ATS Test Documentation

Release 0.1

Feifei Cai

Jul 20, 2017

Contents

Contents:

CHAPTER 1

HTTP

Keep-alive

1. Keep-alive incoming connections.

   ```
   proxy.config.http.keep_alive_enabled_in
   ```

   Set to `1`: Test if client re-uses the TCP connection to Traffic Server for new HTTP transactions.

   Set to `0`: Test if client opens new TCP connections to Traffic Server for new HTTP transactions.

2. Keep-alive outgoing connections.

   ```
   proxy.config.http.keep_alive_enabled_out
   ```

   Set to `1`: Test if Traffic Server re-uses the TCP connection to origin server for new HTTP transactions.

   Set to `0`: Test if Traffic Server opens new TCP connections to origin server for new HTTP transactions.

3. Keep-alive connections for POST requests.

   ```
   proxy.config.http.keep_alive_post_out
   ```

   Set to `1`: Test if Traffic Server re-uses the TCP connection for new POST requests.

   Set to `0`: Test if Traffic Server opens new TCP connections for new POST requests.

Done: test_keepalive, test_keepalive2

Connection Timeouts

1. Keep-alive connection timeout.

   ```
   proxy.config.http.keep_alive_no_activity_timeout_in
   proxy.config.http.keep_alive_no_activity_timeout_out
   ```

Test if Traffic Server closes TCP connection to client after `N` seconds since completing last HTTP transaction.

Test if Traffic Server closes TCP connection to origin server after `N` seconds since completing last HTTP transaction.

2. No activity transacion timeout.

```
proxy.config.http.transaction_no_activity_timeout_in
proxy.config.http.transaction_no_activity_timeout_out
```

Test if Traffic Server keeps connection to client for `N` seconds when a transaction stalls.

Test if Traffic Server keeps connection to origin server for `N` seconds when a transaction stalls.

3. Active transaction timeout.

```
proxy.config.http.transaction_active_timeout_in
proxy.config.http.transaction_active_timeout_out
```

Test if Traffic Server keeps connection to client for `N` seconds, then close the connection even if the transfer to client is not complete.

Test if Traffic Server keeps connection to origin server for `N` seconds, then close the connection even if the transfer to origin server is not complete.

4. No activity connection timeout.

```
proxy.config.http.accept_no_activity_timeout
```

Set to `N`: Test if Traffic Server closes a connection that has no activity after `N` seconds.

5. Active background fill timeout.

```
proxy.config.http.background_fill_active_timeout
```

Set to `N`: Test if Traffic Server keeps a background fill for `N` seconds before giving up and dropping the origin server connection.

6. Background fill completed threshold.

```
proxy.config.http.background_fill_completed_threshold
```

Set to `0.x`: Test the proportion of total document size already transferred when a client aborts at which the proxy continues fetching the document from the origin server to get it into the cache (a background fill).

Done: test_timeouts

Origin Server Connect Attempts

1. Connect attempts timeout.

```
proxy.config.http.connect_attempts_timeout
```

Set to `N`: Test if Traffic Server attemps to connect to origin server for `N` seconds before receive the first byte.

2. Connect attempts timeout for POST/PUT requests.

```
proxy.config.http.post_connect_attempts_timeout
```

Set to `N`: Test if Traffic Server attemps to connect to origin server for `N` seconds before receive the first byte, when the client requests is a `POST` or `PUT` request.

3. Maximum number of connection retries.

```
proxy.config.http.connect_attempts_max_retries
```

Set to `N`: Test if Traffic Server retries `N` times, before the origin is marked dead.

4. Maximum number of connection retries to dead server.

```
proxy.config.http.connect_attempts_max_retries_dead_server
```

Set to `N`: Test if Traffic Server retries `N` times, after the origin is marked dead.

5. Maximum number of origin server connections.

```
proxy.config.http.server_max_connections
```

Set to `N`: Test if Traffic Server limits the number of socket connections across all origin servers to `N`.

6. Minimum number of origin server keep-alive connections.

```
proxy.config.http.origin_min_keep_alive_connections
```

Set to `N`: Test if Traffic Server keeps at least `N` connections open to origin server when connection is opened.

7. Maximum number of connection attempts with round-robin.

```
proxy.config.http.connect_attempts_rr_retries
```

Set to `N`: Test if Traffic Server retries `N` times before a round-robin entry is marked as 'down', if a server has round-robin DNS entries.

8. Server down cache time.

```
proxy.config.http.down_server.cache_time
```

Set to `N`: Test if Traffic Server remembers an origin server as unreachable for `N` seconds.

9. Server down abort threshold.

```
proxy.config.http.down_server.abort_threshold
```

Set to `N`: Test that, after a client abandons a request because an origin server was too slow in sending the response header, if Traffic Server marks that origin server as unavailable after `N` seconds.

Done: test_connect_attempts

100 Continue Response

```
proxy.config.http.send_100_continue_response
```

Set to `0`: Test if Traffic Server buffers the request until the post body has been received and then send the request to origin.

Set to `1`: Test if Traffic Server immediately returns a `100 Continue` response when receiving a `POST` request with `Expect: 100-continue` header, without waiting for the post body.

408 Post Timeout Response

```
proxy.config.http.send_408_post_timeout_response
```

Set to `1`: Test if Traffic Server sends a `408 Request Timeout` response when a `POST` request timeout happens.

Redirection

```
proxy.config.http.redirection_enabled
proxy.config.http.number_of_redirections
```

Set `proxy.config.http.redirection_enabled` to `1`, `proxy.config.http.number_of_redirections` to `N`: Test if Traffic Server do the redirecion for client, with a limit number of `N-1`.

Server Session

```
proxy.config.http.share_server_sessions
proxy.config.http.server_session_sharing.match
proxy.config.http.server_session_sharing.pool
proxy.config.http.attach_server_session_to_client
```

...

Headers

1. Via

   ```
   proxy.config.http.insert_request_via_str
   proxy.config.http.insert_response_via_str
   ```

   Set to `0` - `4`: Test if `via` header shows up with configurated verbosity in request or response.

2. Server

   ```
   proxy.config.http.response_server_enabled
   proxy.config.http.response_server_str
   ```

   Set to `0`: Test if Traffic Server sends response without a `Server` header.

   Set to `1` - `2`: Test if Traffic Server adds a `Server` header in response.

3. Age

   ```
   proxy.config.http.insert_age_in_response
   ```

Set to `0`: Test if Traffic Server sends response without an `Age` header.

Set to `1`: Test if Traffic Server sends response with an `Age` header.

4. From

```
proxy.config.http.anonymize_remove_from
```

Set to `1`: Test if Traffic Server removes the `from` header.

5. Referrer

```
proxy.config.http.anonymize_remove_referer
```

Set to `1`: Test if Traffic Server removes the `referrer` header.

6. User-agent

```
proxy.config.http.anonymize_remove_user_agent
```

Set to `1`: Test if Traffic Server removes the `User-agent` header.

7. Cookie

```
proxy.config.http.anonymize_remove_cookie
```

Set to `1`: Test if Traffic Server removes the `Cookie` header.

8. Client-IP

```
proxy.config.http.anonymize_remove_client_ip
```

Set to `1`: Test if Traffic Server removes the `Client-IP` header.

9. X-Forwarded-For

```
proxy.config.http.insert_squid_x_forwarded_for
```

Set to `1`: Test if Traffic Server adds the client IP address to the `X-Forwarded-For` header.

10. Accept-Encoding

```
proxy.config.http.normalize_ae_gzip
```

Set to `1`: Test if Traffic Server normalizes all `Accept-Encoding` headers.

11. Others

```
proxy.config.http.anonymize_other_header_list
```

Set to `header1;header2;...`: Test if Traffic Server removes these comma separated list of headers from outgoing requests.

Domain Expansion

```
proxy.config.http.enable_url_expandomatic
```

Set to `1`: Test if Traffic Server does domain expansion, which resolves unqualified hostnames by prepending with `www.` and appending with `.com` before redirecting to the expanded address.

Chunked Response

```
proxy.config.http.chunking_enabled
```

Set to `0` - `3`: Test if Traffic Server can generate a chunked response.

Done: test_chunked

HTTP 1.1 Requests

```
proxy.config.http.send_http11_requests
```

Set to `0` - `3`: Test if Traffic Server uses HTTP/1.1 to communicate with the origin server.

Use Client Tartget Address

```
proxy.config.http.use_client_target_addr
```

Set to `1`: Test if Traffic Server uses the same origin server address as the client in fully transparent ports.

CHAPTER 2

SSL

Certificate Configurations

Configure server certificate and private key file.

Test if SSL connection works.

Protocol Version

```
proxy.config.ssl.SSLv2
proxy.config.ssl.SSLv3
proxy.config.ssl.TLSv1
proxy.config.ssl.TLSv1_1
proxy.config.ssl.TLSv1_2
```

Set `proxy.config.ssl.SSLv2` to `0`, ..., `proxy.config.ssl.TLSv1` to `1`, ...: Test if Traffic Server does not use SSL version 2, ..., uses TLS version 1, ..., when it handshakes with client.

Protocol Version (Traffic Server as client)

```
proxy.config.ssl.client.SSLv2
proxy.config.ssl.client.SSLv3
proxy.config.ssl.client.TLSv1
proxy.config.ssl.client.TLSv1_1
proxy.config.ssl.client.TLSv1_2
```

Set `proxy.config.ssl.SSLv2` to `0`, ..., `proxy.config.ssl.TLSv1` to `1`, ...: Test if Traffic Server does not use SSL version 2, ..., uses TLS version 1, ..., when it handshakes with origin server.

Cipher Suite

```
proxy.config.ssl.server.cipher_suite
```

Set `proxy.config.ssl.server.cipher_suite` to `ECDHE-RSA-AES256-GCM-SHA384:...`: Test if Traffic Server is using a cipher suite in the list when it handshakes with client.

Cipher Suite (Traffic Server as client)

```
proxy.config.ssl.client.cipher_suite
```

Set `proxy.config.ssl.client.cipher_suite` to `ECDHE-RSA-AES256-GCM-SHA384:...`: Test if Traffic Server is using a cipher suite in the list when it handshakes with origin server.

Multicert Loading

The ssl_multicert.config file lets you configure Traffic Server to use multiple SSL server certificates to terminate the SSL sessions. If you have a Traffic Server system with more than one IP address assigned to it, then you can assign a different SSL certificate to be served when a client requests a particular IP address or host name.

Rewrite with new_tsqa: test-multicert-loading

Privilege Elevation

Set `records.config`:

```
CONFIG proxy.config.ssl.cert.load_elevated INT 1
CONFIG proxy.config.plugin.load_elevated INT 1

CONFIG proxy.config.diags.debug.enabled INT 1
CONFIG proxy.config.diags.debug.tags STRING 'privileges'
```

Rewrite with new_tsqa: test-privilege-elevation

Verify Client's Certificate

We can authenticate client during SSL handshake, and client is required to provide a certificate.

```
proxy.config.ssl.client.certification_level
```

Set `records.config`:

This is a test certificate signed by a test CA, which is not in system PKI or browser. So, we need to specify the CA certificate:

```
# 1: optional 2: required
CONFIG proxy.config.ssl.client.certification_level INT 2

# specify CA file name and path, who signed client's certificate
```

```
CONFIG proxy.config.ssl.CA.cert.filename STRING ca.crt
CONFIG proxy.config.ssl.CA.cert.path STRING etc/trafficserver
```

Test if Traffic Server verifies client's certificate.

Verify Origin Server's Certificate

We can configure it to verify the origin server certificate with the Certificate Authority (CA).

Notice: By default, Traffic Server does not verify the origin server!

```
proxy.config.ssl.client.verify.server
```

Set *records.config*:

```
CONFIG proxy.config.ssl.client.verify.server INT 1

# specify CA file name and path, who signed origin server's certificate
CONFIG proxy.config.ssl.client.CA.cert.filename STRING ca.crt
CONFIG proxy.config.ssl.client.CA.cert.path STRING etc/trafficserver
```

Test if Traffic Server verifies origin server's certificate.

Verified by Origin Server

Send origin server certificate for verification. Here Traffic Server is SSL client, and origin server requires to verify Traffic Server.

```
proxy.config.ssl.client.cert.filename
proxy.config.ssl.client.cert.path
proxy.config.ssl.client.private_key.filename
proxy.config.ssl.client.private_key.path
```

Test if Traffic Server passed origin server's verification.

SNI

Done: test_https

Session Reuse

1. Session Ticket
2. Session ID

OCSP Stapling

```
proxy.config.ssl.ocsp.enabled
```

By default, Traffic Server does not enable OCSP Stapling.

1. Good OCSP response.

   Generate a test certificate with OCSP extensions; start an OCSP server. Test if Traffic Server staples the **good** OCSP response and sends it to client along with certificate in SSL handshake.

2. Revoked OCSP response.

   Generate a test certificate with OCSP extensions, then revoke it; start an OCSP server. Test if Traffic Server staples the **revoked** OCSP response and sends it to client along with certificate in SSL handshake.

3. Unknown OCSP response.

   Generate a test certificate with OCSP extensions, then remove the entry in test CA's database; start an OCSP server. Test if Traffic Server staples the **unknown** OCSP response and sends it to client along with certificate in SSL handshake.

Dual Certificate (ECDSA + RSA)

...

CHAPTER 3

Remap

Map Type

1. map

   Translates an incoming request URL to the appropriate origin server URL.

   Test if `map` rule works.

   Done: test_remap

2. map_with_recv_port

   Exactly like `map` except that `map_with_recv_port` uses the port at which the request was received to perform the mapping instead of the port present in the request.

   Test if `map_with_recv_port` rule works.

3. map_with_referer

   Extended version of `map`, which can be used to activate "deep linking protection", where target URLs are only accessible when the Referer header is set to a URL that is allowed to link to the target.

   Test if `map_with_referer` rule works.

4. reverse_map

   Translates the URL in origin server redirect responses to point to Traffic Server.

   Test if `reverse_map` rule works.

5. redirect

   Redirects HTTP requests permanently without having to contact the origin server. Permanent redirects notify the browser of the URL change (by returning an HTTP status code 301) so that the browser can update bookmarks.

   Test if `redirect` rule works.

6. redirect_temporary

Redirects HTTP requests temporarily without having to contact the origin server. Temporary redirects notify the browser of the URL change for the current request only (by returning an HTTP status code 307).

Test if `redirect_temporary` rule works.

Precedence

Remap rules are not processed top-down, but based on an internal priority. Once these rules are executed we pick the first match based on configuration file parse order.

1. `map_with_recv_port` and `regex_map_with_recv_port`
2. `map and regex_map` and `reverse_map`
3. `redirect` and `redirect_temporary`
4. `regex_redirect` and `regex_redirect_temporary`

Test if it follows the above priority.

Test if it follows the parse order within same priority.

Match-All

A map rule with a single `/` acts as a wildcard, it will match any request. This should be use with care, and certainly only once at the end of the remap.config file. E.g.

```
map / http://all.example.com
```

Test if `Match-All` rule works.

Regex Map Type

1. regex_map

   The `regex` qualifier can also be used for `map`. When present, `map` mappings are checked first. If there is a match, then it is chosen without evaluating the "regular expression" mapping rules.

   Test if *regex_map* rule works.

2. map_with_recv_port

   The `regex` qualifier can also be used for `map_with_recv_port`. When present, `map_with_recv_port` mappings are checked first. If there is a match, then it is chosen without evaluating the "regular expression" mapping rules.

   Test if `regex_map_with_recv_port` rule works.

3. redirect

   The `regex` qualifier can also be used for `redirect`. When present, `redirect` mappings are checked first. If there is a match, then it is chosen without evaluating the "regular expression" mapping rules.

   Test if *regex_redirect* rule works.

4. redirect_temporary

   The `regex` qualifier can also be used for `redirect_temporary`. When present, `redirect_temporary` mappings are checked first. If there is a match, then it is chosen without evaluating the "regular expression" mapping rules.

   Test if `regex_redirect_temporary` rule works.

Plugin Chaining

Test if plugins can be configured to be evaluated in a specific order, passing the results from one in to the next.

ACL Filters

Test if ACL filters work.

Named Filters

Test if named filters work.

Including Additional Remap Files

Test if `.include` directive works.

CHAPTER 4

Cache

Cache Control

1. Cache

   ```
   proxy.config.http.cache.http
   ```

   Set to `0`: Test if Traffic Server disables caching HTTP requests.

   Set to `1`: Test if Traffic Server enables caching HTTP requests.

Caching HTTP Objects

1. Client Directives

   By default, Traffic Server does not cache objects with the following request headers:

   - `Authorization`
   - `Cache-Control: no-store`
   - `Cache-Control: no-cache`
   - `Cookie` (for text objects)

   We need to test the following scenarios:

   - Test if Traffic Server does not cache objects with the above request headers.
   - Test if Traffic Server can ignore client no-cache headers.
   - Test if Traffic Server caches cookied content.

2. Origin Server Directives

By default, Traffic Server does not cache objects with the following response headers:

- `Cache-Control: no-store`
- `Cache-Control: private`
- `WWW-Authenticate`
- `Set-Cookie`
- `Cache-Control: no-cache`
- `Expires`

We need to test the following scenarios:

- Test if Traffic Server does not cache objects with the above response headers.
- Test if Traffic Server can ignore `WWW-Authenticate` headers.
- Test if Traffic Server can ignore server `no-cache` headers.

Heuristic Expiration

```
proxy.config.http.cache.heuristic_min_lifetime
proxy.config.http.cache.heuristic_max_lifetime
proxy.config.http.cache.heuristic_lm_factor
```

Test if the heuristic minimum lifetime, maximum lifetime and aging factor take effect.

```
proxy.config.http.cache.fuzz.time
proxy.config.http.cache.fuzz.probability
proxy.config.http.cache.fuzz.min_time
```

Test if the fuzz time, probability and minimum time take effect.

Dynamic Content & Content Negotiation

```
proxy.config.http.cache.vary_default_text
```

Test if Traffic Server varies on the specified header for text documents.

```
proxy.config.http.cache.vary_default_images
```

Test if Traffic Server varies on the specified header for images.

```
proxy.config.http.cache.vary_default_other
```

Test if Traffic Server varies on the specified header for anything other than text and images.

Negative Response Caching

```
proxy.config.http.negative_caching_enabled
```

Set to `0`: Test if Traffic Server does not cache negative responses when a requested page does not exist.

Set to `1`: Test if Traffic Server caches negative responses when a requested page does not exist.

```
proxy.config.http.negative_caching_lifetime
```

Set to `N`: Test if Traffic Server keeps the negative responses valid in cache for `N` seconds.

Negative Revalidate

PUSH

```
proxy.config.http.push_method_enabled
```

Set to `1`: Test if Traffic Server allows to deliver content directly to the cache without a user request via `PUSH` method.

CHAPTER 5

Others

Log

1. Configuration

   Rewrite with new_tsqa: test-log-configuration

2. Refcounting

   Rewrite with new_tsqa: test-log-refcounting

3. Heartbeat

   ```
   proxy.config.http.record_heartbeat
   ```

   Set to `1`: Test if Traffic Server enables `traffic_cop`'s heartbeat logging.

Traffic Line

Rewrite with new_tsqa: test-trafficline-metrics

Intercept Plugin

Set `remap.config`:

```
intercept.so
```

Done: test_example

CHAPTER 6

Indices and tables

- genindex
- modindex
- search