverse proxy feature replaces the ProxyPass iRule available on the F5 Networks site* `http://devcentral.f5.com`.

A typical use of a reverse proxy server is to grant Internet users access to application servers that are behind a firewall and therefore have private IP addresses and unregistered DNS entries.

About URI translation

You can configure the BIG-IP® system to perform URI translation on HTTP requests. Suppose that a company named `Siterequest` has a website `www.siterequest.com`, which has a public IP address and a registered DNS entry, and therefore can be accessed from anywhere on the Internet.

Furthermore, suppose that `Siterequest` has two application servers with private IP addresses and unregistered DNS entries, inside the company's firewall. The application servers are visible within the internal network as `appserver1.siterequest.com` and `appserver2.siterequest.com`.

Because these servers have no public DNS entries, any client system that tries to access one of these servers from outside the company network receives a `no such host` error.

As the illustration shows, you can prevent this problem by configuring the BIG-IP system to act as a reverse proxy server:

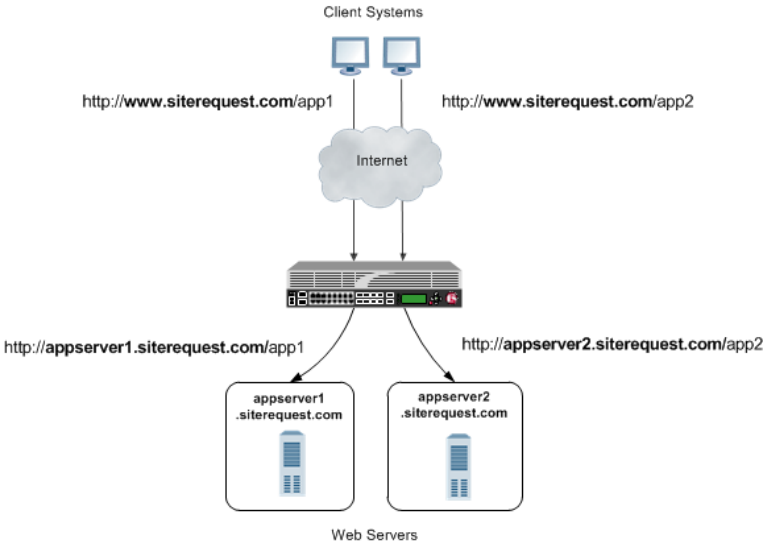

Figure 4: The BIG-IP system as a reverse proxy server for URI translation

In the example, the company `Siterequest` has decided to enable Web access to the internal application servers, without exposing them to the Internet directly. Instead, the company has integrated the servers with the web server `siterequest.com` so that `http://www.siterequest.com/sales` is mapped internally to `http://appserver1.siterequest.com/sales`, and `http://siterequest.com/marketing` is mapped internally to `http://appserver2.example.com/marketing`. This is a typical reverse-proxy configuration.

To configure the BIG-IP system to perform this translation, you create a Rewrite profile and configure one or more URI rules. A *URI rule* specifies the particular URI translation that you want the BIG-IP system to perform. Specifically, a URI rule translates the scheme, host, port, or path of any client URI, server URI, or both. A URI rule also translates any domain and path information in the `Set-Cookie` header of the response when that header information matches the information in the URI rule.

Rules for matching requests to URI rules

The BIG-IP® system follows these rules when attempting to match a request to a URI rule:

- A request does not need to match any entry. That is, if no entries match and there is no catch-all entry, then the Rewrite profile has no effect.
- Each request matches one entry only, which is the entry with the most specific host and path.
- If multiple entries match, then the BIG-IP system uses the entry with the deepest path name on the left side of the specified mapping.
- The BIG-IP system matches those requests that contain host names in URIs before matching requests that do not contain host names in URIs.
- The BIG-IP system processes the specified entries in the mapping from most-specific to least-specific, regardless of the order specified in the actual Rewrite profile.

About URI Rules

When creating a URI rule, you must specify the client and server URIs in these ways:

- When the URI is a path prefix only, the path must be preceded by and followed by a `/`, for example, `/sales/`.
- When the URI contains more than the path prefix (such as, a host), the URI must also contain a scheme and must be followed by a `/`, for example, `http://www.siterequest/sales/`.

About Set-Cookie header translation

A URI rule automatically performs translation on any domain and path information in the `Set-Cookie` header of a response when that header information matches the information in the URI rule.

When the `Set-Cookie` header information that you want the BIG-IP® system to translate does not match the information in an existing URI rule, you can create a separate *Set-Cookie rule* to perform this translation. You need to create a `Set-Cookie` rule only when the header information does not match the information specified in an existing URI rule.

The specific parts of the `Set-Cookie` header that you can specify for translation are:

- Client domain
- Client path
- Server domain
- Server path

You can specify that the BIG-IP system translate all of this information or a subset of this information, depending on your needs.

About XML profiles

You can use the BIG-IP® system to perform XML content-based routing whereby the system routes requests to an appropriate pool, pool member, or virtual server based on specific content in an XML document. For example, if your company transfers information in XML format, you could use this feature to examine the XML content with the intent to route the information to the appropriate department.

You can configure content-based routing by creating an XML profile and associating it with a virtual server. In the XML profile, define the matching content to look for in the XML document. Next, specify how to route the traffic to a pool by writing simple iRules®. When the system discovers a match, it triggers an iRule event, and then you can configure the system to route traffic to a virtual server, a pool, or a node.

The following example shows a simple XML document that the system could use to perform content-based routing. It includes an element called FinanceObject used in this implementation.

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eai="http://192.168.149.250/eai_enu/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
     <eai:SiebelEmployeeDelete
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
             <FinanceObject xsi:type="xsd:string">Route to
Financing</FinanceObject>
             <SiebelMessage xsi:type="ns:ListOfEmployeeInterfaceTopElmt"
xmlns:ns="http://www.siebel.com/xml">
```

```
                    <ListOfEmployeeInterface
xsi:type="ns:ListOfEmployeeInterface">
                           <SecretKey>123456789</SecretKey>
                                    <Employee>John</Employee>
                                    <Title>CEO</Title>
                    </ListOfEmployeeInterface>
               </SiebelMessage>
     </eai:SiebelEmployeeDelete>
   </soapenv:Body>
  </soapenv:Envelope>
```

About HTTP2 profiles

You can configure a virtual server with the BIG-IP® system HTTP/2 profile to provide gateway functionality for HTTP 2.0 traffic, minimizing the latency of requests by multiplexing streams and compressing headers.

A client initiates an HTTP/2 request to the BIG-IP system, the HTTP/2 virtual server receives the request on port `443`, and sends the request to the appropriate server. When the server provides a response, the BIG-IP system compresses and caches it, and sends the response to the client.

Important: *The BIG-IP system supports HTTP/2 for client-side connections only. This means that when a client that supports HTTP/2 connects to a virtual server that has an HTTP/2 profile assigned to it, the resulting server-side traffic (such as traffic sent to pool members) is sent over HTTP/1.1.*

Source address persistence is not supported by the HTTP/2 profile.

Summary of HTTP/2 profile functionality

By using the HTTP/2 profile, the BIG-IP system provides the following functionality for HTTP/2 requests.

Creating concurrent streams for each connection.
You can specify the maximum number of concurrent HTTP requests that are accepted on a HTTP/2 connection. If this maximum number is exceeded, the system closes the connection.

Limiting the duration of idle connections.
You can specify the maximum duration for an idle HTTP/2 connection. If this maximum duration is exceeded, the system closes the connection.

Enabling a virtual server to process HTTP/2 requests.
You can configure the HTTP/2 profile on the virtual server to receive HTTP, SPDY, and HTTP/2 traffic, or to receive only HTTP/2 traffic, based in the activation mode you select. (Note the HTTP/2 profile to receive only HTTP/2 traffic is primarily intended for troubleshooting.)

Inserting a header into the request.
You can insert a header with a specific name into the request. The default name for the header is `X-HTTP/2`.

Important: *The HTTP/2 protocol is incompatible with NTLM protocols. Do not use the HTTP/2 protocol with NTLM protocols.*

About HTTP/2 profiles

The BIG-IP® system includes an HTTP/2 profile type that you can use to manage HTTP/2 traffic, improving the efficiency of network resources while reducing the perceived latency of requests and responses. The LTM HTTP/2 profile enables you to achieve these advantages by multiplexing streams and compressing headers with Transport Layer Security (TLS) or Secure Sockets Layer (SSL) security.

The HTTP/2 protocol uses a binary framing layer that defines a frame type and purpose in managing requests and responses. The binary framing layer determines how HTTP messages are encapsulated and transferred between the client and server, a significant benefit of HTTP 2.0 when compared to earlier versions.

All HTTP/2 communication occurs by means of a connection with bidirectional streams. Each stream includes messages, consisting of one or more frames, that can be interleaved and reassembled using the embedded stream identifier within each frame's header. The HTTP/2 profile enables you to specify a maximum frame size and write size, which controls the total size of combined data frames, to improve network utilization.

Multiplexing streams

You can use the HTTP/2 profile to multiplex streams (interleaving and reassembling the streams), by specifying a maximum number of concurrent streams permitted for a single connection. Also, because multiplexing streams on a single TCP connection compete for shared bandwidth, you can use the profile's Priority Handling settings to configure stream prioritization and define the relative order of delivery. For example, a Strict setting processes higher priority streams to completion before processing lower priority streams; whereas, a Fair setting allows higher priority streams to use more bandwidth than lower priority streams, without completely blocking the lower priority streams.

Additionally, you can specify the way that the HTTP/2 profile controls the flow of streams. The Receive Window setting allows HTTP/2 to stall individual upload streams, as needed. For example, if the BIG-IP system is unable to process a slow stream on a connection, but is able to process other streams on the connection, it can use the Receive Window setting to specify a frame size for the slow stream, thus delaying that upload stream until the size is met and the receiver is able to process it, while concurrently proceeding to process frames for another stream.

Compressing headers

When you configure the HTTP/2 profile's Header Table Size setting, you can compress HTTP headers to conserve bandwidth. Compressing HTTP headers reduces the object size, which reduces required bandwidth. For example, you can specify a larger table value for better compression, but at the expense of using more memory.

HTTP/2 profile settings

This table provides descriptions of the HTTP/2 profile settings.

Setting	Default	Description
Name		Specifies the name of the HTTP/2 profile.
Parent Profile	**http2**	Specifies the profile that you want to use as the parent profile. Your new profile inherits all settings and values from the parent profile specified.
Concurrent Streams Per Connection	`10`	Specifies the number of concurrent requests allowed to be outstanding on a single HTTP/2 connection.

Setting	Default	Description
Connection Idle Timeout	300	Specifies the number of seconds an HTTP/2 connection is left open idly before it is closed.
Insert Header	**Disabled**	Specifies whether an HTTP header that indicates the use of HTTP/2 is inserted into the request sent to the origin web server.
Insert Header Name	X-HTTP/2	Specifies the name of the HTTP header controlled by the **Insert Header Name** setting.
Activation Modes	**Select Modes**	Specifies how a connection is established as a HTTP/2 connection.
Selected Modes	**ALPN NPN**	Used only with an **Activation Modes** selection of **Select Modes**, specifies the extension, **ALPN** for HTTP/2 or **NPN** for SPDY, used in the HTTP/2 profile. The order of the extensions in the **Selected Modes Enabled** list ranges from most preferred (first) to least preferred (last). Clients typically use the first supported extension. At least one HTTP/2 mode must be included in the **Enabled** list. The values **ALPN** and **NPN** specify that the TLS Application Layer Protocol Negotiation (ALPN) and Next Protocol Negotiation (NPN) will be used to determine whether HTTP/2 or SPDY should be activated. Clients that use TLS, but only support HTTP will work as if HTTP/2 is not present. The value **Always** specifies that all connections function as HTTP/2 connections. Selecting **Always** in the **Activation Mode** list is primarily intended for troubleshooting.
Priority Handling	**Strict**	Specifies how the HTTP/2 profile handles priorities of concurrent streams within the same connection. Selecting **Strict** processes higher priority streams to completion before processing lower priority streams. Selecting **Fair** enables higher priority streams to use more bandwidth than lower priority streams, without completely blocking the lower priority streams.
Receive Window	32	Specifies the *receive window*, which is HTTP/2 protocol functionality that controls flow, in KB. The receive window allows the HTTP/2 protocol to stall individual upload streams when needed.
Frame Size	2048	Specifies the size of the data frames, in bytes, that the HTTP/2 protocol sends to the client. Larger frame sizes improve network utilization, but can affect concurrency.
Write Size	16384	Specifies the total size of combined data frames, in bytes, that the HTTP/2 protocol sends in a single write function. This setting controls the size of the TLS records when the HTTP/2 protocol is used over Secure Sockets Layer (SSL). A large write size causes the HTTP/2 protocol to buffer more data and improves network utilization.
Header Table Size	4096	Specifies the size of the header table, in KB. The HTTP/2 protocol compresses HTTP headers to save bandwidth. A larger table size allows better compression, but requires more memory.

About SPDY profiles

You can use the BIG-IP® system SPDY (pronounced "speedy") profile to minimize latency of HTTP requests by multiplexing streams and compressing headers. When you assign a SPDY profile to an HTTP virtual

server, the HTTP virtual server informs clients that a SPDY virtual server is available to respond to SPDY requests.

When a client sends an HTTP request, the HTTP virtual server manages the request as a standard HTTP request. It receives the request on port `80`, and sends the request to the appropriate server. When the server provides a response, the BIG-IP system inserts an HTTP header into the response (to inform the client that a SPDY virtual server is available to handle SPDY requests), compresses and caches it, and sends the response to the client.

A client that is enabled to use the SPDY protocol sends a SPDY request to the BIG-IP system, the SPDY virtual server receives the request on port `443`, converts the SPDY request into an HTTP request, and sends the request to the appropriate server. When the server provides a response, the BIG-IP system converts the HTTP response into a SPDY response, compresses and caches it, and sends the response to the client.

Summary of SPDY profile functionality

By using the SPDY profile, the BIG-IP system provides the following functionality for SPDY requests.

Creating concurrent streams for each connection.
You can specify the maximum number of concurrent HTTP requests that are accepted on a SPDY connection. If this maximum number is exceeded, the system closes the connection.

Limiting the duration of idle connections.
You can specify the maximum duration for an idle SPDY connection. If this maximum duration is exceeded, the system closes the connection.

Enabling a virtual server to process SPDY requests.
You can configure the SPDY profile on the virtual server to receive both HTTP and SPDY traffic, or to receive only SPDY traffic, based in the activation mode you select. (Note that setting this to receive only SPDY traffic is primarily intended for troubleshooting.)

Inserting a header into the response.
You can insert a header with a specific name into the response. The default name for the header is `X-SPDY`.

Important: *The SPDY protocol is incompatible with NTLM protocols. Do not use the SPDY protocol with NTLM protocols. For additional details regarding this limitation, please refer to the SPDY specification:* *`http://dev.chromium.org/spdy/spdy-authentication`.*

SPDY profile settings

This table provides descriptions of the SPDY profile settings.

Setting	Default	Description
Name		Type the name of the SPDY profile.
Parent Profile	**spdy**	Specifies the profile that you want to use as the parent profile. Your new profile inherits all settings and values from the parent profile specified.
Concurrent Streams Per Connection	`10`	Specifies how many concurrent requests are allowed to be outstanding on a single SPDY connection.
Connection Idle Timeout	`300`	Specifies how many seconds a SPDY connection is left open idly before it is closed.

Setting	Default	Description
Activation Mode	**NPN**	Specifies how a connection is established as a SPDY connection. The value **NPN** specifies that the Transport Layer Security (TLS) Next Protocol Negotiation (NPN) extension determines whether the SPDY protocol is used. Clients that use TLS, but only support HTTP will work as if SPDY is not present. The value **Always** specifies that all connections must be SPDY connections, and that clients only supporting HTTP are not able to send requests. Selecting **Always** in the **Activation Mode** list is primarily intended for troubleshooting.
Insert Header	**Disabled**	Specifies whether an HTTP header that indicates the use of SPDY is inserted into the request sent to the origin web server.
Insert Header Name	`X-SPDY`	Specifies the name of the HTTP header controlled by the **Insert Header Name** setting.
Protocol Versions	**All Versions Enabled**	Used only with an **Activation Mode** selection of **NPN**, specifies the protocol and protocol version (**http1.1**, **spdy2**, **spdy3**, **spdy3.1**, or **All Version Enabled**) used in the SPDY profile. The order of the protocols in the **Selected Versions Enabled** list ranges from most preferred (first) to least preferred (last). Adding **http1.1** to the **Enabled** list allows HTTP1.1 traffic to pass. If **http1.1** is not added to the **Enabled** list, clients that do not support http1.1 are blocked. Clients typically use the first supported protocol. At least one SPDY version must be included in the **Enabled** list.
Priority Handling	**Strict**	Specifies how the SPDY profile handles priorities of concurrent streams within the same connection. Selecting **Strict** processes higher priority streams to completion before processing lower priority streams. Selecting **Fair** enables higher priority streams to use more bandwith than lower priority streams, without completely blocking the lower priority streams.
Receive Window	`32`	Specifies the *receive window*, which is SPDY protocol functionality that controls flow, in KB. The receive window allows the SPDY protocol to stall individual upload streams when needed. This functionality is only available in SPDY3.
Frame Size	`2048`	Specifies the size of the data frames, in bytes, that the SPDY protocol sends to the client. Larger frame sizes improve network utilization, but can affect concurrency.
Write Size	`16384`	Specifies the total size of combined data frames, in bytes, that the SPDY protocol sends in a single write function. This setting controls the size of the TLS records when the SPDY protocol is used over Secure Sockets Layer (SSL). A large write size causes the SPDY protocol to buffer more data and improves network utilization.

SOCKS profiles

You can use the BIG-IP® system SOCKS profile to configure the BIG-IP system to handle proxy requests and function as a gateway. By configuring browser traffic to use the proxy, you can control whether to allow or deny a requested connection. To implement the profile, you must associate it with a virtual server.

SOCKS profile settings

Protocol Versions

You can specify one or more versions of SOCKS.

- **Socks4** indicates protocol support for SOCKS version 4.
- **Socks4A** indicates protocol support for SOCKS 4A, which adds host name support to version 4.
- **Socks5** specifies protocol support for SOCKS version 5, which includes host name and IPv6 support.

DNS Resolver

You must specify a DNS resolver to use for DNS inquiries handled by the virtual servers associated with this profile. If no DNS resolver exists on the system, you can create one at **DNS** > **Caches** > **Cache List** > **Create**.

Route Domain

You can specify a route domain to be used for outbound connect requests.

Tunnel Name

You must specify a tunnel that is used for outbound connect requests, enabling other virtual servers to receive connections initiated by the proxy service. A pre-configured tunnel `socks-tunnel` is available.

Default Connect Handling

You can specify the behavior of the proxy service when handling outbound requests.

- Enabled (checked) indicates that the proxy service delivers outbound requests directly, regardless of the presence of listening servers.
- Disabled (check box cleared) indicates that the proxy service delivers outbound requests only if another virtual server is listening on the tunnel for the requested outbound connection. With this setting, virtual servers are required, and the system processes the outbound traffic before it leaves the device.

About FIX profiles

The BIG-IP® system FIX profile provides you with the ability to use Financial Information eXchange (FIX) protocol messages in routing, load balancing, persisting, and logging connections. The BIG-IP system uses the FIX profile to examine the header, body, and footer of each FIX message, and then process each message according to the parameters that it contains.

The BIG-IP system supports FIX protocol versions 4.2, 4.4, and 5.0, and uses the key-value pair FIX message format.

Important: *You cannot configure or use the BIG-IP FIX Profile to provide low-latency electronic trading functionality. Instead, you must implement low-latency electronic trading functionality separately.*

About FIX profile tag substitution

The BIG-IP® system's FIX profile provides options for how the FIX messages should be parsed. Once configured, the BIG-IP system compares the FIX profile's Mapping List Sender value (`SenderCompID`) with the value received in the client message. If the values match, then the BIG-IP system provides tag substitution as defined by the data group definition in the corresponding mapping list.

Example

Two or more clients can define a FIX tag differently. On the BIG-IP server side, you can define a dictionary for each client that maps a client tag to a server tag. For example, a server might use `20001` for an analyst's average price target, and `20002` as a client twitter feed name. Then, in the dictionary for the first client, the tag `10001` is mapped to `20001`, and, for the second client, the tag `30001` is mapped to `20001`.

About steering traffic using the FIX profile

The BIG-IP® system's FIX profile can direct, or steer, FIX messages to a destination pool in accordance with the FIX login message that it receives, and the configured iRules®. Once a pool member is selected, which is only required one time for a connection, all messages in the same FIX session are forwarded, or persisted, to that pool member.

About validating FIX messages

The BIG-IP® system validates each Financial Information eXchange (FIX) protocol message, allowing and denying transmission accordingly. If a FIX message is valid, the BIG-IP system allows transmission, triggers the `FIX_MESSAGE` iRule event, and optionally logs the message. If a FIX message is invalid, the BIG-IP system logs the error, and either disallows transmission or drops the connection, as configured by the profile.

The BIG-IP system provides two types of parsing validation: full parsing validation and quick parsing validation.

Full Parsing Validation

When *full parsing validation* is applied, all fields are validated.

Quick Parsing Validation

When *quick parsing validation* is applied, the following fields are validated.

- The first three fields: 8 (BeginString), 9 (BodyLength), and 35 (MsgType).
- The last field.
- Field 49 (SenderCompID).
- Fields requested by an iRule tag query command.
- Fields in the message that precede the fields requested by an iRule tag query command.

For example, consider the following message: `8=FIX.4.2|9=100|35-A|600=X|700=Y|800=Z...`. In this example, the first three fields are always parsed: `8`, `9`, and `35`. If the iRule command `FIX::tag 700` runs, then the fields preceding `700` in the example are parsed, specifically `600` (in addition to the first three fields).

The following table describes the different types of quick parsing validation that the BIG-IP system provides.

FIX Message	Description	Quick Parsing Validation	Example
`Message sequence no <num> from <senderCompID> error: There is no = in the field starting at byte <byte offset of the field>`	Field is not in the format of `tag=val`.	This error is partially checked when using quick parsing validation.	`35=A;123xyz;`. The second field is missing an = sign.

FIX Message	Description	Quick Parsing Validation	Example
`Message sequence no <num> from <senderCompID> error: the field starting at byte <byte offset of the field> has invalid tag`	The tag is not an integer.	This error is partially checked when using quick parsing validation.	`35=A;abc=xyz;`. The tag `abc` in the second field is not an integer.
`Message sequence no <num> from <senderCompID> error: there is no value found in the field starting at byte <byte offset of the field>`	A value is missing.	This error is partially checked when using quick parsing validation.	`35=A;50=;`. The second field is missing a value.
`The first (second, third) tag should be 8 (9, 35), but get <wrong value> from < senderCompID>`	The first three tags are not `8`, `9`, and `35`.	This error is fully checked when using quick parsing validation.	None.
`Length mismatch: message sequence no <num> from <senderCompID> should be tag10 after <length> bytes, but encounter <val1 val2>`	The length is mismatched.	This error is fully checked when using quick parsing validation.	None.
`Checksum mismatch: message sequence <num> from <senderCompID> declares checksum as <claimed value>, but calculated checksum from received data is <real value>`	The checksum is mismatched.	This error is fully checked when using quick parsing validation.	None.
`Message from <IP address> is longer than allowed`	The message length is greater than 4MB.	This error is fully checked when using quick parsing validation.	None.

About using SSL encryption for FIX messages

You can configure a virtual server to use client and server SSL encryption with FIX protocol messages, as necessary, for transactions across the Internet, or for compliance purposes.

About logging FIX messages

The BIG-IP® system provides optional logging of each FIX message for auditing purposes. You can log events either locally on the BIG-IP system or remotely, using the BIG-IP system's high-speed logging mechanism. The recommended way to store logs is on a pool of remote logging servers.

For local logging, the high-speed logging mechanism stores the logs in either the Syslog or the MySQL database on the BIG-IP system, depending on a destination that you define. For remote logging, the high-speed logging mechanism sends log messages to a pool of logging servers that you define.

Report Log Publisher

The report log publisher setting enables you to log any error messages for FIX traffic, either locally, by using the default **local-db-publisher**, or remotely, by using high-speed logging.

Message Log Publisher

The message log publisher setting enables you to log all FIX messages, either locally, by using the default **local-db-publisher**, or remotely, by using high-speed logging.

About FIX profile statistics

The BIG-IP® system's FIX profile provides statistics that enable you to evaluate and analyze the characteristics of FIX traffic. In addition to virtual server statistics, the following table describes statistics that are specific to the FIX profile.

Statistic	Description
`Current connections`	Specifies the current number of FIX connections.
`Number messages`	Specifies the total number of FIX messages.
`Total message size`	Specifies the total size for all FIX messages.
`Number messages last interval`	Specifies the number of FIX messages sent during the last interval.

You can view statistics for the FIX profile by using tmsh, for example, by typing `tmsh show ltm profile fix <fix_profile_name>` to view a summary of FIX traffic statistics, or `tmsh show sys fix-connection` to view FIX traffic statistics for each client.

About GTP profiles

You can create a GPRS Tunneling Protocol (GTP) profile type on the BIG-IP® system to manage Global System for Mobile Communication (GSM), Universal Mobile Telecommunications System (UMTS), and latterly Long-Term Evolution (LTE) subscriber traffic across User Datagram Protocol (UDP) connections. The BIG-IP system supports GTP versions 1 and 2 on UDP connections. When configuring the GTP profile, you can specify the maximum number of messages held in the GTP ingress queue.

About WebSocket profiles

You can use the BIG-IP® system to manage WebSocket traffic. When you assign a WebSocket profile to a virtual server, the virtual server informs clients that a WebSocket virtual server is available to respond to WebSocket requests.

WebSocket frames that contain payload data are masked with a 32-bit key. You can detemine what the BIG-IP system does with this key by specifying one of these values:

Table 4: WebSocket Masking Settings

Option	When you want to do this
Preserve	Preserve the mask of the packet received, and make no change. ASM and other modules receive masked frames.
Unmask	Remove the mask from the packet and remask it using the same mask when sending the traffic to the server. (Default value)
Remask	Remove the mask received from the client. The system generates a new, random mask when sending the traffic to the server.
Selective	Preserve the mask of the packet received, and make no changes unless an Application Security Policy is associated with the virtual server. In that case, unmask the packet, allow ASM™ to examine the WebSocket payload, and remask it when sending the traffic to the server.

Video Quality of Experience profiles

The BIG-IP® system's video Quality of Experience (QoE) profile enables you to assess an audience's video session or overall video experience, providing an indication of customer satisfaction. The QoE profile uses static information, such as bitrate and duration of a video, and video metadata, such as URL and content type, in monitoring video streaming. Additionally, the QoE profile monitors dynamic information, such as the variable video downloading rate. By measuring the video playing rate and downloading rate, the user experience can be assessed and defined in terms of a single mean opinion score (MOS) of the video session, and a level of customer satisfaction can be derived. QoE scores are logged in the `ltm` log file, located in `/var/log`, which you can evaluate as necessary.

About the video Quality of Experience profile

The BIG-IP® system's video Quality of Experience (QoE) profile enables you to assess an audience's video session or overall video experience, providing an indication of customer satisfaction. The QoE profile uses static information, such as bitrate and duration of a video, and video metadata, such as URL and content

type, in monitoring video streaming. Additionally, the QoE profile monitors dynamic information, which reflects the real-time network condition.

By considering both the static video parameters and the dynamic network information, the user experience can be assessed and defined in terms of a single mean opinion score (MOS) of the video session, and a level of customer satisfaction can be derived. QoE scores are logged in the `ltm` log file, located in `/var/log`, which you can evaluate as necessary.

Note that for QoE to properly process video files, the video web servers must be compliant with supported video MIME types, for example, the following MIME types.

MIME Type	Suffix
`video/mp4`	`.f4v`
`video/mp4`	`.mp4`
`video/x-flv`	`.flv`
`video/x-m4v`	`.m4v`
`video/quicktime`	`.m4v`
`application/x-mpegURL`	`.m3u8`
`video/mp2t`	`.ts`

About mean opinion score

The video Quality of Experience (QoE) profile provides a mean opinion score (MOS), derived from static and dynamic parameters associated with a video stream. The following table summarizes the resultant values.

MOS	Quality	Description
5	Excellent	Indicates a superior level of quality, with imperceptible degradation in the video stream.
4	Good	Indicates an above-average level of quality, with perceptible degradation that is acceptable.
3	Fair	Indicates an average level of quality, with perceptible degradation that detracts from the video experience.
2	Poor	Indicates a below-average level of quality, with perceptible degradation that significantly detracts from the video experience.
1	Bad	Indicates a substandard level of quality, with perceptible degradation that proves to be significantly inferior and potentially unacceptable.

Content Profiles

Introduction to HTML content modification

When you configure an HTML profile on the BIG-IP® system, the system can modify HTML content that passes through the system, according to your specifications. For example, if you want the BIG-IP system to detect all content of type `text/html` and then remove all instances of the HTML `img` tag with the `src` attribute, you can configure an HTML profile accordingly, and assign it to the virtual server. The HTML profile ensures that the BIG-IP system removes those instances of the tag from any HTML content that passes through the virtual server.

Or, you can configure an HTML profile to match on a certain tag and attribute in HTML content when a particular iRule event is triggered, and then create an iRule that includes a command to replace the value of the matched attribute with a different attribute. The BIG-IP system includes several iRule commands that you can use when the `Raise Event on Comment` or `Raise Event on Tag` events are triggered. For more information on iRule commands related to HTML content modification, see the F5 Networks web site `http://devcentral.f5.com`.

HTML tag removal and replacement are just two of several HTML rules that you can configure to manipulate HTML content. An *HTML rule* defines the specific actions that you want the BIG-IP system to perform on a specified type HTML content.

About content selection types

When you create an HTML type of profile, you can specify the type of content that you want the BIG-IP® system to match on when determining which content to modify.

The types of content that you specify must be valid values for the `Content-Type` header of an HTTP response.

Typical `Content-Type` header values that you can specify are:

- text/html
- text/xhtml
- text/xml
- text/javascript

You must specify at least one valid content type if you want the BIG-IP system to match content based on an HTML rule that you create.

Types of HTML rules

You can create several different types of HTML rules for modifying HTML content.

HTML rule type	Description
Remove Comments	Removes comments within HTML content. There are no matching and no actions associated with this type of HTML rule.
Raise Event on Comments	Raises an `HTML_COMMENT_MATCHED` iRule event. There are no matching and no actions associated with this type of HTML rule.
Remove Attribute	Matches on the specified tag name, attribute name, and attribute value, and then removes the HTML tag attribute.
Append HTML	Matches on the specified tag name, attribute name, and attribute value, and then appends the specified HTML content to the tag delimiter.
Prepend HTML	Matches on the specified tag name, attribute name, and attribute value, and then prepends the specified HTML content to the tag delimiter.
Raise Event on Tag	Raises an `HTML_TAG_MATCHED` iRule event. With this type of rule, you can match on the tag name, attribute name, and attribute value.
Remove Tag	Matches on the specified tag name, attribute name, and attribute value, and then removes the HTML tag.

Sample HTML rules configuration

When you create an HTML rule, you can specify the actions that you want the BIG-IP® system to take based on that type of rule. The actions you specify vary depending on the type of HTML rule you are creating.

For example, suppose you want to replace this HTML content:

```
<img src="image/bigip8900.jpg"/>
```

with:

```
<img src="image/bigip8900.png"/>
```

To do this, you can create an HTML profile with a content selection type of `text/html` and the `Raise Event on Tag` rule.

In the example, the `Raise Event on Tag` rule specifies these match settings:

Tag name:	img
Attribute name:	src
Attribute value:	image/bigip8900.jpg

These settings instruct the BIG-IP system to match on any <img> tag that includes the `src` attribute and the attribute value `image/bigip8900.jpg`.

After creating the HTML profile, you can write an iRule that specifies the `HTML_TAG_MATCHED` event, as well as the `HTML::tag attribute replace` command, which specifies the new attribute value. When the traffic reaches the virtual server, the BIG-IP system triggers the event, matches on the tag and attribute specified in the HTML rule, and replaces the old attribute value with the new value specified in the iRule.

Session Persistence Profiles

Introduction to session persistence profiles

Using the BIG-IP® system, you can configure session persistence. When you configure *session persistence*, the BIG-IP system tracks and stores session data, such as the specific pool member that serviced a client request. The primary reason for tracking and storing session data is to ensure that client requests are directed to the same pool member throughout the life of a session or during subsequent sessions.

In addition, session persistence can track and store other types of information, such as user preferences or a user name and password.

The BIG-IP system offers several types of session persistence, each one designed to accommodate a specific type of storage requirement for session data. The type of persistence that you implement depends on where and how you want to store client-specific information, such as items in a shopping cart or airline ticket reservations.

For example, you might store airline ticket reservation information in a back-end database that all servers can access, or on the specific server to which the client originally connected, or in a cookie on the client's machine. When you enable persistence, returning clients can bypass load balancing and instead connect to the server to which they last connected in order to access their saved information.

The BIG-IP system keeps session data for a period of time that you specify.

The primary tool for configuring session persistence is to configure a persistence profile and assign it to a virtual server. If you want to enable persistence for specific types of traffic only, as opposed to all traffic passing through the virtual server, you can write an iRule.

A *persistence profile* is a pre-configured object that automatically enables persistence when you assign the profile to a virtual server. By using a persistence profile, you avoid having to write a program to implement a type of persistence.

Each type of persistence that the BIG-IP system offers includes a corresponding default persistence profile. These persistence profiles each contain settings and setting values that define the behavior of the BIG-IP system for that type of persistence. You can either use the default profile or create a custom profile based on the default.

Persistence profile types

You can configure persistence profile settings to set up session persistence on the BIG-IP® system. You can configure these settings when you create a profile or after profile creation by modifying the profile's settings.

The persistence types that you can enable using a persistence profile are:

Cookie persistence

Cookie persistence uses the HTTP cookie header to persist connections across a session. Most application servers insert a session ID into responses that is used by developers to access data stored in the server session (shopping carts and so on). Load balancing services use this value to enable persistence. This technique prevents the issues associated with simple persistence because the session ID is unique.

Destination address affinity persistence

Also known as sticky persistence, destination address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the destination IP address of a packet.

Hash persistence

Hash persistence allows you to create a persistence hash based on an existing iRule. Hash persistence allows the use of multiple values within a request to enable persistence. To avoid problems with simple persistence, for example, a hash value may be created based on Source IP, Destination IP, Destination Port. While not necessarily unique to every session, this technique results in a more even distribution of load across servers. You generally use this type of persistence technique with stateless applications or streaming content (video and audio).

Microsoft® Remote Desktop Protocol persistence

Microsoft® Remote Desktop Protocol (MSRDP) persistence tracks sessions between clients and servers running the Microsoft® Remote Desktop Protocol (RDP) service.

SIP persistence

SIP persistence is an application-specific type of persistence used for servers that receive Session Initiation Protocol (SIP) messages sent through UDP, SCTP, or TCP. You generally use this persistence technique with stateful applications that depend on the client being connected to the same application instance throughout the life of the session.

Source address affinity persistence

Also known as simple persistence, source address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the source IP address of a packet. You generally use this type of persistence technique with stateless applications or streaming content (video and audio) as a means to more evenly distribute load.

SSL persistence

Because SSL sessions need to be established and are very much tied to a session between client and server, failing to persist SSL-secured sessions results in renegotiation of the session,. Regnegotiation requires a noticeable amount of time and can result in user dissatisfaction. To avoid unnecessary renegotiation, the BIG-IP system uses the SSL session ID to ensure that a session is properly routed to the application instance to which the session first connected. Even when the client's IP address changes, the BIG-IP® system still recognizes the connection as being persistent based on the session ID. You generally use this persistence technique with stateful applications that depend on the client being connected to the same application instance throughout the life of the session.

Universal persistence

Universal persistence uses any piece of data (network, application protocol, payload) to persist a session. This technique requires the BIG-IP system to be able to inspect and ultimately extract any piece of data from a request or response. This technique is the basis for application-specific persistence solutions addressing popular applications like SIP, WTS, and more recently, VMware View. With universal persistence, you can write an expression that defines the data that the BIG-IP system will persist on in a packet. The expression, written using the same expression syntax that you use in iRules®, defines some sequence of bytes to use as a session ID. You generally use this persistence technique with stateful applications that depend on the client being connected to the same application instance throughout the life of the session.

Session persistence and iRules

Instead of configuring a persistence profile, which enables a persistence type for all sessions passing through the virtual server, you can write an iRule, which enables a persistence type for particular requests (for example, for HTTP traffic that includes a certain cookie version only).

You can also use an iRule to enable persistence for SSL-terminated requests, that is, requests that the BIG-IP® system terminates by performing decryption and re-encryption and by handling SSL certificate authentication. In iRules® of this type, you can use an HTTP header insertion iRule command to insert an SSL session ID as a header into an HTTP request.

HTTP requests and sesssion persistence

When you configure a persistence profile on a virtual server, the BIG-IP® system tracks a pointer to the pool member that serviced a client request. Configuring a persistence profile for a virtual server ensures that client requests are directed to the same pool member throughout the lifetime of a session.

By default, the BIG-IP system performs load balancing for each TCP connection, rather than for each HTTP request. After the initial TCP connection is load balanced, the system sends all HTTP requests seen on the same connection to the same pool member. You can change this behavior if you want the system can make a new load balancing decision according to changing persistence information in HTTP requests. You do this by configuring a OneConnect™ profile and assigning the profile to a virtual server. A OneConnect profile causes the system to detach server-side connections so that the system can perform load balancing for each request within the TCP connection and send the HTTP requests to different destination servers if necessary.

Criteria for session persistence

For most persistence types, you can specify the criteria that the BIG-IP® system uses to send all requests from a given client to the same pool member. These criteria are based on the virtual server or servers that are hosting the client connection. To specify these criteria, you configure the **Match Across Services**, **Match Across Virtual Servers**, and **Match Across Pools**settings contained within persistence profiles. Before configuring a persistence profile, it is helpful to understand these settings.

Note: *For the Cookie persistence type, these global settings are only available the Cookie Hash method specifically.*

The Match Across Services setting

When you enable the **Match Across Services** setting within a persistence profile, the BIG-IP® system attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node only when the virtual server hosting the connection has the same virtual address as the virtual server hosting the initial persistent connection. Connection requests from the client that go to other virtual servers with different virtual addresses, or those connection requests that do not use persistence, are load balanced according to the load balancing method defined for the pool.

For example, suppose you configure virtual server mappings where the virtual server `v1:http` has persistence enabled and references the `http_pool` (containing the nodes `n1:http` and `n2:http`), and the virtual server `v1:ssl` has persistence enabled and references the pool `ssl_pool` (containing the nodes `n1:ssl` and `n2:ssl`).

Suppose the client makes an initial connection to `v1:http`, and the load balancing algorithm assigned to the pool `http_pool` chooses `n1:http` as the node. If the client subsequently connects to `v1:ssl`, the BIG-IP® system uses the persistence session established with the first connection to determine the pool member that should receive the connection request, rather than the load balancing method. the BIG-IP system should then send the third connection request to `n1:ssl`, which uses the same node as the `n1:http` node that currently hosts the client's first connection with which it shares a persistent session.

If the same client then connects to a virtual server with a different virtual address (for example, `v2:ssl`), the BIG-IP system starts tracking a new persistence session, using the load balancing method to determine which node should receive the connection request. The system starts a new persistence session because the requested virtual server uses a different virtual address (v2) than the virtual server hosting the first persistent connection request (v1).

Important: *For the **Match Across Services** setting to be effective, virtual servers that use the same virtual address, as well as those that use SSL persistence, should include the same node addresses in the virtual server mappings.*

Note: *With respect to Cookie profiles, this setting applies to the Cookie Hash method only.*

The Match Across Virtual Servers setting

You can set the BIG-IP® system to maintain persistence for all sessions requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client. When you enable the **Match Across Virtual Servers** setting within a persistence profile, the system attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node. Connection requests from the client that do not use persistence are load balanced according to the currently selected load balancing method.

Note: *With respect to Cookie profiles, this setting applies to the Cookie Hash method only.*

Warning: *For this setting to be effective, virtual servers that use pools with TCP or SSL persistence should include the same member addresses in the virtual server mappings.*

The Match Across Pools setting

When you enable the **Match Across Pools** setting within a persistence profile, the BIG-IP® system can use any pool that contains a given persistence record. The default is disabled (cleared).

Warning: *Enabling this setting can cause the BIG-IP system to direct client traffic to a pool other than that specified by the virtual server.*

Note: *With respect to Cookie profiles, this setting applies to the Cookie Hash method only.*

Protocol Profiles

About protocol profiles

Some of the BIG-IP® system profiles that you can configure are known as protocol profiles. The protocol profiles types are:

- Fast L4
- Fast HTTP
- UDP
- SCTP

For each protocol profile type, the BIG-IP system provides a pre-configured profile with default settings. In most cases, you can use these default profiles as is. If you want to change these settings, you can configure protocol profile settings when you create a profile, or after profile creation by modifying the profile's settings.

To configure and manage protocol profiles, log in to the BIG-IP Configuration utility, and on the Main tab, expand **Local Traffic**, and click **Profiles**.

The Fast L4 profile type

The purpose of a Fast L4 profile is to help you manage Layer 4 traffic more efficiently. When you assign a Fast L4 profile to a virtual server, the Packet Velocity® ASIC (PVA) hardware acceleration within the BIG-IP® system (if supported) can process some or all of the Layer 4 traffic passing through the system. By offloading Layer 4 processing to the PVA hardware acceleration, the BIG-IP system can increase performance and throughput for basic routing functions (Layer 4) and application switching (Layer 7).

You can use a Fast L4 profile with these types of virtual servers: Performance (Layer 4), Forwarding (Layer 2), and Forwarding (IP).

PVA hardware acceleration

When you implement a Fast L4 profile, you can instruct the system to dynamically offload flows in a connection to ePVA hardware, if your BIG-IP system supports such hardware. When you enable the **PVA Offload Dynamic** setting in a Fast L4 profile, you can then configure these values:

- The number of client packets before dynamic ePVA hardware re-offloading occurs. The valid range is from **0** (zero) through **10**. The default is **1**.
- The number of server packets before dynamic ePVA hardware re-offloading occurs. The valid range is from **0** (zero) through **10**. The default is **0**.

The Server Sack, Server Timestamp, and Receive Window settings

The table shown describes three of the Fast L4 profle settings -- Server Sack, Server Timestamp, and Receive Window.

Setting	Description
Server Sack	Specifies whether the BIG-IP system processes Selective ACK (Sack) packets in cookie responses from the server. The default is disabled.
Server Timestamp	Specifies whether the BIG-IP system processes timestamp request packets in cookie responses from the server. The default is disabled.
Receive Window	Specifies the amount of data the BIG-IP system can accept without acknowledging the server. The default value is `0` (zero).

The Fast HTTP profile type

The Fast HTTP profile is a configuration tool designed to speed up certain types of HTTP connections. This profile combines selected features from the TCP Express, HTTP, and OneConnect™ profiles into a single profile that is optimized for the best possible network performance. When you associate this profile with a virtual server, the virtual server processes traffic packet-by-packet, and at a significantly higher speed.

You might consider using a Fast HTTP profile when:

- You do not need features such as remote server authentication, SSL traffic management, and TCP optimizations, nor HTTP features such as data compression, pipelining, and RAM Cache.
- You do not need to maintain source IP addresses.
- You want to reduce the number of connections that are opened to the destination servers.
- The destination servers support connection persistence, that is, HTTP/1.1, or HTTP/1.0 with `Keep-Alive` headers. Note that IIS servers support connection persistence by default.
- You need basic iRule support only (such as limited Layer 4 support and limited HTTP header operations). For example, you can use the iRule events `CLIENT_ACCEPTED`, `SERVER_CONNECTED`, and `HTTP_REQUEST`.

A significant benefit of using a Fast HTTP profile is the way in which the profile supports connection persistence. Using a Fast HTTP profile ensures that for client requests, the BIG-IP® system can transform or add an HTTP `Connection` header to keep connections open. Using the profile also ensures that the BIG-IP system pools any open server-side connections. This support for connection persistence can greatly reduce the load on destination servers by removing much of the overhead caused by the opening and closing of connections.

Note: *The Fast HTTP profile is incompatible with all other profile types. Also, you cannot use this profile type in conjunction with VLAN groups, or with the IPv6 address format.*

When writing iRules®, you can specify a number of events and commands that the Fast HTTP profile supports.

You can use the default `fasthttp` profile as is, or create a custom Fast HTTP profile.

About TCP profiles

TCP profiles are configuration tools that help you to manage TCP network traffic. Many of the configuration settings of TCP profiles are standard SYSCTL types of settings, while others are unique to the BIG-IP® system.

TCP profiles are important because they are required for implementing certain types of other profiles. For example, by implementing TCP, HTTP, Rewrite, HTML, and OneConnect™ profiles, along with a persistence profile, you can take advantage of various traffic management features, such as:

- Content spooling, to reduce server load
- OneConnect, to pool idle server-side connections
- Layer 7 session persistence, such as hash or cookie persistence
- iRules® for managing HTTP traffic
- HTTP data compression
- HTTP pipelining
- URI translation
- HTML content modification
- Rewriting of HTTP redirections

The BIG-IP® system includes several pre-configured TCP profiles that you can use as is. In addition to the default `tcp` profile, the system includes TCP profiles that are pre-configured to optimize LAN and WAN traffic, as well as traffic for mobile users. You can use the pre-configured profiles as is, or you can create a custom profile based on a pre-configured profile and then adjust the values of the settings in the profiles to best suit your particular network environment.

About tcp-lan-optimized profile settings

The `tcp-lan-optimized` profile is a pre-configured profile type that can be associated with a virtual server. In cases where the BIG-IP virtual server is load balancing LAN-based or interactive traffic, you can enhance the performance of your local-area TCP traffic by using the `tcp-lan-optimized` profile.

If the traffic profile is strictly LAN-based, or highly interactive, and a standard virtual server with a TCP profile is required, you can configure your virtual server to use the `tcp-lan-optimized` profile to enhance LAN-based or interactive traffic. For example, applications producing an interactive TCP data flow, such as SSH and TELNET, normally generate a TCP packet for each keystroke. A TCP profile setting such as **Slow Start** can introduce latency when this type of traffic is being processed. By configuring your virtual server to use the **tcp-lan-optimized** profile, you can ensure that the BIG-IP system delivers LAN-based or interactive traffic without delay.

A `tcp-lan-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for LAN-based traffic.

You can use the `tcp-lan-optimized` profile as is, or you can create another custom profile, specifying the `tcp-lan-optimized` profile as the parent profile.

About tcp-wan-optimized profile settings

The `tcp-wan-optimized` profile is a pre-configured profile type. In cases where the BIG-IP system is load balancing traffic over a WAN link, you can enhance the performance of your wide-area TCP traffic by using the `tcp-wan-optimized` profile.

If the traffic profile is strictly WAN-based, and a standard virtual server with a TCP profile is required, you can configure your virtual server to use a `tcp-wan-optimized` profile to enhance WAN-based traffic. For example, in many cases, the client connects to the BIG-IP virtual server over a WAN link, which is generally slower than the connection between the BIG-IP system and the pool member servers. By configuring your virtual server to use `the tcp-wan-optimized` profile, the BIG-IP system can accept the data more quickly, allowing resources on the pool member servers to remain available. Also, use of this profile can increase the amount of data that the BIG-IP system buffers while waiting for a remote client to accept that data. Finally, you can increase network throughput by reducing the number of short TCP segments that the BIG-IP® system sends on the network.

A `tcp-wan-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for WAN-based traffic.

You can use the `tcp-wan-optimized` profile as is, or you can create another custom profile, specifying the `tcp-wan-optimized` profile as the parent profile.

About tcp-mobile-optimized profile settings

The `tcp-mobile-optimized` profile is a pre-configured profile type, for which the default values are set to give better performance to service providers' 3G and 4G customers. Specific options in the pre-configured profile are set to optimize traffic for most mobile users, and you can tune these settings to fit your network. For files that are smaller than 1 MB, this profile is generally better than the `mptcp-mobile-optimized` profile. For a more conservative profile, you can start with the `tcp-mobile-optimized` profile, and adjust from there.

Note: *Although the pre-configured settings produced the best results in the test lab, network conditions are extremely variable. For the best results, start with the default settings and then experiment to find out what works best in your network.*

This list provides guidance for relevant settings

- Set the **Proxy Buffer Low** to the **Proxy Buffer High** value minus 64 KB. If the **Proxy Buffer High** is set to less than 64K, set this value at 32K.
- The size of the **Send Buffer** ranges from 64K to 350K, depending on network characteristics. If you enable the **Rate Pace** setting, the send buffer can handle over 128K, because rate pacing eliminates some of the burstiness that would otherwise exist. On a network with higher packet loss, smaller buffer sizes perform better than larger. The number of loss recoveries indicates whether this setting should be tuned higher or lower. Higher loss recoveries reduce the goodput.
- Setting the **Keep Alive Interval** depends on your fast dormancy goals. The default setting of `1800` seconds allows the phone to enter low power mode while keeping the flow alive on intermediary devices. To prevent the device from entering an idle state, lower this value to under 30 seconds.
- The **Congestion Control** setting includes delay-based and hybrid algorithms, which might better address TCP performance issues better than fully loss-based congestion control algorithms in mobile environments. The Illinois algorithm is more aggressive, and can perform better in some situations, particularly when object sizes are small. When objects are greater than 1 MB, goodput might decrease with Illinois. In a high loss network, Illinois produces lower goodput and higher retransmissions.
- For 4G LTE networks, specify the **Packet Loss Ignore Rate** as `0`. For 3G networks, specify `2500`. When the **Packet Loss Ignore Rate** is specified as more than `0`, the number of retransmitted bytes and receives SACKs might increase dramatically.
- For the **Packet Loss Ignore Burst** setting, specify within the range of `6-12`, if the **Packet Loss Ignore Rate** is set to a value greater than `0`. A higher **Packet Loss Ignore Burst** value increases the chance of unnecessary retransmissions.
- For the **Initial Congestion Window Size** setting, round trips can be reduced when you increase the initial congestion window from `0` to `10` or `16`.

- Enabling the **Rate Pace** setting can result in improved goodput. It reduces loss recovery across all congestion algorithms, except Illinois. The aggressive nature of Illinois results in multiple loss recoveries, even with rate pacing enabled.

A `tcp-mobile-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for mobile traffic.

You can use the `tcp-mobile-optimized` profile as is, or you can create another custom profile, specifying the `tcp-mobile-optimized` profile as the parent profile.

About mptcp-mobile-optimized profile settings

The `mptcp-mobile-optimized` profile is a pre-configured profile type for use in reverse proxy and enterprise environments for mobile applications that are front-ended by a BIG-IP® system. This profile provides a more aggressive starting point than the `tcp-mobile-optimized` profile. It uses newer congestion control algorithms and a newer TCP stack, and is generally better for files that are larger than 1 MB. Specific options in the pre-configured profile are set to optimize traffic for most mobile users in this environment, and you can tune these settings to accommodate your network.

Note: *Although the pre-configured settings produced the best results in the test lab, network conditions are extremely variable. For the best results, start with the default settings and then experiment to find out what works best in your network.*

The enabled **Multipath TCP** (MPTCP) option enables multiple client-side flows to connect to a single server-side flow in a forward proxy scenario. MPTCP automatically and quickly adjusts to congestion in the network, moving traffic away from congested paths and toward uncongested paths.

The **Congestion Control** setting includes delay-based and hybrid algorithms, which can address TCP performance issues better than fully loss-based congestion control algorithms in mobile environments. Refer to the online help descriptions for assistance in selecting the setting that corresponds to your network conditions.

The enabled **Rate Pace** option mitigates bursty behavior in mobile networks and other configurations. It can be useful on high latency or high BDP (bandwidth-delay product) links, where packet drop is likely to be a result of buffer overflow rather than congestion.

An `mptcp-mobile-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for mobile traffic.

You can use the `mptcp-mobile-optimized` profile as is, or you can create another custom profile, specifying the `mptcp-mobile-optimized` profile as the parent profile.

About MPTCP settings

The TCP Profile provides you with multipath TCP (MPTCP) functionality, which eliminates the need to reestablish connections when moving between 3G/4G and WiFi networks. For example, when using MPTCP functionality, if a WiFi connection is dropped, a 4G network can immediately provide the data while the device attempts to resume a WiFi connection, thus preventing a loss of streaming. The TCP profile provides three MPTCP settings: **Enabled**, **Passthrough**, and **Disabled**.

You can use the MPTCP **Enabled** setting when you know all of the available MPTCP flows related to a specific session. The BIG-IP® system manages each flow as an individual TCP flow, while splitting and rejoining flows for the MPTCP session. Note that overall flow optimization, however, cannot be guaranteed; only the optimization for an individual flow is guaranteed.

The MPTCP **Passthrough** setting enables MPTCP header options to pass through, while recognizing that not all corresponding flows to the sessions will be going through the BIG-IP system. This passthrough functionality is especially beneficial when you want to respect the MPTCP header options, but recognize that not all corresponding flows to the session will be flowing through the BIG-IP system. In Passthrough mode, the BIG-IP system allows MPTCP options to pass through, while managing the flow as a FastL4 flow. The MPTCP **Passthrough** setting redirects flows that come into a Layer 7 virtual server to a Fast L4 proxy server. This configuration enables flows to be added or dropped, as necessary, as the user's coverage changes, without interrupting the TCP connection. If a Fast L4 proxy server fails to match, then the flow is blocked.

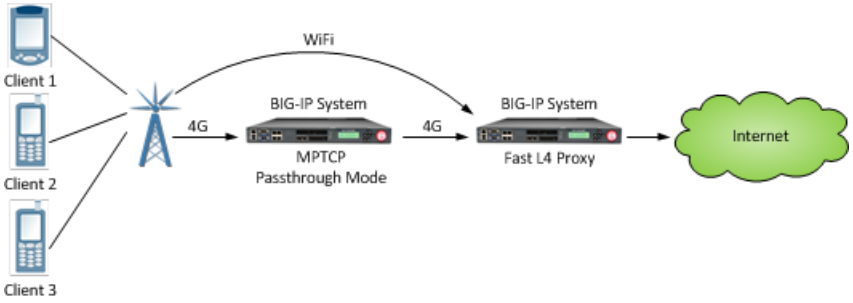

Figure 5: An MPTCP passthrough configuration

When you do not need to support MPTCP header options, you can select the MPTCP **Disabled** setting, so that the BIG-IP system ignores all MPTCP options and simply manages all flows as TCP flows.

The UDP profile type

The UDP profile is a configuration tool for managing UDP network traffic.

Because the BIG-IP® system supports the OpenSSL implementation of datagram Transport Layer Security (TLS), you can optionally assign both a UDP and a Client SSL profile to certain types of virtual servers.

The SCTP profile type

The BIG-IP® system includes a profile type that you can use to manage Stream Control Transmission Protocol (SCTP) traffic. *Stream Control Transmission Protocol (SCTP)* is a general-purpose, industry-standard transport protocol, designed for message-oriented applications that transport signalling data. The design of SCTP includes appropriate congestion-avoidance behavior, as well as resistance to flooding and masquerade attacks.

Unlike TCP, SCTP includes the ability to support *multistreaming functionality*, which permits several streams within an SCTP connection. While a *TCP stream* refers to a sequence of bytes, an *SCTP stream* represents a sequence of data messages. Each data message (or chunk) contains an integer ID that identifies a stream, an application-defined Payload Protocol Identifier (PPI), a Stream sequence number, and a Transmit Serial Number (TSN) that uniquely identifies the chunk within the SCTP connection. Chunk delivery is acknowledged using TSNs sent in selective acknowledgements (ACKs) so that every chunk can be independently acknowledged. This capability demonstrates a significant benefit of streams, because it eliminates head-of-line blocking within the connection. A lost chunk of data on one stream does not prevent other streams from progressing while that lost chunk is retransmitted.

SCTP also includes the ability to support *multihoming functionality*, which provides path redundancy for an SCTP connection by enabling SCTP to send packets between multiple addresses owned by each endpoint. SCTP endpoints typically configure different IP addresses on different network interfaces to provide redundant physical paths between the peers. For example, a client and server might be attached to separate VLANs. The client and server can each advertise two IP addresses (one per VLAN) to the other peer. If either VLAN is available, then SCTP can transport packets between the peers.

You can use SCTP as the transport protocol for applications that require monitoring and detection of session loss. For such applications, the SCTP mechanisms to detect session failure actively monitor the connectivity of a session.

The Any IP profile type

With the Any IP profile, you can enforce an idle timeout value on IP traffic other than TCP and UDP traffic. You can use the BIG-IP® Configuration utility to create, view details for, or delete Any IP profiles.

When you configure an idle timeout value, you specify the number of seconds for which a connection is idle before the connection is eligible for deletion. The default value is 60 seconds. Possible values that you can configure are:

Specify
Specifies the number of seconds that the Any IP connection is to remain idle before it can be deleted. When you select **Specify**, you must also type a number in the box.

Immediate
Specifies that you do not want the connection to remain idle, and that it is therefore immediately eligible for deletion.

Indefinite
Specifies that Any IP connections can remain idle indefinitely.

About SSL Profiles

About SSL profiles

When you want the BIG-IP system to process application traffic over SSL, you can configure the system to perform the SSL handshake that destination servers normally perform. This ability for the BIG-IP system to offload SSL processing from a destination server is an important feature of the BIG-IP system.

The most common way to configure the BIG-IP system is to create a Client SSL profile, which makes it possible for the BIG-IP system to decrypt client requests before sending them on to a server, and encrypt server responses before sending them back to the client.

Within a Client SSL profile specifically, you can specify multiple certificate/key pairs, one per key type. This enables the system to accept all types of cipher suites that a client might support as part of creating a secure connection. The system then decrypts the client data, manipulates any headers or payload according to the way that you configured the Client SSL profile, and by default, sends the request in clear text to the target server for processing.

For those sites that require enhanced security on their internal network, you can configure a Server SSL profile. With a Server SSL profile, the BIG-IP system re-encrypts the request before sending it to the destination server. When the server returns an encrypted response, the BIG-IP system decrypts and then re-encrypts the response, before sending the response back to the client.

For more information on configuring SSL and Online Certificate Status Protocol (OCSP) profiles, see the guide *BIG-IP System: SSL Administration* at `http://support.f5.com`.

Authentication Profiles

Introduction to authentication profiles

A significant feature of the BIG-IP® system is its ability to support Pluggable Authentication Module (PAM) technology. *PAM* technology allows you to choose from a number of different authentication and authorization schemes to use to authenticate or authorize network traffic.

The goal of PAM technology is to separate an application, such as the BIG-IP system, from its underlying authentication technology. This means that you can dictate the particular authentication/authorization technology that you want the BIG-IP system to use to authenticate application traffic coming into the BIG-IP system.

To this end, the BIG-IP system offers several authentication schemes, known as authentication modules. These *authentication modules* allow you to use a remote system to authenticate or authorize application requests that pass through the BIG-IP system.

The BIG-IP system normally routes remote authentication traffic through a Traffic Management Microkernel (TMM) switch interface (that is, an interface associated with a VLAN and a self IP address), rather than through the management interface. Therefore, if the TMM service is stopped for any reason, remote authentication is not available until the service is running again.

BIG-IP system authentication modules

The BIG-IP® system authentication modules that you can implement for remote authentication are:

Lightweight Directory Access Protocol (LDAP)
The BIG-IP system can authenticate or authorize network traffic using data stored on a remote LDAP server or a Microsoft® Windows® Active Directory® server. Client credentials are based on basic HTTP authentication (user name and password).

Remote Authentication Dial-In User Service (RADIUS)
The BIG-IP system can authenticate network traffic using data stored on a remote RADIUS server. Client credentials are based on basic HTTP authentication (user name and password).

TACACS+
The BIG-IP system can authenticate network traffic using data stored on a remote TACACS+ server. Client credentials are based on basic HTTP authentication (user name and password).

SSL client certificate LDAP
The BIG-IP system can authorize network traffic using data stored on a remote LDAP server. Client credentials are based on SSL certificates, as well as defined user groups and roles.

Online Certificate Status Protocol (OCSP)
The BIG-IP system can check on the revocation status of a client certificate using data stored on a remote OCSP server. Client credentials are based on SSL certificates.

Certificate Revocation List Distribution Point (CRLDP)
The BIG-IP system can use CRL distribution points to determine revocation status.

Important: *When you create remote authentication objects and profiles, the BIG-IP® system places them into your current administrative partition. Note that the default profile always resides in partition* `Common`.

The LDAP authentication module

An *LDAP authentication module* is a mechanism for authenticating or authorizing client connections passing through a BIG-IP® system. This module is useful when your authentication or authorization data is stored on a remote LDAP server or a Microsoft Windows Active Directory server, and you want the client credentials to be based on basic HTTP authentication (that is, user name and password).

With the LDAP authentication module, the BIG-IP® system can indicate that the authentication was a success or failure, or that the LDAP server needs a credential of some sort.

Additionally, the system can take some action based on certain information that the server returns in the LDAP query response. For example, LDAP response information can indicate the user's group membership, or it can indicate that the user's password has expired. To configure the BIG-IP system to return specific data in an LDAP response, you can write an iRule, using the commands `AUTH::subscribe`, `AUTH::unsubscribe`, and `AUTH::response_data`. For more information, see the F5 Networks DevCentral web site, `http://devcentral.f5.com`.

The RADIUS authentication module

A *RADIUS authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. You use this module when your authentication data is stored on a remote RADIUS server. In this case, client credentials are based on basic HTTP authentication (that is, user name and password).

The TACACS+ authentication module

A *TACACS+ authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. You use this module when your authentication data is stored on a remote TACACS+ server. In this case, client credentials are based on basic HTTP authentication (that is, user name and password).

The SSL client certificate LDAP authentication module

An *SSL client certificate LDAP authentication module* is a mechanism for authorizing client connections passing through a BIG-IP® system. With the SSL client certificate LDAP authentication module, you can use a remote LDAP server to impose access control on application traffic. The module bases this access control on SSL certificates, as well as user groups and roles that you specify.

With the SSL client certificate LDAP authentication module, Local Traffic Manager™ can indicate that the authorization was a success or failure, or that the LDAP server needs a credential of some sort.

Additionally, the system can take some action based on certain information that the server returns in the LDAP query response. For example, LDAP response information can indicate the user's group membership, or it can indicate that the user's password has expired. To configure the BIG-IP system to return specific data in an LDAP response, you can write an iRule, using the commands `AUTH::subscribe`, `AUTH::unsubscribe`, and `AUTH::response_data`. For more information, see the F5 Networks DevCentral web site, `http://devcentral.f5.com`.

Search results and corresponding authorization status

This table shows the results of certificate-based authorization being performed.

Result of search	Authorization status
No records match	Authorization fails
One record matches	Authorization succeeds and is subject to groups and roles
Two or more records match	Authorization fails, due to invalid database entries

SSL client certificate authorization

Before you can implement an SSL client certificate LDAP module, you must understand the two different types of credentials that the BIG-IP® system uses to authorize application traffic using data on a remote LDAP server. These two types of credentials are:

- SSL certificates
- Groups and roles

With SSL client certificate LDAP authorization, the BIG-IP® system can authorize clients based on signed client certificates issued by trusted CAs. Then, to further enhance the ability of the system to authorize client requests, you can also specify groups and roles. Basing authorization on certificates as well as groups and roles provides the flexibility you need to control client access to system resources.

SSL certificates for LDAP authorization

During the process of authorizing a client, the BIG-IP® system must search the LDAP database. When using certificate-based authorization, the system can search the LDAP database in three ways:

User

If certificates are not stored in the LDAP database, you can configure the system to extract a user name from the certificate presented as part of the incoming client request. The system then checks to see if an entry for the user exists in the LDAP database. This scenario is a good choice for a company that acts as its own Certificate Authority, where the company is assured that if the certificate is verified, then the user is authorized.

Certificate Map

If you create an object and class that map certificates to users in the LDAP database, you can then configure the system to search for a certificate in the map, and retrieve a user from that map. The system then checks to ensure that the user in the LDAP database is a valid user.

Certificate

Many LDAP server environments already incorporate certificates into the user information stored in the LDAP database. One way of configuring authorization in LDAP server environments is to configure

the system to compare an incoming certificate to the certificate stored in the LDAP database for the user associated with the client request. If the certificate is found in the user's LDAP profile, access is granted to the user, and the request is granted.

Groups and roles for LDAP authorization

In addition to enabling certificate-based authorization, you can also configure authorization based on groups and roles.

Groups

Because LDAP servers already have the concept and structure of groups built into them, the BIG-IP® system can include groups in its authorization feature. To enable the use of groups for authorization purposes, you must indicate the base and scope under which the system will search for groups in the LDAP database. Also, you must specify values for a group name and a member name. Once you have completed these tasks, the system can search through the list of valid groups until a group is found that has the current user as a member.

Roles

Unlike a group, a role is a setting directly associated with a user. Any role-based authorization that the BIG-IP system performs depends on the LDAP database having the concept of roles built into it. To determine if a user should be granted access to a resource, the BIG-IP system searches through the roles assigned to the user and attempts to match that role to a valid role defined by the administrator.

The SSL OCSP authentication module

An *SSL OCSP authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. More specifically, an SSL *Online Certificate Status Protocol (OCSP)* authentication module checks the revocation status of an SSL certificate, as part of authenticating that certificate.

A single SSL OCSP profile can target a specific responder, or multiple SSL OCSP profiles can target the same responder. Each responder itself is associated with a certificate authority (CA), and multiple responders can be associated with the same CA.

Note: *The BIG-IP system allows you to enable both the CRL and the OCSP options. Most users need to enable either one or the other, but not both. However, in the rare case that you want to enable both options, be aware that both the search of the CRL file, and the connection to the responder must be successful. Otherwise, the BIG-IP system cannot obtain status.*

Note that an OCSP responder object is an object that you create that includes a URL for an external OCSP responder. You must create a separate OCSP responder object for each external OCSP responder. When you subsequently create an OCSP configuration object, the configuration object contains a reference to any OCSP responder objects that you have created.

If you want to use CRLs instead of OCSP, you configure an SSL profile.

What is OCSP?

Online Certificate Status Protocol (OCSP) is a third-party software application and industry-standard protocol that offers an alternative to a certificate revocation list (CRL) when using public-key technology. On the BIG-IP system, OCSP ensures that the BIG-IP system always obtains real-time revocation status

during the certificate verification process. A *CRL* is a list of revoked client certificates, which a server system can check during the process of verifying a client certificate.

OCSP is based on a client/server model, where a client system requests revocation status of a certificate, and a server system sends the response. Thus, when you implement the SSL OCSP authentication module, the BIG-IP system acts as the OCSP client, and an external system, known as an OCSP responder, acts as the OCSP server. An *OCSP responder* is therefore an external server that sends certificate revocation status, upon request, to the BIG-IP system.

For more information on OCSP, see RFC 2560 at URL `http://www.ietf.org`.

Limitations of Certificate Revocation Lists

Using OCSP to check on the revocation status of client certificates offers distinct advantages over the use of a CRL.

When presented with a client certificate, the BIG-IP system sometimes needs to assess the revocation state of that certificate before accepting the certificate and forwarding the connection to a target server. The standard method of assessing revocation status is a CRL, which is stored in a separate CRL file on each machine in your configuration. Although CRLs are considered to be a standard way of checking revocation status of SSL certificates, a CRL is updated only at fixed intervals, thus presenting a risk that the information in the CRL is outdated at the time that the status check occurs.

Also, having to store a separate CRL file on each machine presents other limitations:

- All CRL files must be kept in sync.
- Having a separate CRL file on each machine poses a security risk.
- Multiple CRL files cannot be administered from a central location.

The CRLDP authentication module

A CRLDP authentication module is a mechanism for authenticating client connections passing through a BIG-IP® system. You implement a CRLDP authentication module when you want the BIG-IP system to use CRL distribution points as the mechanism for checking the revocation status of SSL certificates.

This CRLDP authentication feature is based on a technology known as *Certificate Revocation List Distribution Points (CRLDP)*. CRLDP is an industry-standard protocol that offers an alternative method for checking a standard certificate revocation list (CRL) to determine revocation status. CRLDP is also an alternative to using Online Certificate Status Protocol (OCSP).

A CRLDP authentication module uses CRL distribution points to check the revocation status of an SSL certificate, as part of authenticating that certificate. *CRL distribution points* are a mechanism used to distribute certificate revocation information across a network. More specifically, a distribution point is a Uniform Resource Identifier (URI) or directory name specified in a certificate that identifies how the server obtains CRL information. Distribution points can be used in conjunction with CRLs to configure certificate authorization using any number of LDAP servers.

Message Routing Profiles

Overview: Diameter message routing

The Diameter protocol provides message-routing functionality that the BIG-IP® system supports in a load-balancing configuration.

Diameter message routing configuration

In a message routing configuration, the BIG-IP system manages requests and responses among peers. The following illustration shows a Diameter routing configuration with requests from Client 1 and Client 2 to servers located in different destination realms, Realm-A and Realm-B.

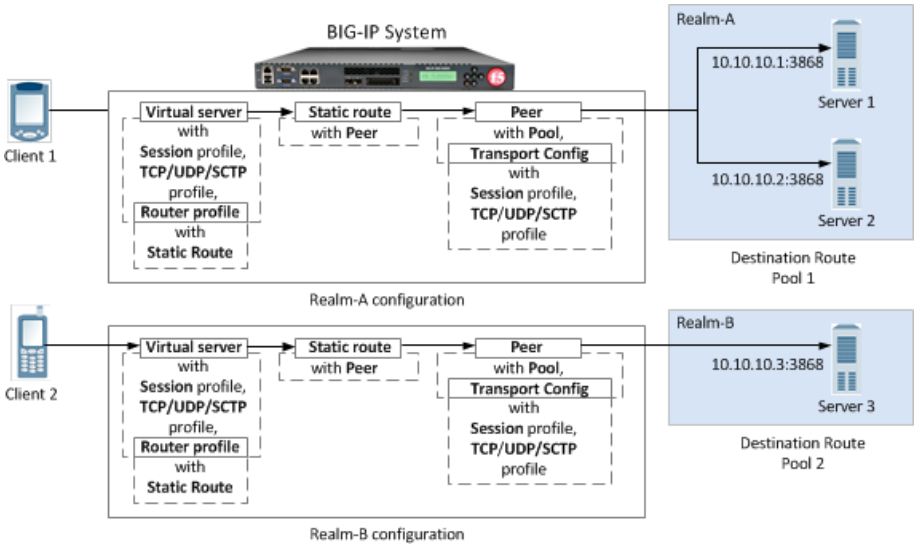

Figure 6: A Diameter message routing configuration

A typical Diameter message routing configuration with two realms involves configuring the following items.

Functionality	Description
Pool	A pool for each realm directs Diameter traffic to servers.
Session profile	A session profile for each realm configures a session as a set of messages between two Diameter nodes on behalf of a user.
Transport configuration	An optional transport configuration for each realm defines how the BIG-IP system connects with the servers on your network when routing messages. You can assign a transport configuration to a virtual server or peer, as needed.
Peer	Each BIG-IP message-routing peer routes messages to a destination host. In this example, BIG-IP message-routing peers route messages to `10.10.10.1:3868`, `10.10.10.2:3868`, and `10.10.10.3:3868`.

Functionality	Description
Static Route	Each static route specifies a set of peers in a destination realm to use in forwarding messages. In this example, `Realm-A` includes `Peer 1` , and `Realm-B` includes `Peer 2`.
Router profile	A router profile configures Diameter message routing parameters and static routes to be used by a virtual server in routing Diameter messages.
Virtual server	Manages Diameter traffic to and from each realm and pool members.

About the Diameter session profile

The Diameter *session profile* includes Diameter protocol parameters that can be used by a virtual server or transport configuration in managing Diameter traffic. The profile enables you to configure the properties of a Diameter session as a set of messages between two diameter nodes on behalf of a user. Note that those same two diameter nodes can also include multiple active user sessions. The session profile provides you with parameters to configure settings for timeout, watchdog failures, and message-size, as well as persistence, rewrite, and capabilities-handshake functionality.

Functionality	Description
Settings	Configure timeout functionality, watchdog failures, and message size.
Persistence	Configure persistence functionality, including a type, AVP, and timeout.
Rewrite	Provide AVP rewriting to conceal clients from servers, as well as to conceal servers from clients.
Capabilities Handshake	When the Diameter session profile is configured as a proxy, the BIG-IP system generates capabilities-exchange messages, sending a Capabilities-Exchange-Request (CER) and responding with a Capabilities-Exchange-Answer (CEA), to establish a diameter session with connected nodes.

You can apply different session profiles to different transport configurations, and then apply the different transport configurations to different message routing peers, which point to different physical pools. You can also apply different session profiles by applying one session profile to the transport configuration, and a different session profile to the virtual server.

About message routing Diameter transport configuration

Message routing Diameter *Transport Config* functionality defines how the BIG-IP® system connects with the servers on your network when routing and load balancing messages. With the transport configuration settings, you can assign a TCP, UDP, or SCTP profile, and Diameter session profile, as well as iRules®, a source port, and source address translation.

About message routing peers

A message routing *peer* defines how the BIG-IP® system routes messages to destination hosts. When you configure a message routing peer, you define a pool of destination hosts, and a connection method for them, an optional transport configuration configured with a Diameter session profile, as needed, the number of connections to a destination host, and a ratio value for selection of a peer. After defining the peers, you can use those peers in configuring static routes.

If a peer does not specify a pool, the BIG-IP system uses the destination IP address and port of the ingress message's connection. If a peer specifies a pool without pool members, the message is unroutable.

If a peer does not specify a transport configuration, the BIG-IP system uses the transport type of the message's originating connection.

About Diameter peer selection

When you configure a Diameter static route, the BIG-IP® system provides two modes for peer selection: sequential and ratio.

In *sequential mode*, the BIG-IP system uses peers in the order specified by the Peers Selected list. If a message is retried, the next peer in the Peers Selected list is used.

In *ratio mode*, the BIG-IP system uses peers in accordance with the peer's ratio value, which you specify when configuring each peer. The relative ratio value for each peer determines whether a peer is selected from the list. For example, a peer with a ratio value of 1 is typically selected over a peer with a ratio value of 2. The lower the ratio value, the greater the probability for selection.

Before configuring a mode for peer selection, you must first configure each peer, using the Peer tab, to include peers in the Available list.

About static routes

The message routing functionality *Static Routes* enables you to configure a route that specifies a set of peers to use in forwarding messages. When you configure a static route, you can specify an application ID, destination realm, origin realm, virtual server, peer selection mode, and peers.

The required static route attributes (each of which must match the respective request parameter) are prioritized in this order:

1. Destination Realm
2. Application Id
3. Origin Realm
4. Virtual Server

A static route is a *default route* when each of these attributes is set to the default (wildcard) value.

About the Diameter router profile

With the Diameter router profile, you can configure Diameter routing parameters to be used by a virtual server in routing Diameter messages. When you configure a Diameter router profile, you can specify persistence, rewrite, and capabilities-handshake functionality.

Diameter AVP names

This list specifies supported Diameter Attribute-Value Pair (AVP) names.

AVP Names

- `ACCOUNTING-REALTIME-REQUIRED`
- `ACCOUNTING-RECORD-NUMBER`

- ACCOUNTING-RECORD-TYPE
- ACCOUNTING-SUB-SESSION-ID
- ACCT-APPLICATION-ID
- ACCT-INTERIM-INTERVAL
- ACCT-MULTI-SESSION-ID
- ACCT-SESSION-ID
- AUTH-APPLICATION-ID
- AUTH-GRACE-PERIOD
- AUTH-REQUEST-TYPE
- AUTH-SESSION-STATE
- AUTHORIZATION-LIFETIME
- CALLING-STATION-ID
- CLASS
- DESTINATION-HOST
- DESTINATION-REALM
- DISCONNECT-CAUSE
- E2E-SEQUENCE
- ERROR-MESSAGE
- ERROR-REPORTING-HOST
- EVENT-TIMESTAMP
- EXPERIMENTAL-RESULT
- EXPERIMENTAL-RESULT-CODE
- FAILED-AVP
- FIRMWARE-REVISION
- FRAMED-IP-ADDRESS
- HOST-IP-ADDRESS
- INBAND-SECURITY-ID
- MULTI-ROUND-TIME-OUT
- ORIGIN-HOST
- ORIGIN-REALM
- ORIGIN-STATE-ID
- PRODUCT-NAME
- PROXY-HOST
- PROXY-INFO
- PROXY-STATE
- RE-AUTH-REQUEST-TYPE
- REDIRECT-HOST
- REDIRECT-HOST-USAGE
- REDIRECT-MAX-CACHE-TIME
- RESULT-CODE
- ROUTE-RECORD
- SESSION-BINDING
- SESSION-ID
- SESSION-SERVER-FAILOVER
- SESSION-TIMEOUT
- SUBSCRIPTION-ID
- SUBSCRIPTION-ID-DATA
- SUBSCRIPTION-ID-TYPE

- SUPPORTED-VENDOR-ID
- TERMINATION-CAUSE
- USER-EQUIPMENT-INFO
- USER-EQUIPMENT-TYPE
- USER-EQUIPMENT-VALUE
- USER-NAME
- VENDOR-ID
- VENDOR-SPECIFIC-APPLICATION-ID

Overview: Configuring a SIP proxy

You can use the BIG-IP® system as a Session Initiation Protocol (SIP) proxy. When the BIG-IP system is placed between your SIP routers, session border controllers, and soft switches, you can configure the system to route and load balance SIP messages across the servers on your SIP network.

This graphic illustrates the relationships of the configuration objects that you must configure on the BIG-IP system.

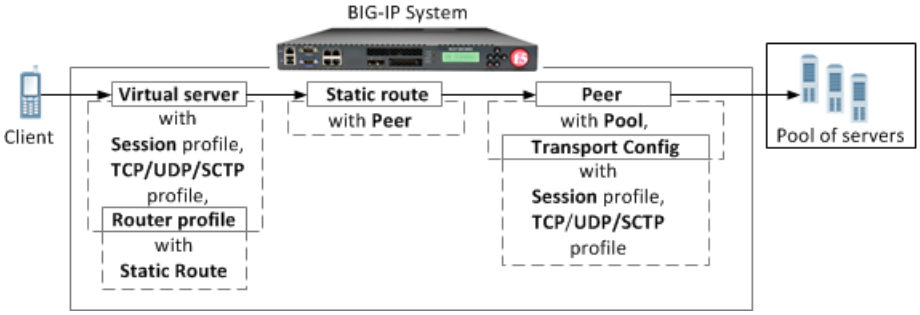

Figure 7: SIP proxy configuration objects

Task summary

About managing MRF SIP session traffic

Through the SIP Session Profile, you can use Message Routing Framework (MRF) to manage SIP traffic across pool members by means of configuring and using Via headers. When you configure Via headers to manage SIP traffic, dependencies between settings apply, enabling you to steer traffic and control requests and responses, as necessary.

Note: *When a client Via header only specifies an address, without specifying a port, the BIG-IP® system uses default port* `5060`*. For example, if a client sends a request with Via header* `SIP/2.0/TCP 192.168.20.1`*, in SIP session traffic scenario 1 (default), the BIG-IP system sends a response to the client with Via header* `SIP/2.0/TCP 192.168.20.1/5060`*.*

Example: SIP session traffic scenario 1 (default)

In SIP session traffic scenario 1 (default), the BIG-IP system receives a request with a Via1 header from a client, and inserts a Via2 header into the request before forwarding the request to the server. When the server provides a response, the BIG-IP system removes the Via2 header from the response, before forwarding the response to the client. If the originating connection no longer exists, the Via2 header that BIG-IP system

inserted is no longer available; consequently, the BIG-IP system uses the Via1 header, forwarding the message to the client IP address and port specified by that Via header.

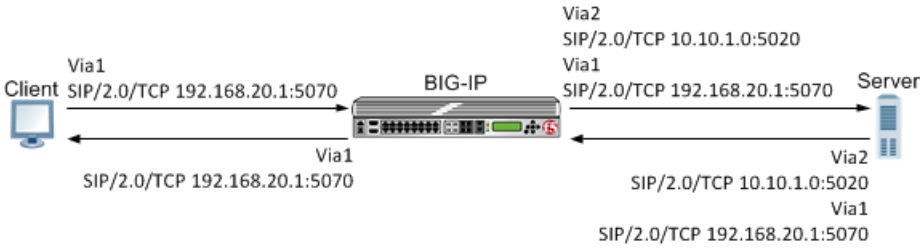

Figure 8: An example of SIP session traffic scenario 1 (default)

When configuring this scenario, the following SIP Session Profile settings apply.

SIP Session Profile control	Setting or value
Honor Via	**Enabled**
Do Not Connect Back	**Disabled**
Insert Via Header	**Enabled**
Custom Via	Not applicable

Example: SIP session traffic scenario 2

In SIP session traffic scenario 2, the BIG-IP system receives a request with a Via1 header from a client, and inserts a Via2 header into the request before forwarding the request to the server. When the server provides a response, the BIG-IP system removes the Via2 header from the response, before forwarding the response to the client. When the originating connection no longer exists, then the BIG-IP system drops the response message and increments the statistic for **Messages failed due to connection dropped**.

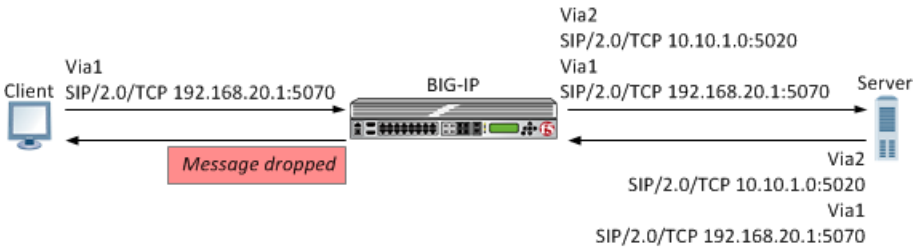

Figure 9: An example of SIP session traffic scenario 2

When configuring this scenario, the following SIP Session Profile settings apply.

SIP Session Profile control	Setting or value
Honor Via	**Enabled**
Do Not Connect Back	**Enabled**
Insert Via Header	**Enabled**
Custom Via	Not applicable

Example: SIP session traffic scenario 3

In SIP session traffic scenario 3, the BIG-IP system receives a request with a Via1 header from a client, and inserts a Via2 header into the request before forwarding the request to the server. When the server

provides a response, the BIG-IP system removes the Via2 header from the response, before forwarding the response to the client. If the originating connection no longer exists, then the Via header that BIG-IP system inserted is no longer available; consequently, the BIG-IP system uses the next available Via header, but, because the **Honor Via** setting is **Disabled**, the BIG-IP system does not forward the message to the client IP address and port specified by that Via header.

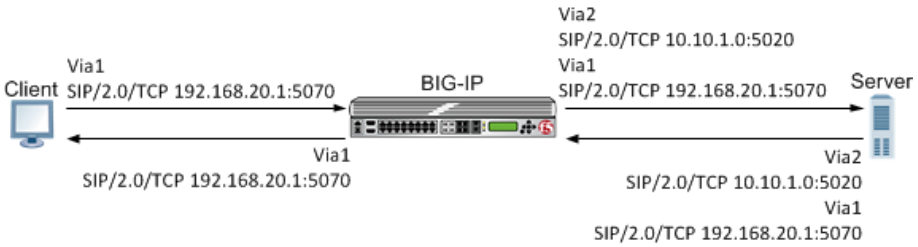

Figure 10: An example of SIP session traffic scenario 3

When configuring this scenario, the following SIP Session Profile settings apply.

SIP Session Profile control	Setting or value
Honor Via	**Disabled**
Do Not Connect Back	**Disabled**
Insert Via Header	**Enabled**
Custom Via	Not applicable

Example: SIP session traffic scenario 4

In SIP session traffic scenario 4, the BIG-IP system receives a request with a Via1 header from a client, and inserts a Via2 header into the request before forwarding the request to the server. When the server provides a response, the response from the BIG-IP to the client must be managed by means of an iRule, for example, `MR::message nexthop TMM:flow_id` or `MR::message route virtual vs_name host ip:port`.

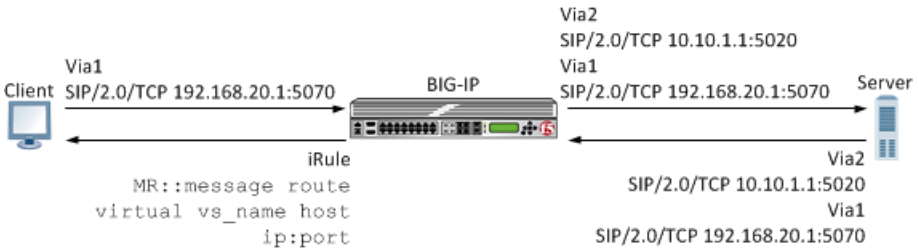

Figure 11: An example of SIP session traffic scenario 4

When configuring this scenario, the following SIP Session Profile settings apply.

SIP Session Profile control	Setting or value
Honor Via	Not applicable
Do Not Connect Back	Not applicable
Insert Via Header	**Enabled**
Custom Via	Custom Via header value, for example: `SIP/2.0/TCP www.siterequest.com:4343` or `SIP/2.0/SCTP 10.10.4.32`

Example: SIP session traffic scenario 5

In SIP session traffic scenario 5, the BIG-IP system receives a request with a Via1 header from a client, but does not insert a Via header into the request before forwarding the request to the server. When the server provides a response, the BIG-IP system uses the client Via1 header in the response to forward the message to the client IP address and port specified by that Via header.

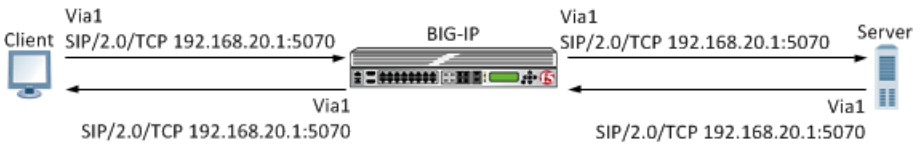

Figure 12: An example of SIP session traffic scenario 5

When configuring this scenario, the following SIP Session Profile settings apply.

SIP Session Profile control	Setting or value
Honor Via	**Enabled**
Do Not Connect Back	Not applicable
Insert Via Header	**Disabled**
Custom Via	Not applicable

Example: SIP session traffic scenario 6

In SIP session traffic scenario 6, the BIG-IP system receives a request with a Via1 header from a client, but does not insert a Via header into the request before forwarding the request to the server. Instead, the BIG-IP system uses the Via1 header specified in the request. When the server provides a response, the BIG-IP system uses the Via1 header in the response, but does not forward the message to the client IP address and port specified by that Via header.

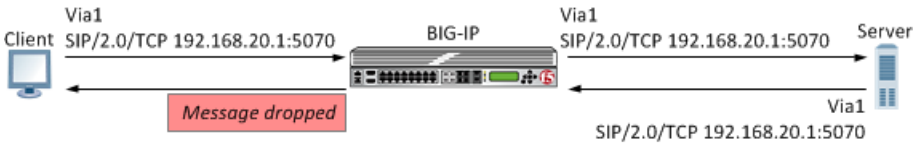

Figure 13: An example of SIP session traffic scenario 6

When configuring this scenario, the following SIP Session Profile settings apply.

SIP Session Profile control	Setting or value
Honor Via	**Disabled**
Do Not Connect Back	Not applicable
Insert Via Header	**Disabled**
Custom Via	Not applicable

Overview: Configuring a SIP message routing firewall

You can use the BIG-IP® system Session Initiation Protocol (SIP) message routing functionality in a firewall configuration to provide stateful handling of SIP communication and media flows. A virtual server handles

the SIP communications and related media flows, allowing them to pass through otherwise restrictive firewall rules. You configure a Local Traffic message routing SIP profile, router profile, and virtual server, and then use that configuration with an Advanced Firewall Manager™ (AFM™) DoS profile. In this firewall configuration, the SIP session profile, SIP router profile, and virtual server use Application Level Gateway (ALG) functionality, where the BIG-IP system does not perform address translation or subscriber registration tracking.

Note: *When using ALG functionality, you cannot use a SIP router profile with an operation mode that is configured to use load balancing settings. Instead, you need to use a SIP router profile with the operation mode configured to use Application Level Gateway settings.*

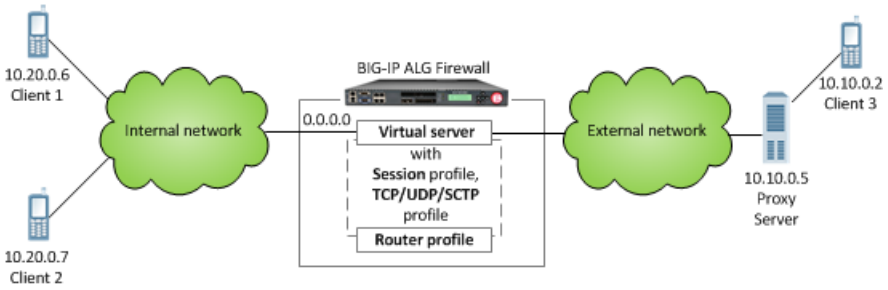

Figure 14: A SIP firewall configuration

Other Profiles

Introduction to other profiles

In addition to the profiles described in previous chapters, you can configure these BIG-IP® system profiles:

- OneConnect™
- NTLM
- Statistics
- Stream

For each profile type, the BIG-IP system provides a pre-configured profile with default settings. In most cases, you can use these default profiles as is. If you want to change these settings, you can configure profile settings when you create a profile, or after profile creation by modifying the profile's settings.

About OneConnect profiles

The OneConnect profile type implements the BIG-IP® system's OneConnect feature. This feature can increase network throughput by efficiently managing connections created between the BIG-IP system and back-end pool members. You can use the OneConnect feature with any TCP-based protocol, such as HTTP or RTSP.

How does OneConnect work?

The *OneConnect* feature works with request headers to keep existing server-side connections open and available for reuse by other clients. When a client makes a new connection to a virtual server configured with a OneConnect profile, the BIG-IP system parses the request, selects a server using the load-balancing method defined in the pool, and creates a connection to that server. When the client's initial request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse.

When another connection is subsequently initiated to the virtual server, if an existing server-side flow to the pool member is open and idle, the BIG-IP system applies the OneConnect source mask to the IP address in the request to determine whether the request is eligible to reuse the existing idle connection. If the request is eligible, the BIG-IP system marks the connection as non-idle and sends a client request over that connection. If the request is not eligible for reuse, or an idle server-side flow is not found, the BIG-IP system creates a new server-side TCP connection and sends client requests over the new connection.

Note: *The BIG-IP system can pool server-side connections from multiple virtual servers if those virtual servers reference the same OneConnect profile and the same pool. Also, the re-use of idle connections can cause the BIG-IP system to appear as though the system is not load balancing traffic evenly across pool members.*

About client source IP addresses

The standard address translation mechanism on the BIG-IP system translates only the destination IP address in a request and not the source IP address (that is, the client node's IP address). However, when the

OneConnect feature is enabled, allowing multiple client nodes to re-use a server-side connection, the source IP address in the header of each client node's request is always the IP address of the client node that initially opened the server-side connection. Although this does not affect traffic flow, you might see evidence of this when viewing certain types of system output.

The OneConnect profile settings

When configuring a OneConnect profile, you specify this information:

Source mask
The mask applied to the source IP address to determine the connection's eligibility to reuse a server-side connection.

Maximum size of idle connections
The maximum number of idle server-side connections kept in the connection pool.

Maximum age before deletion from the pool
The maximum number of seconds that a server-side connection is allowed to remain before the connection is deleted from the connection pool.

Maximum reuse of a connection
The maximum number of requests to be sent over a server-side connection. This number should be slightly lower than the maximum number of HTTP `Keep-Alive` requests accepted by servers in order to prevent the server from initiating a connection close action and entering the `TIME_WAIT` state.

Idle timeout override
The maximum time that idle server-side connections are kept open. Lowering this value may result in a lower number of idle server-side connections, but may increase request latency and server-side connection rate.

OneConnect and HTTP profiles

Content switching for HTTP requests

When you assign both a OneConnect profile and an HTTP profile to a virtual server, and an HTTP client sends multiple requests within a single connection, the BIG-IP system can process each HTTP request individually. The BIG-IP system sends the HTTP requests to different destination servers as determined by the load balancing method. Without a OneConnect profile enabled for the HTTP virtual server, the BIG-IP system performs load-balancing only once for each TCP connection.

HTTP version considerations

For HTTP traffic to be eligible to use the OneConnect feature, the web server must support HTTP `Keep-Alive` connections. The version of the HTTP protocol you are using determines to what extent this support is available. The BIG-IP system therefore includes a *OneConnect transformations* feature within the HTTP profile, specifically designed for use with HTTP/1.0 which by default does not enable `Keep-Alive` connections. With the OneConnect transformations feature, the BIG-IP system can transform HTTP/1.0 connections into HTTP/1.1 requests on the server side, thus allowing those connections to remain open for reuse.

The two different versions of the HTTP protocol treat `Keep-Alive` connections in these ways:

HTTP/1.1 requests
HTTP `Keep-Alive` connections are enabled by default in HTTP/1.1. With HTTP/1.1 requests, the server does not close the connection when the content transfer is complete, unless the client sends a `Connection: close` header in the request. Instead, the connection remains active in anticipation of

the client reusing the same connection to send additional requests. For HTTP/1.1 requests, you do not need to use the OneConnect transformations feature.

HTTP/1.0 requests

HTTP `Keep-Alive` connections are not enabled by default in HTTP/1.0. With HTTP/1.0 requests, the client typically sends a `Connection: close` header to close the TCP connection after sending the request. Both the server and client-side connections that contain the `Connection: close header` are closed once the response is sent. When you assign a OneConnect profile to a virtual server, the BIG-IP system transforms `Connection: close` headers in HTTP/1.0 client-side requests to `X-Cnection: close` headers on the server side, thereby allowing a client to reuse an existing connection to send additional requests.

OneConnect and NTLM profiles

NT Lan Manager (NTLM) `HTTP 401` responses prevent the BIG-IP® system from detaching the server-side connection. As a result, a late FIN from a previous client connection might be forwarded to a new client that re-used the connection, causing the client-side connection to close before the NTLM handshake completes. If you prefer NTLM authentication support when using the OneConnect feature, you should configure an NTLM profile in addition to the OneConnect profile.

OneConnect and SNATs

When a client makes a new connection to a virtual server that is configured with a OneConnect profile and a source network address translation (SNAT) object, the BIG-IP system parses the HTTP request, selects a server using the load-balancing method defined in the pool, translates the source IP address in the request to the SNAT IP address, and creates a connection to the server. When the client's initial HTTP request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse. When a new connection is initiated to the virtual server, the BIG-IP system performs SNAT address translation on the source IP address and then applies the OneConnect source mask to the translated SNAT IP address to determine whether it is eligible to reuse an existing idle connection.

About NTLM profiles

NT LAN Manager (NTLM) is an industry-standard technology that uses an encrypted challenge/response protocol to authenticate a user without sending the user's password over the network. Instead, the system requesting authentication performs a calculation to prove that the system has access to the secured NTLM credentials. NTLM credentials are based on data such as the domain name and user name, obtained during the interactive login process.

The NTLM profile within the BIG-IP® system optimizes network performance when the system is processing NT LAN Manager traffic. When both an NTLM profile and a OneConnect™ profile are associated with a virtual server, the local traffic management system can take advantage of server-side connection pooling for NTLM connections.

How does the NTLM profile work?

When the NTLM profile is associated with a virtual server and the server replies with the HTTP 401 Unauthorized HTTP response message, the NTLM profile inserts a cookie, along with additional profile options, into the HTTP response. The information is encrypted with a user-supplied passphrase and associated with the serverside flow. Further client requests are allowed to reuse this flow only if they present the

NTLMConnPool cookie containing the matching information. By using a cookie in the NTLM profile, the BIG-IP system does not need to act as an NTLM proxy, and returning clients do not need to be re-authenticated.

The NTLM profile works by parsing the HTTP request containing the NTLM type 3 message and securely storing the following pieces of information (aside from those which are disabled in the profile):

- User name
- Workstation name
- Target server name
- Domain name
- Cookie previously set (cookie name supplied in the profile)
- Source IP address

With the information safely stored, the BIG-IP system can then use the data as a key when determining which clientside requests to associate with a particular serverside flow. You can configure this using the NTLM profile options. For example, if a server's resources can be openly shared by all users in that server's domain, then you can enable the Key By NTLM Domain setting, and all serverside flows from the users of the same domain can be pooled for connection reuse without further authentication. Or, if a server's resources can be openly shared by all users originating from a particular IP address, then you can enable the Key By Client IP Address setting and all serverside flows from the same source IP address can be pooled for connection reuse.

The Statistics profile type

The Statistics profile provides user-defined statistical counters. Each profile contains 32 settings (Field1 through Field32), which define named counters. Using a Tcl-based iRule command, you can use the names to manipulate the counters while processing traffic.

For example, you can create a profile named `my_stats`, which assigns the counters `tot_users`, `cur_users`, and `max_users` to the profile settings **Field1**, **Field2**, and **Field3** respectively. You can then write an iRule named `track_users`, and then assign the `my_stats` profile and the `track_users` iRule to a virtual server named `stats-1`.

In this example, the counter `tot_users` counts the total number of connections, the counter `cur_users` counts the current number of connections, and the counter `max_users` retains the largest value of the counter `cur_users`.

```
profile stats my_stats {
  defaults from stats
  field1 tot_users
  field2 cur_users
  field3 max_users
}

rule track_users {
  when CLIENT_ACCEPTED {
    STATS::incr my_stats tot_users
    STATS::setmax my_stats max_users [STATS::incr my_stats cur_users]
}
}

virtual stats-1 {
  destination 10.10.55.66:http
```

```
    ip protocol tcp
    profile http my_stats tcp
    pool pool1
    rule track_users
}
```

The Stream profile type

You can use the *Stream profile* to search and replace strings within a data stream, such as a TCP connection.

Note that list types are case-sensitive for pattern strings. For example, the system treats the pattern string `www.f5.com` differently from the pattern string `www.F5.com`. You can override this case sensitivity by using the Linux `regexp` command.

The Request Logging profile type

A *Request Logging profile* gives you the ability to configure data within a log file for HTTP requests and responses, according to parameters that you specify.

The DNS Logging profile type

A DNS logging profile gives you the ability to log DNS queries and responses, according to parameters that you specify.

Legal Notices

Legal notices

Publication Date

This document was published on November 10, 2017.

Publication Number

MAN-0539-02

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.

RF Interference Warning

This is a Class A product. In a domestic environment this product may cause radio interference, in which case the user may be required to take adequate measures.

FCC Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This unit generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Any modifications to this device, unless expressly approved by the manufacturer, can void the user's authority to operate this equipment under part 15 of the FCC rules.

Canadian Regulatory Compliance

This Class A digital apparatus complies with Canadian ICES-003.

Standards Compliance

This product conforms to the IEC, European Union, ANSI/UL and Canadian CSA standards applicable to Information Technology products at the time of manufacture.

Index

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X