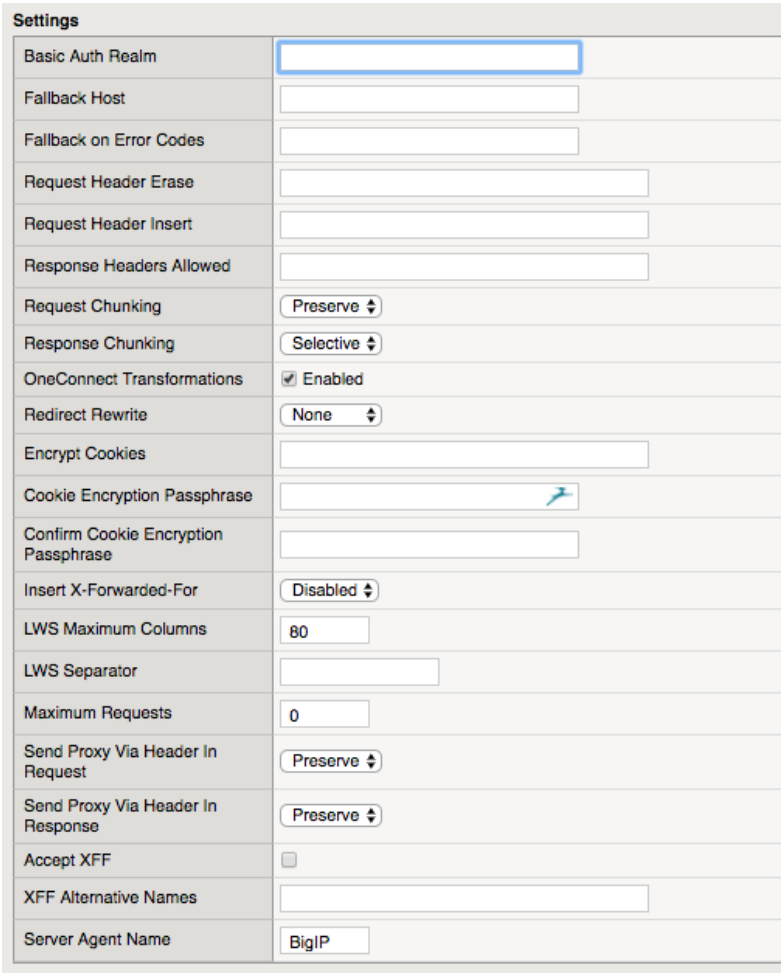
What is HTTP Part V - HTTP Profile Basic Settings

Jason Rahm, 2017-05-10

In the first four parts of this series on HTTP, we laid the foundation for understanding what's to come. In this article, we'll focus on the basic settings in the BIG-IP HTTP profile. Before we dive into the HTTP-specific settings, however, let's briefly discuss the nature of a profile. In the BIG-IP, every application is serviced by one or more virtual servers. There are literally hundreds of individual settings one can make to tune the various protocols used to service applications. Tweaking every individual setting for every individual application is not only time consuming, but at risk for human error. A profile allows an administrator to configure a sub-set of (usually) protocol specific settings that can then be applied to one or more virtual servers. Profiles can also be tiered in that you can have a baseline parent profile, and then use that to create child profiles based on that parent profile to tweak individual settings as necessary. The built-in profiles can be changed but it is not recommended you do so. The recommendation is to create a child based on a built-in profile, and customize from there.

The settings we'll focus on today are shown in the screen capture below. Note that this is TMOS version 12.1 HF1. The HTTP profile has oscillated over the versions to include or disclude various feature sets, so don't be alarmed if your view is different. There are a few proxy modes available in the HTTP profile, but we'll focus on reverse proxy mode.



Basic Auth Realm

The realm is a scope of protection for resources on an origin server. Think of realms as partitions where each can have their own authentication schemes. When authentication is required, the server should respond with a **status of 401** and the **WWW-Authenticate header**. If you set this field to testrealm, the header value returned to the client will be **Basic realm="testrealm"**. On the client end, when this response is received, a browser pop-up will trigger and provide a message like **Please enter your username and password for :** though some browsers do not provide the realm in this message.

Fallback

The two fallback fields in the profile are useful for handling adverse conditions in availability of your server pool of resources. The Fallback on Error Codes field allows you to set status codes in a space-separated list that will redirect to your Fallback Host when the server responds with any of those codes. This is useful hiding back end issues for a better user experience as well as securing any underlying server information leakage. The Fallback Host by itself is utilized when there are no available nodes to send requests to. When this happens, a **status of 302** (a temporary redirect) with the specified host is returned to the client.

Header Insertions, Erasures, & Allowances

The profile supports limited insert/remove/sanitize functionality. On requests, you can select a single header to insert and a single header to remove. There is a precedence for insertions and removals you'll want to keep in mind if you are inserting and removing the same header. Not sure why you'd do that but I tested and it appears that the HSTS and X-Forwarded-For features will be inserted first, if either are specified in the Request Header Erase they will be removed, and if also specified in the Request Header Insert, will be re-inserted on the way to the server.

On responses, you can specify a space-separated list of allowed headers and headers not in that list will be removed before the response is returned to the client. If any of that doesn't meet your needs, you can further customize the experience with local traffic policies or iRules, which we'll dig into in part nine of this series.

Chunking

The request/response chunking fields help deal with compression. We'll talk about compression in part eight of this series. These fields control how BIG-IP handles chunked content from clients and servers as indicated in the **Transfer-Encoding header**. The table below shows how the settings in the profile handle chunked and unchecked data for requests and responses.

Setting	Content State	Action on Request	Action on Response
Preserve	Chunked	processes chunked content, sends to server unchanged	processes chunked content, sends to client unchanged
	Unchunked	processes content, sends to server unchanged	processes content, sends to client unchanged
Selective	Chunked	unchunks HTTP content, processes it, re-adds the chunk headers, sends updated content to server	unchunks HTTP content, processes it, re-adds the chunk headers, sends updated content to client
	Unchunked	processes HTTP content, sends to server unchanged	processes HTTP content, sends to server unchanged
Rechunk	Chunked	unchunks HTTP content, processes it, re-adds the chunk headers, sends updated content to server	unchunks HTTP content, processes it, re-adds the chunk headers, sends updated content to client
	Unchunked	processes HTTP content, adds transfer encoding headers and chunk headers to the response, sends the chunked request to the server	processes HTTP content, adds transfer encoding headers and chunk headers to the response, sends the chunked request to the client

OneConnect Transformations

We'll address OneConnect in part seven of this series, but basically, by enabling transformations AND having a OneConnect profile, it allows the system to change non-keepalive connection requests on the client side to keep alive connections on the server side. If this feature is enabled WITHOUT a OneConnect profile, which is default, there is no action taken.

Redirect Rewrites

SSL offload is a very popular component for BIG-IP deployments. Traffic from client->BIG-IP is typically secured with SSL, and often, not protected from BIG-IP->server. Some servers are either not configured or just not able to handle returning secured URLs if the server is serving unsecured content. There are many ways to handle this, but this particular setting can be configured to assure that all or request-matching URLs are rewritten from http:// to https:// links. For redirects to node IP addresses, you can also configure the redirect to be rewritten to the virtual server address instead. The default for this setting is to not rewrite redirects. The actions taken for each option in the rewrites is as follows:

- None: Specifies that the system does not rewrite the URI in any HTTP redirect responses.
- All: Specifies that the system rewrites the URI in all HTTP redirect responses.
- Matching: Specifies that the system rewrites the URI in any HTTP redirect responses that match the request URI.
- Nodes: Specifies that if the URI contains a node IP address instead of a host name, the system changes it to the virtual server address.

Note that it is not content that is being rewritten here. Only the Location header in the redirect is updated.

Cookie Encryption

This is a security feature, enabling you to take an unencrypted cookie in a server response, encrypt it with 192-bit AES cipher, and b64 encode it before sending the response to the client. Multiple cookies can be specified in a space-separated list. This can also be done in iRules against all cookies without calling out specific cookie names.

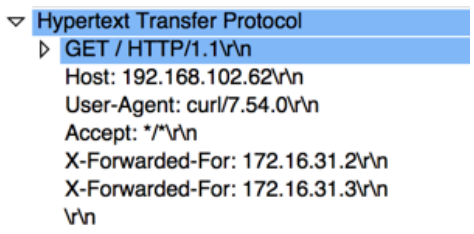
X-Forwarded-For

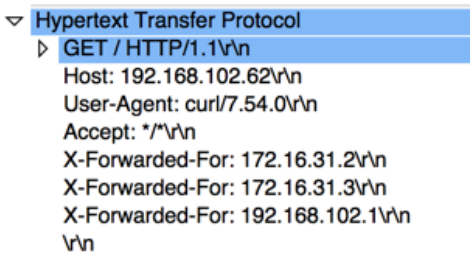
The X-Forwarded-For header is a de-facto standard for passing client IP addresses to servers when the true nature of the force (ahem, the IP path) is not visible to servers. This is common with network address translation being prevalent between clients and server. By default the Insert field is disabled, but if enabled, the client-side IP address of the packets being received by BIG-IP will be inserted into the X-Forwarded-For header and sent to the server. As X-Forwarded-For is a de-facto standard, that means that it is no standard at all, and some proxies with insert an additional header and some will just append to a header. The Accept XFF and XFF Alternate Names fields are used for system level statistics and whether they can be trusted. More details on that (specific to ASM) can be found in solution K12264.

Enabling X-Forwarded-For in the HTTP profile **does not** sanitize existing X-Forwarded-For headers already present in the client request. If there are two present in the request, then BIG-IP if configured to insert one will add a third. You can test this easily with curl:

```
curl -H "X-Forwarded-For: 172.16.31.2" -H "X-Forwarded-For: 172.16.31.3" http://www.test.local/
```

Here are my wireshark captures on my Ubuntu test server before and after enabling the XFF header insertion in the profile:





If you want to pass one X-Forwarded-For and one only to the server, an iRule is a good idea to achieve this behavior.

Linear White Space

There are two fields for handling linear white space: max columns (default: 80) and the separator (default: \r\n). Linear white space is any number of spaces, tabs, newlines IF followed by at least one space or tab. From RFC2616:

> A CRLF is allowed in the definition of TEXT only as part of a header field continuation. It is expected that the folding LWS will be replaced with a single SP before interpretation of the TEXT value.

I've not had a reason to ever change these settings. you can test for the presence of linear white space in a header with the HTTP::header lws command. This stack thread has a good answer on what linear white space is.

Max Requests

This is pretty straightforward. The default of zero does not limit the number of HTTP requests per connection, but changing this number will limit client requests on a per-connection basis to the value specified.

Via Headers

Regardless of request or response, this setting controls how BIG-IP handles the **Via heade**r. Like trace route for IP packets, the Via header will be appended with each intermediary along the way, so when the message reaches its destination, the HTTP path (not the IP path) should be known (whereas all HTTP/1.1 proxies are required to participate, HTTP/1.0 proxies that may or may not append the Via header.) When updating the header, the proxy is required to identify their hostname (or a pseudonym) and the HTTP version of the **previous** server in the path. Each proxy appends the information in sequential order to the header. Here's an example Via header from Mozilla's developer portal.

```
Via: 1.1 vegur
Via: HTTP/1.1 GWA
Via: 1.0 fred, 1.1 p.example.net
```

The available selections for this setting, request or response, is to preserve, remove, or append the Via header.

Server Agent Name

This is the value that BIG-IP populates in the **Server header** for response traffic. The default of BigIP gives away infrastructure clues, so many implementations will change this default. The Server header is analogous in response traffic to the User Agent header in client traffic.

What questions do you have? Drop a comment below if there is anything I can clarify. Join me next week when we move on to the HTTP profile enforcement settings.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com