Traffic Control Documentation

Release 1.1.3

Comcast Cable

Jun 18, 2018

Contents

The vast majority of today's Internet traffic is media files being sent from a single source to many thousands or even millions of destinations. Content Delivery Networks make that one-to-many distribution possible in an economical way.

Traffic Control is an Open Source implementation of a Content Delivery Network.

The following documentation sections are available:

CHAPTER 1

CDN Basics

A review of the basic functionality of a Content Delivery Network.

1.1 CDN Basics

Traffic Control is a CDN control plane, see the topics below to familiarize yourself with the basic concepts of a CDN.

1.1.1 Content Delivery Networks

The vast majority of today's Internet traffic is media files (often video or audio) being sent from a single source (the *Content Provider*) to many thousands or even millions of destinations (the *Content Consumers*). Content Delivery Networks are the technology that make that one-to-many distribution possible in an economical way. A Content Delivery Network (CDN) is a distributed system of servers for delivering content over HTTP. These servers are deployed in multiple locations with the goal of optimizing the delivery of content to the end users, while minimizing the traffic on the network. A CDN typically consists of the following:

- **Caching Proxies** The proxy (cache or caching proxy) is a server that both proxies the requests and caches the results for reusing.
- **Content Router** The Content Router ensures that the end user is connected to the optimal cache for the location of the end user and content availability.
- **Health Protocol** The Health Protocol monitors the usage of the caches and tenants in the CDN.
- **Configuration Management System** In many cases a CDN encompasses hundreds of servers across a large geographic area. The Configuration Management System allows an operator to manage these servers.
- **Log File Analysis System** Every transaction in the CDN gets logged. The Log File Analysis System aggregates all of the log entries from all of the servers to a central location for analysis and troubleshooting.

1.1.2 HTTP 1.1

For a comprehensive look at Traffic Control, it is important to understand basic HTTP 1.1 protocol operations and how caches function. The example below illustrates the fulfillment of an HTTP 1.1 request in a situation without CDN or proxy, followed by viewing the changes after inserting different types of (caching) proxies. Several of the examples below are simplified for clarification of the essentials.

For complete details on HTTP 1.1 see RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1.

Below are the steps of a client retrieving the URL `http://www.origin.com/foo/bar/fun.html` using HTTP/1.1 without proxies:

1. The client sends a request to the Local DNS (LDNS) server to resolve the name `www.origin.com` to an IPv4 address.
2. If the LDNS does not have this name (IPv4 mapping cached), it sends DNS requests to the ., .com, and .origin.com authoritative servers until it receives a response with the address for `www.origin.com`. Per the DNS SPEC, this response has a Time To Live (TTL), which indicates how long this mapping can be cached at the LDNS server. In the example, the IP address found by the LDNS server for www.origin.com is 44.33.22.11.

   Note: While longer DNS TTLs of a day (86400 seconds) or more are quite common in other use cases, in CDN use cases DNS TTLs are often below a minute.

3. The client opens a TCP connection from a random port locally to port 80 (the HTTP default) on 44.33.22.11, and sends this (showing the minimum HTTP 1.1 request, typically there are additional headers):

```
GET /foo/bar/fun.html HTTP/1.1
Host: www.origin.com
```

4. The server at `www.origin.com` looks up the Host: header to match that to a configuration section, usually referred to as a virtual host section. If the Host: header and configuration section match, the search continues for the content of the path `/foo/bar/fun.html`, in the example, this is a file that contains `<html><body>This is a fun file</body></html>`, so the server responds with the following:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 45

<html><body>This is a fun file</body></html>
```

   At this point, HTTP transaction is complete.

1.1.3 Caching Proxies

The main function of a CDN is to proxy requests from clients to origin servers and cache the results. To proxy, in the CDN context, is to obtain content using HTTP from an origin server on behalf of a client. To cache is to store the results so they can be reused when other clients are requesting the same content. There are three types of proxies in use on the Internet today which are described below.

➡ Reverse Proxy

A reverse proxy acts on behalf of the origin server. The client is mostly unaware it is communicating with a proxy and not the actual origin. All EDGE caches in a Traffic Control CDN are reverse proxies. To the end user a Traffic Control based CDN appears as a reverse proxy since it retrieves content from the origin server, acting on behalf of that origin server. The client requests a URL that has a hostname which resolves to the reverse proxy's IP address and, in compliance with the HTTP 1.1 specification, the client sends a `Host:` header to the reverse proxy including the hostname in the URL. The proxy looks up the hostname in a list of mappings to find the origin hostname; if the origin hostname is not found in the list, the proxy connects to the origin host and requests the path of the original URL. The proxy then stores the URL in cache and serves the contents to the client. When there are subsequent requests for the same URL, a caching proxy serves the content out of cache thereby reducing latency and network traffic.

See also:

ATS documentation on reverse proxy.

To insert a reverse proxy into the previous HTTP 1.1 example, the reverse proxy requires provisioning for `www.origin.com`. By adding a remap rule to the cache, the reverse proxy then maps requests to this origin. The content owner must inform the clients, by updating the URL, to receive the content from the cache and not from the origin server directly. For this example, the remap rule on the cache is: `http://www-origin-cache.cdn.com http://www.origin.com`.

Note: In the previous example minimal headers were shown on both the request and response. In the examples that follow, the origin server response is more realistic.

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8

<html><body>This is a fun file</body></html>
```

The client is given the URL `http://www-origin-cache.cdn.com/foo/bar/fun.html` (note the different hostname) and when attempting to obtain that URL, the following occurs:

1. The client sends a request to the LDNS server to resolve the name `www-origin-cache.cdn.com` to an IPv4 address.
2. Similar to the previous case, the LDNS server resolves the name `www-origin-cache.cdn.com` to an IPv4 address, in this example, this address is 55.44.33.22.
3. The client opens a TCP connection from a random port locally, to port 80 (the HTTP default) on 55.44.33.22, and sends the following:

   ```
   GET /foo/bar/fun.html HTTP/1.1
   Host: www-origin-cache.cdn.com
   ```

4. The reverse proxy looks up `www-origin-cache.cdn.com` in its remap rules, and finds the origin is `www.origin.com`.
5. The proxy checks its cache to see if the response for `http://www-origin-cache.cdn.com/foo/bar/fun.html` is already in the cache.

6a. If the response is not in the cache:

1. The proxy uses DNS to get the IPv4 address for `www.origin.com`, connect to it on port 80, and sends:

```
GET /foo/bar/fun.html HTTP/1.1
Host: www.origin.com
```

2. The origin server responds with the headers and content as shown:

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8

<html><body>This is a fun file</body></html>
```

3. The proxy sends the origin response on to the client adding a `Via:` header (and maybe others):

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8
Age: 0
Via: http/1.1 cache01.cdn.kabletown.net (ApacheTrafficServer/4.2.1␣
↪[uScSsSfUpSeN:t cCSi p sS])
Server: ATS/4.2.1

  <html><body>This is a fun file</body></html>
```

6b. If it *is* in the cache:

The proxy responds to the client with the previously retrieved result:

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8
Age: 39711
Via: http/1.1 cache01.cdn.kabletown.net (ApacheTrafficServer/4.2.1␣
↪[uScSsSfUpSeN:t cCSi p sS])
Server: ATS/4.2.1

<html><body>This is a fun file</body></html>
```

Forward Proxy

A forward proxy acts on behalf of the client. The origin server is mostly unaware of the proxy, the client requests the proxy to retrieve content from a particular origin server. All MID caches in a Traffic Control based CDN are forward proxies. In a forward proxy scenario, the client configuration is with the proxy's

IP address and port. The client always connects to the forward proxy for content. The content provider does not have to change the URL the client obtains, and is unaware of the proxy in the middle.

See also:

ATS documentation on forward proxy.

Below is an example of the client retrieving the URL `http://www.origin.com/foo/bar/fun.html` through a forward proxy:

1. The client requires configuration to use the proxy, as opposed to the reverse proxy example. Assume the client configuration is through preferences entries or other to use the proxy IP address 99.88.77.66 and proxy port 8080.
2. To retrieve `http://www.origin.com/foo/bar/fun.html` URL, the client connects to 99.88.77.66 on port 8080 and sends:

```
GET http://www.origin.com/foo/bar/fun.html HTTP/1.1
```

Note: In this case, the client places the entire URL after GET, including protocol and hostname (`http://www.origin.com`), but in the reverse proxy and direct-to-origin case it puts only the path portion of the URL (`/foo/bar/fun.html`) after the GET.

3. The proxy verifies whether the response for `http://www-origin-cache.cdn.com/foo/bar/fun.html` is already in the cache.

4a. If it is not in the cache:

1. The proxy uses DNS to obtain the IPv4 address for `www.origin.com`, connects to it on port 80, and sends:

```
GET /foo/bar/fun.html HTTP/1.1
Host: www.origin.com
```

2. The origin server responds with the headers and content as shown below:

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8

<html><body>This is a fun file</body></html>
```

3. The proxy sends this on to the client adding a `Via:` header (and maybe others):

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8
Age: 0
Via: http/1.1 cache01.cdn.kabletown.net (ApacheTrafficServer/4.2.1␣
↪[uScSsSfUpSeN:t cCSi p sS])
```

(continues on next page)

(continued from previous page)

```
Server: ATS/4.2.1

<html><body>This is a fun file</body></html>
```

4b. If it *is* in the cache:

The proxy responds to the client with the previously retrieved result:

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8
Age: 99711
Via: http/1.1 cache01.cdn.kabletown.net (ApacheTrafficServer/4.2.1␣
↪[uScSsSfUpSeN:t cCSi p sS])
Server: ATS/4.2.1

<html><body>This is a fun file</body></html>
```

➡ Transparent Proxy

Neither the origin nor the client are aware of the actions performed by the transparent proxies. A Traffic Control based CDN does not use transparent proxies. If you are interested you can learn more about transparent proxies on wikipedia.

1.1.4 Cache Control Headers and Revalidation

The HTTP/1.1 spec allows for origin servers and clients to influence how caches treat their requests and responses. By default, the Traffic Control CDN will honor cache control headers. Most commonly, origin servers will tell the downstream caches how long a response can be cached:

```
HTTP/1.1 200 OK
Date: Sun, 14 Dec 2014 23:22:44 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Cache-Control: max-age=86400
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8

<html><body>This is a fun file</body></html>
```

In the above response, the origin server tells downstream caching systems that the maximum time to cache this response for is 86400 seconds. The origin can also add a `Expires:` header, explicitly telling the cache the time this response is to be expired. When a response is expired it usually doesn't get deleted from the cache, but, when a request comes in that would have hit on this response if it was not expired, the cache *revalidates* the response. In stead of requesting the object again from the origin server, the cache will send a request to the origin indicating what version of the response it has, and asking if it has changed. If it changed, the server will send a `200 OK` response, with the new data. If it has not changed, the origin server will send back a `304 Not Modified` response indicating the response is still valid, and that the cache can reset the timer on the response expiration. To indicate what version the client (cache) has it will add an `If-Not-Modified-Since:` header, or an `If-None-Match:` header. For example, in the `If-None-Match:` case, the origin will send and `ETag` header that uniquely identifies the response. The client can use that in an revalidation request like:

```
GET /foo/bar/fun.html HTTP/1.1
If-None-Match: "1aa008f-2d-50a3559482cc0"
Host: www.origin.com
```

If the content has changed (meaning, the new response would not have had the same ETag) it will respond with `200 OK`, like:

```
HTTP/1.1 200 OK
Date: Sun, 18 Dec 2014 3:22:44 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50aa00feadd"
Cache-Control: max-age=604800
Content-Length: 49
Connection: close
Content-Type: text/html; charset=UTF-8

<html><body>This is NOT a fun file</body></html>
```

If the Content did not change (meaning, the response would have had the same ETag) it will respond with `304 Not Modified`, like:

```
304 Not Modified
Date: Sun, 18 Dec 2014 3:22:44 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Sun, 14 Dec 2014 23:18:51 GMT
ETag: "1aa008f-2d-50a3559482cc0"
Cache-Control: max-age=604800
Content-Length: 45
Connection: close
Content-Type: text/html; charset=UTF-8
```

Note that the 304 response only has headers, not the data.

CHAPTER 2

Traffic Control Overview

An introduction to the Traffic Control architecture, components, and their integration.

2.1 Traffic Control Overview

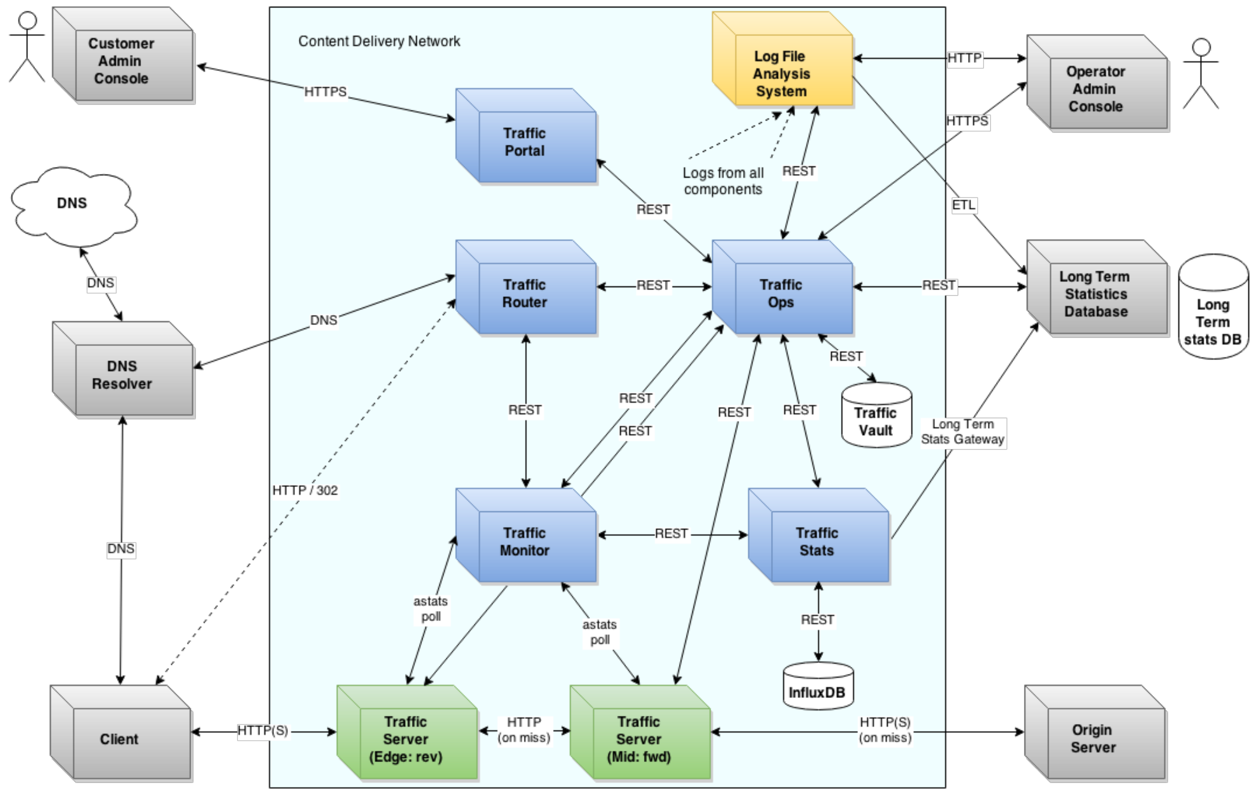
Introduces the Traffic Control architecture, components, and their integration.

2.1.1 Introduction

Traffic Control is a control plane for a CDN, which includes all of the components mentioned in the CDN Basics section, except for the Log File Analysis System. The caching software chosen for Traffic Control is Apache Traffic Server (ATS). Although the current release only supports ATS as a cache, this may change with future releases.

Traffic Control was first developed at Comcast for internal use and released to Open Source in April of 2015.

Traffic Control implements the blue boxes in the architecture diagram below.



In the next sections each of these components will be explained further.

2.1.2 Traffic Ops

Traffic Ops is the tool for administration (configuration and monitoring) of all components in a Traffic Control CDN. The CDN administrator uses Traffic Ops to manage servers, cache groups, delivery services, etc. In many cases, a configuration change requires propagation to several, or even all, caches and only explicitly after or before the same change propagates to Traffic Router. Traffic Ops takes care of this required consistency between the different components and their configuration. Traffic Ops exposes its data through a series of HTTP APIs and has a user interface that is interactive and viewable using a standard web browser.

Traffic Ops uses a MySQL or PostgreSQL database to store the configuration information, and the Mojolicious framework to generate the user interface and APIs. Not all configuration data is in this database however; for sensitive data, like SSL private keys or token based authentication shared secrets, a separate key-value store is used, allowing the operator to harden the server that runs this key-value store better from a security perspective (i.e only allow Traffic Ops access it with a cert). The Traffic Ops server, by design, needs to be accessible from all the other servers in the Traffic Control CDN.

Traffic Ops generates all the application specific configuration files for the caches and other servers. The caches and other servers check in with Traffic Ops at a regular interval (default 15 minutes) to see if updated configuration files require application.

Traffic Ops also runs a collection of periodic checks to determine the operational readiness of the caches. These periodic checks are customizable by the Traffic Ops admin using extensions.

➡ Traffic Ops Extension

Traffic Ops Extensions are a way to enhance the basic functionality of Traffic Ops in a custom manner. There are three types of extensions:

- Check Extensions - Allows you to add custom checks to the "Health->Server Checks" view.
- Configuration Extension - Allows you to add custom configuration file generators.
- Data source Extensions - Allows you to add data sources for the graph views and usage APIs.

2.1.3 Traffic Router

Traffic Router's function is to send clients to the most optimal cache. Optimal in this case is based on a number of factors:

- Distance between the cache and the client (not necessarily measured in meters, but quite often in layer 3 network hops). Less network distance between the client and cache yields better performance, and lower network load. Traffic Router helps clients connect to the best performing cache for their location at the lowest network cost.
- Availability of caches and health / load on the caches. A common issue in Internet and television distribution scenarios is having many clients attempting to retrieve the same content at the same time. Traffic Router helps clients route around overloaded or down caches.
- Availability of content on a particular cache. Reusing of content through cache HITs is the most important performance gain a CDN can offer. Traffic Router sends clients to the cache that is most likely to already have the desired content.

Traffic routing options are often configured at the Delivery Service level.

➡ Delivery Service

As discussed in the basic concepts section, the EDGE caches are configured as reverse proxies, and the Traffic Control CDN looks from the outside as a very large reverse proxy. Delivery Services are often referred to a reverse proxy remap rule. In most cases, a Delivery Service is a one to one mapping to a FQDN that is used as a hostname to deliver the content. Many options and settings regarding how to optimize the content delivery, which is configurable on a Delivery Service basis. Some examples of these Delivery Service settings are:

- Cache in RAM, cache on disk, or do not cache at all.
- Use DNS or HTTP Content routing (see below).
- Limits on transactions per second and bandwidth.
- Protocol (http or https).
- Token based authentication settings.
- Header rewrite rules.

Delivery Services are also for use in allowing multi-tenants to coexist in the Traffic Control CDN without interfering with each other, and to keep information about their content separated.

Localization

Traffic Router uses a JSON input file called the *coverage zone map* to determine what *cachegroup* is closest to the client. If the client IP address is not in this coverage zone map, it falls back to *geo*, using the maxmind database to find the client's location, and the geo coordinates from Traffic Ops for the cachegroup.

Traffic Router is inserted into the HTTP retrieval process by making it DNS authoritative for the domain of the CDN delivery service. In the example of the reverse proxy, the client was given the `http://www-origin-cache.cdn.com/foo/bar/fun.html url`. In a Traffic Control CDN, URLs start with either `tr.` or `edge.`, for HTTP or DNS content routing respectively. These names are configurable via properties files within the Traffic Router installation.

DNS Content Routing

For a DNS delivery service the client receives a URL with a hostname beginning with `edge.` (e.g. http://edge.dsname.cdn.com/foo/bar/fun.html). When the LDNS server is resolving this `edge.dsname.cdn.com` hostname to an IP address, it ends at Traffic Router because it is the authoritative DNS server for `cdn.com` and the domains below it, and subsequently responds with a list of IP addresses from the eligible caches based on the location of the LDNS server. When responding, Traffic Router does not know the actual client IP address or the path that the client is going to request. The decision on what cache IP address (or list of cache IP addresses) to return is solely based on the location of the LDNS server and the health of the caches. The client then connects to port 80 on the cache, and sends the `Host: edge.dsname.cdn.com` header. The configuration of the cache includes the remap rule `http://edge.dsname.cdn.com http://origin.dsname.com` to map that edge name to an origin hostname.

HTTP Content Routing

For an HTTP delivery service the client receives a URL with a hostname beginning with `tr.` (e.g. http://tr.dsname.cdn.com/foo/bar/fun.html), the LDNS server resolves this `tr.dsname.cdn.com` to an IP address, but in this case Traffic Router returns its own IP address. The client opens a connection to port 80 on the Traffic Router's IP address, and sends:

```
GET /foo/bar/fun.html HTTP/1.1
Host: tr.dsname.cdn.com
```

Traffic Router uses an HTTP 302 to redirect the client to the best cache. For example:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
Location: http://atsec-nyc-02.dsname.cdn.com/foo/bar/fun.html
Content-Length: 0
Date: Tue, 13 Jan 2015 20:01:41 GMT
```

The information Traffic Router can consider when selecting a cache in this case is much better:

- The client's IP address (the other side of the socket).
- The URL path the client is requesting.
- All HTTP 1.1 headers.

The client follows the redirect and performs a DNS request for the IP address for `atsec-nyc-02.dsname.cdn.com`, and normal HTTP steps follow, except the sending of the Host: header when connected to the cache is `Host:  atsec-nyc-02.dsname.cdn`, and the configuration of the cache includes the remap rule (e.g.``http://atsec-nyc-02.dsname.cdn http://origin.dsname.com``).

Traffic Router sends all requests for the same path in a delivery service to the same cache in a cache group using consistent hashing, in this case all caches in a cache group are not carrying the same content, and there is a much larger combined cache in the cache group.

In many cases DNS content routing is the best possible option, especially in cases where the client is receiving small objects from the CDN like images and web pages.

Traffic Router is redundant and horizontally scalable by adding more instances into the DNS hierarchy using NS records.

2.1.4 Traffic Monitor

Traffic Monitor is a Java/Tomcat application that monitors the caches in a CDN for a variety of metrics. These metrics are for use in determining the overall health of a given cache and the related delivery services. A given CDN can operate a number of Traffic Monitors, from a number of geographically diverse locations, to prevent false positives caused by network problems at a given site.

Traffic Monitors operate independently, but use the state of other Traffic Monitors in conjunction with their own state, to provide a consistent view of CDN cache health to upstream applications such as Traffic Router. Health Protocol governs the cache and Delivery Service availability.

Traffic Monitor provides a view into CDN health using several RESTful JSON endpoints, which are consumed by other Traffic Monitors and upstream components such as Traffic Router. Traffic Monitor is also responsible for serving the overall CDN configuration to Traffic Router, which ensures that the configuration of these two critical components remain synchronized as operational and health related changes propagate through the CDN.

Cache Monitoring

Traffic Monitor polls all caches configured with a status of `REPORTED` or `ADMIN_DOWN` at an interval specified as a configuration parameter in Traffic Ops. If the cache is set to `ADMIN_DOWN` it is marked as unavailable but still polled for availability and statistics. If the cache is explicitly configured with a status of `ONLINE` or `OFFLINE`, it is not polled by Traffic Monitor and presented to Traffic Router as configured, regardless of actual availability.

Traffic Monitor makes HTTP requests at regular intervals to a special URL on each EDGE cache and consumes the JSON output. The special URL is a plugin running on the Apache Traffic Server (ATS) caches called astats, which is restricted to Traffic Monitor only. The astats plugin provides insight into application and system performance, such as:

- Throughput (e.g. bytes in, bytes out, etc).
- Transactions (e.g. number of 2xx, 3xx, 4xx responses, etc).
- Connections (e.g. from clients, to parents, origins, etc).
- Cache performance (e.g.: hits, misses, refreshes, etc).
- Storage performance (e.g.: writes, reads, frags, directories, etc).
- System performance (e.g: load average, network interface throughput, etc).

Many of the application level statistics are available at the global or aggregate level, some at the Delivery Service (remap rule) level. Traffic Monitor uses the system level performance to determine the overall health of the cache by evaluating network throughput and load against values configured in Traffic Ops. Traffic Monitor also uses throughput and transaction statistics at the remap rule level to determine Delivery Service health.

If astats is unavailable due to a network related issue or the system statistics have exceeded the configured thresholds, Traffic Monitor will mark the cache as unavailable. If the delivery service statistics exceed the configured thresholds, the delivery service is marked as unavailable, and Traffic Router will start sending clients to the overflow destinations for that delivery service, but the cache remains available to serve other content,

See also:

For more information on ATS Statistics, see the ATS documentation

➡ Health Protocol

Redundant Traffic Monitor servers operate independently from each other but take the state of other Traffic Monitors into account when asked for health state information. In the above overview of cache monitoring, the behavior of Traffic Monitor pertains only to how an individual instance detects and handles failures. The Health Protocol adds another dimension to the health state of the CDN by merging the states of all Traffic Monitors into one, and then taking the *optimistic* approach when dealing with a cache or Delivery Service that might have been marked as unavailable by this particular instance or a peer instance of Traffic Monitor.

Upon startup or configuration change in Traffic Ops, in addition to caches, Traffic Monitor begins polling its peer Traffic Monitors whose state is set to `ONLINE`. Each `ONLINE` Traffic Monitor polls all of its peers at a configurable interval and saves the peer's state for later use. When polling its peers, Traffic Monitor asks for the raw health state from each respective peer, which is strictly that instance's view of the CDN's health. When any `ONLINE` Traffic Monitor is asked for CDN health by an upstream component, such as Traffic Router, the component gets the health protocol influenced version of CDN health (non-raw view).

In operation of the health protocol, Traffic Monitor takes all health states from all peers, including the locally known health state, and serves an optimistic outlook to the requesting client. This means that, for example, if three of the four Traffic Monitors see a given cache or Delivery Service as exceeding its thresholds and unavailable, it is still considered available. Only if all Traffic Monitors agree that the given object is unavailable is that state propagated to upstream components. This optimistic approach to the Health Protocol is counter to the "fail fast" philosophy, but serves well for large networks with complicated geography and or routing. The optimistic Health Protocol allows network failures or latency to occur without affecting overall traffic routing, as Traffic Monitors can and do have a different view of the network when deployed in geographically diverse locations. Short polling intervals of both the caches and Traffic Monitor peers help to reduce customer impact of outages.

It is not uncommon for a cache to be marked unavailable by Traffic Monitor - in fact, it is business as usual for many CDNs. A hot video asset may cause a single cache (say cache-03) to get close to it's interface capacity, the health protocol "kicks in", and Traffic Monitor marks cache-03 as unavailable. New clients want to see the same asset, and now, Traffic Router will send these customers to another cache (say cache-01) in the same cachegroup. The load is now shared between cache-01 and cache-03. As clients finish watching the asset on cache-03, it will drop below the threshold and gets marked available again, and new clients will now go back to cache-03 again.

It is less common for a delivery service to be marked unavailable by Traffic Monitor, the delivery service thresholds are usually used for overflow situations at extreme peaks to protect other delivery services in the CDN from getting impacted.

2.1.5 Traffic Stats

Traffic Stats is a utility written in Go that mines metrics from Traffic Monitor's JSON APIs and stores the data locally in Redis for a short period of time. This data is inherently transient, rolls frequently, and is volatile due to the default in-memory nature of Redis. The transient nature of the data is acceptable, as this system's purpose is to land data in Redis for other tools to consume.

Once in Redis, the data can be extracted and prepared to be sent elsewhere for long term storage. Any number of Traffic Stats instances may run on a given CDN to collect metrics from Traffic Monitor, however, redundancy and integration with a long term metrics storage system is implementation dependent. Traffic Stats does not influence overall CDN operation, but is required in order to display charts in Traffic Operations.

2.1.6 Traffic Portal

Traffic Portal is a user interface for CDN tenants to view performance, and in most cases, change settings of their delivery services. Traffic Portal is an Angular JS application written against the Traffic Ops APIs.

Note: The Traffic Portal is not being released to Open Source in the initial release.

2.1.7 Traffic Server

The caches in a Traffic Control CDN are servers running the Apache Traffic Server software. See ATS documentation for more information. Caches in a Traffic Control CDN are deployed in cache groups.

Cache Group

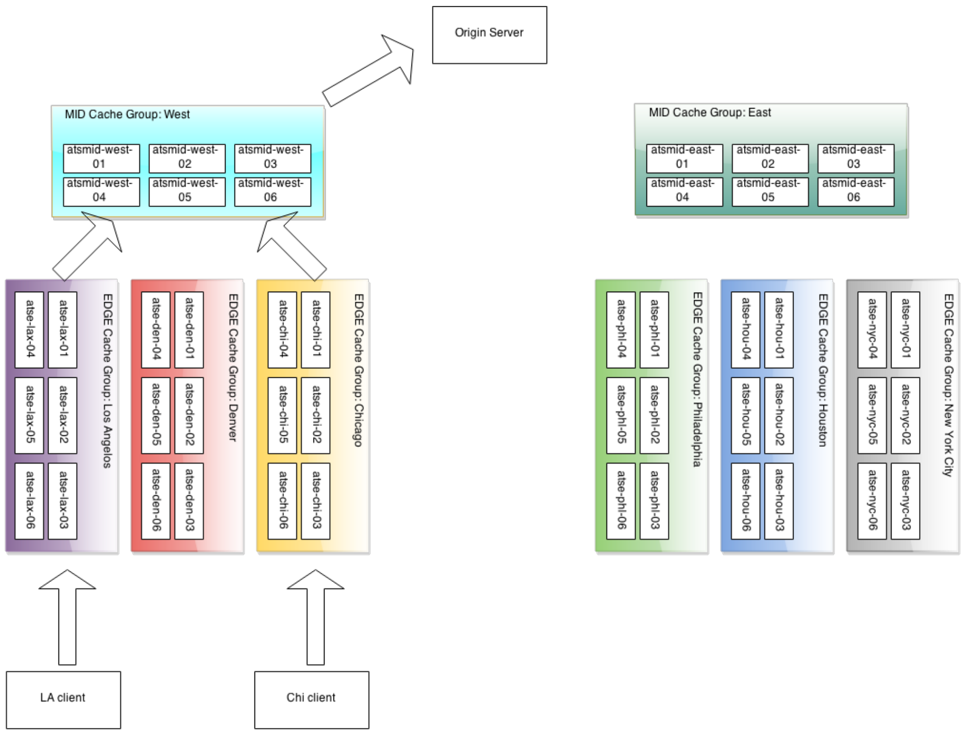
A cache group is a logical group of caches that Traffic Router tries to use as a combined cache. Traffic Router treats all servers in a cache group as though they are in the same physical location, though they are in fact only in the same region (network). A cache group has one single set of geographical coordinates even if the caches that make up the cache group are in different physical locations. The caches in a cache group are not aware of the other caches in the group - there is no clustering software or communications between caches in a cache group.

There are two types of cache groups: EDGE and MID. Traffic Control is a two tier system, where the clients get directed to the EDGE cache group. On cache miss, the cache in the EDGE cache group obtains content from a MID cache group, rather than the origin, which is shared with multiple EDGEs. EDGE cache groups are configured to have one single parent cache group.

Note: Often the EDGE to MID relationship is based on network distance, and does not necessarily match the geographic distance.

A cache group serves a particular part of the network as defined in the coverage zone file. See *The Coverage Zone File and ASN Table*.

Consider the example CDN below:



There are two MID tier cache groups, each assigned with three EDGEs. The lax, den and chi EDGE locations are configured with the West MID as their parent, and the nyc, phl, and hou EDGEs, are configured with the East MID as their parent. On a cache miss, the EDGEs use their assigned parent.

All caches (and other servers) are assigned a Profile in Traffic Ops.

Profile

A Profile is a set of configuration settings and parameters, applied to a server. For a typical cache there are hundreds of configuration settings to apply. The Traffic Ops parameter view contains the defined settings, and bundled into groups using Profiles. Traffic Ops allows for duplication, comparison, import and export of Profiles.

2.1.8 Traffic Vault

Traffic Vault is a keystore used for storing the following types of information:

- **SSL Certificates**
  - Private Key
  - CRT

    - CSR

- **DNSSEC Keys**
    - **Key Signing Key**
        - private key
        - public key
    - **Zone Signing Key**
        - private key
        - public key
- URL Signing Keys

As the name suggests, Traffic Vault is meant to be a "vault" of private keys that only certain users are allowed to access. In order to create, add, and retrieve keys a user must have admin privileges. Keys can be created via the Traffic Ops UI, but they can only be retrieved via the Traffic Ops API. The keystore used by Traffic Vault is Riak. Traffic ops accesses Riak via https on port 8088. Traffic ops uses Riak's rest API with username/password authentication. Information on the API can be found here.

CHAPTER 3

Administrator's Guide

How to deploy and manage a Traffic Control CDN.

3.1 Administrator's Guide

Traffic Control is distributed in source form for the developer, but also as a binary package. This guide details how to install and configure a Traffic Control CDN using the binary packages, as well as how to perform common operations running a CDN.

3.1.1 Installing Traffic Ops

System Requirements

The user must have the following for a successful install:

- CentOS 6
- 4 vCPUs
- 32GB RAM
- 20 GB disk space
- YUM repository with minimally the following dependecies avaliable
  - apr 1.3.9-5
  - apr-util 1.3.9-3
  - apr-util-ldap 1.3.9-3
  - expat-devel 2.0.1-11
  - genisoimage 1.1.9-12
  - httpd 2.2.15

  - httpd-tools 2.2.15
  - libpcap-devel 14:1.4
  - mod_ssl 1:2.2.15-29
  - mysql 5.1.71
  - autoconf 2.63-5.1.
  - automake 1.11.1-4
  - gcc 4.4.7-4
  - gettext 0.17-16
  - libcurl-devel 7.19.7-37
  - libtool 2.2.6-15.5
  - mysql-devel 5.1.73-3
  - perl-CPAN 1.9402-136
  - libcurl 7.19.7-37
  - openssl 1.0.1e-30
  - cloog-ppl 0.15.7-1.2
  - cpp 4.4.7-4
  - cvs 1.11.23-16
  - libgomp 4.4.7-4
  - libidn-devel 1.18-2
  - m4 1.4.13-5
  - mpfr 2.4.1-6
  - perl-Digest-SHA 1:5.47-136
  - ppl 0.10.2-11
  - curl 7.19.7-37
  - openssl-devel 1.0.1e-30
- Access to The Comprehensive Perl Archive Network (CPAN)

Note: The above versions are known to work on CentOS 6.5. Higher versions may work.

Note: Although Traffic Ops supports both MySQL and Postgres as a database, support for MySQL is more mature and better tested. It is best to use MySQL when first getting started, and the rest of this quide assumes MySQL as the database.

Navigating the Install

To begin the install:

1. Install Traffipc Ops: `sudo yum install traffic_ops`

2. After installation of Traffic Ops rpm enter the following command: `sudo /opt/traffic_ops/install/bin/postinstall`

   Example output:

```
trafficops-vm # /opt/traffic_ops/install/bin/postinstall

This script will build and package the required Traffic Ops perl modules.
In order to complete this operation, Development tools such as the gcc
compiler must be installed on this machine.

Hit ENTER to continue:
```

   The first thing the post install will do is install additional packages needed from the yum repo.

   Ater that, it will automatically proceed to installing the required Perl packages from CPAN.

   Note: Especially when installing Traffic Ops for the first time on a system this can take a long time, since many dependencies for the Mojolicous application need to be downloaded. Expect 30 minutes.

   If there are any prompts in this phase, please just answer with the defaults (some CPAN installs can prompt for install questions).

   When this phase is complete, you will see:

```
...
Successfully installed Test-Differences-0.63
Successfully installed DBIx-Class-Schema-Loader-0.07042
Successfully installed Time-HiRes-1.9726 (upgraded from 1.9719)
Successfully installed Mojolicious-Plugin-Authentication-1.26
113 distributions installed
Complete! Modules were installed into /opt/traffic_ops/app/local
Linking perl libraries...
Installing perl scripts


This script will initialize the Traffic Ops database.
Please enter the following information in order to completely
configure the Traffic Ops mysql database.


Database type [mysql]:
```

   The next phase of the install will ask you about the local environment for your CDN.

   Note: before proceeding to this step, the database has to have at least a root password, and needs to be started. When using mysql, please type `service mysqld start` as root in another terminal and follow the instructions on the screen to set the root passwd.

   Note: CentOS files note.

   Example output:

```
Database type [mysql]:
Database name [traffic_ops_db]:
Database server hostname IP or FQDN [localhost]:
Database port number [3306]:
Traffic Ops database user [traffic_ops]:
Password for traffic_ops:
Re-Enter password for traffic_ops:

Error: passwords do not match, try again.

Password for traffic_ops:
Re-Enter password for traffic_ops:

Database server root (admin) user name [root]:
Database server root password:
Database Type: mysql
Database Name: traffic_ops_db
Hostname: localhost
Port: 3306
Database User: traffic_ops
Is the above information correct (y/n) [n]:  y

The database properties have been saved to /opt/traffic_ops/app/conf/
↪production/database.conf

  The database configuration has been saved.  Now we need to set some custom
  fields that are necessary for the CDN to function correctly.


Traffic Ops url [https://localhost]:  https://traffic-ops.kabletown.net
Human-readable CDN Name.  (No whitespace, please) [kabletown_cdn]:
DNS sub-domain for which your CDN is authoritative [cdn1.kabletown.net]:
Fully qualified name of your CentOS 6.5 ISO kickstart tar file, or 'na' to␣
↪skip and add files later [/var/cache/centos65.tgz]:  na
Fully qualified location to store your ISO kickstart files [/var/www/files]:

Traffic Ops URL: https://traffic-ops.kabletown.net
Traffic Ops Info URL: https://traffic-ops.kabletown.net/info
Domainname: cdn1.kabletown.net
CDN Name: kabletown_cdn
GeoLocation Polling URL: https://traffic-ops.kabletown.net/routing/GeoIP2-
↪City.mmdb.gz
CoverageZone Polling URL: https://traffic-ops.kabletown.net/routing/coverage-
↪zone.json

Is the above information correct (y/n) [n]:  y
Parameter information has been saved to /opt/traffic_ops/install/data/json/
↪parameters.json


Adding an administration user to the Traffic Ops database.

Administration username for Traffic Ops:  admin
Password for the admin user admin:
Verify the password for admin:
Do you wish to create an ldap configuration for access to traffic ops [y/n] ?
↪ [n]:  n
```

(continues on next page)

(continued from previous page)

```
creating database
Creating database...
Creating user...
Flushing privileges...
setting up database
Executing 'drop database traffic_ops_db'
Executing 'create database traffic_ops_db'
Creating database tables...
Migrating database...
goose: migrating db environment 'production', current version: 0, target:␣
↪20150316100000
OK    20141222103718_extension.sql
OK    20150108100000_add_job_deliveryservice.sql
OK    20150205100000_cg_location.sql
OK    20150209100000_cran_to_asn.sql
OK    20150210100000_ds_keyinfo.sql
OK    20150304100000_add_ip6_ds_routing.sql
OK    20150310100000_add_bg_fetch.sql
OK    20150316100000_move_hdr_rw.sql
Seeding database...
Database initialization succeeded.
seeding profile data...
name EDGE1 description Edge 1
name TR1 description Traffic Router 1
name TM1 description Traffic Monitor 1
name MID1 description Mid 1
seeding parameter data...
```

Explanation of the information that needs to be provided:

Field	Description
Database type	mysql or postgres
Database name	The name of the database Traffic Ops uses to store the configuration information
Database server hostname IP or FQDN	The hostname of the database server
Database port number	The database port number
Traffic Ops database user	The username Traffic Ops will use to read/write from the database
password for traffic ops	The passwdord for the above database user
Database server root (admin) user name	Priviledged database user that has permission to create the database and user for Traffic Ops
Database server root (admin) user password	The password for the above priviledged database user
Traffic Ops url	The URL to connect to this instance of Traffic Ops, usually https://<traffic ops host FQDN>/
Human-readable CDN Name	The name of the first CDN traffic Ops will be managing
DNS sub-domain for which your CDN is authoritative	The DNS domain that will be delegated to this Traffic Control CDN
name of your CentOS 6.5 ISO kickstart tar file	See *Creating the CentOS Kickstart File*
Administration username for Traffic Ops	The Administration (highest privilege) Traffic Ops user to create; use this user to login for the first time and create other users
Password for the admin user	The passwd for the above user

The postinstall script will now seed the database with some inital configuration settings for the CDN and the servers in the CDN.

The next phase is the download of the geo location database and configuration of information needed for SSL certificates.

Example output:

```
Downloading MaxMind data.
--2015-04-14 02:14:32--  http://geolite.maxmind.com/download/geoip/database/
↪GeoLite2-City.mmdb.gz
Resolving geolite.maxmind.com... 141.101.115.190, 141.101.114.190,␣
↪2400:cb00:2048:1::8d65:73be, ...
Connecting to geolite.maxmind.com|141.101.115.190|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17633433 (17M) [application/octet-stream]
Saving to: “GeoLite2-City.mmdb.gz”

100
↪%[==========================================================================================
↪] 17,633,433  7.03M/s   in 2.4s

2015-04-14 02:14:35 (7.03 MB/s) - “GeoLite2-City.mmdb.gz” saved [17633433/
↪17633433]
```

(continues on next page)

(continued from previous page)

```
Copying coverage zone file to public dir.

Installing SSL Certificates.

  We're now running a script to generate a self signed X509 SSL certificate.
  When prompted to enter a pass phrase, just enter 'pass' each time.  The
  pass phrase will be stripped from the private key before installation.

  When prompted to enter a 'challenge password', just hit the ENTER key.

  The remaining enformation Country, State, Locality, etc... are required to
  generate a properly formatted SSL certificate.

Hit Enter when you are ready to continue:
Postinstall SSL Certificate Creation.

Generating an RSA Private Server Key.

Generating RSA private key, 1024 bit long modulus
..........................++++++
....................++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:

The server key has been generated.

Creating a Certificate Signing Request (CSR)

Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:CO
Locality Name (eg, city) [Default City]:Denver
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:pass
An optional company name []:

The Certificate Signing Request has been generated.
Removing the pass phrase from the server key.
Enter pass phrase for server.key.orig:
writing RSA key

The pass phrase has been removed from the server key.
```

(continues on next page)

(continued from previous page)

```
Generating a Self-signed certificate.
Signature ok
subject=/C=US/ST=CO/L=Denver/O=Default Company Ltd
Getting Private key

A server key and self signed certificate has been generated.

Installing the server key and server certificate.

The private key has been installed.

Installing the self signed certificate.

Saving the self signed csr.

  The self signed certificate has now been installed.

  You may obtain a certificate signed by a Certificate Authority using the
  server.csr file saved in the current directory.  Once you have obtained
  a signed certificate, copy it to /etc/pki/tls/certs/localhost.crt and
  restart Traffic Ops.



SSL Certificates have been installed.

Starting Traffic Ops.

Starting Traffic Ops

Subroutine TrafficOps::has redefined at /opt/traffic_ops/app/local/lib/perl5/
↪Mojo/Base.pm line 38.
Subroutine TrafficOps::has redefined at /opt/traffic_ops/app/local/lib/perl5/
↪Mojo/Base.pm line 38.
Loading config from /opt/traffic_ops/app/conf/cdn.conf
Reading log4perl config from /opt/traffic_ops/app/conf/production/log4perl.
↪conf
Starting hot deployment for Hypnotoad server 32192.

Waiting for Traffic Ops to start.


Shutdown Traffic Ops [y/n] [n]:  n

To start Traffic Ops:  service traffic_ops start
To stop Traffic Ops:   service traffic_ops stop

traffic_ops #
```

Traffic Ops is now installed!

3. Download the web dependencies (this will be added to the installer in the future):

```
traffic_ops # pwd
/opt/traffic_ops/install/bin
traffic_ops # ./download_web_deps
Finished curling https://cdn.datatables.net/1.10.4/js/jquery.dataTables.min.js |␣
↪size is: 78746
```

(continues on next page)

(continued from previous page)

```
Finished curling https://github.com/fancyapps/fancyBox/zipball/v2.1.5 | size is:␣
↪541026
Finished curling http://www.flotcharts.org/downloads/flot-0.8.3.zip | size is:␣
↪649913
Finished curling https://github.com/krzysu/flot.tooltip/releases/download/0.8.4/
↪jquery.flot.tooltip-0.8.4.zip | size is: 7669
Finished curling https://gflot.googlecode.com/svn-history/r154/trunk/flot/jquery.
↪flot.axislabels.js | size is: 17321
Finished curling https://github.com/alpixel/jMenu/archive/master.zip | size is:␣
↪41836
Finished curling https://code.jquery.com/jquery-1.11.2.min.js | size is: 95931
Finished curling https://code.jquery.com/ui/1.11.4/jquery-ui.min.js | size is:␣
↪240427
Finished curling https://code.jquery.com/ui/1.7.3/themes/dark-hive/jquery-ui.css␣
↪| size is: 27499
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_flat_30_cccccc_40x100.png | size is: 180
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_flat_50_5c5c5c_40x100.png | size is: 180
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_glass_40_ffc73d_1x400.png | size is: 131
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_highlight-hard_20_0972a5_1x100.png | size is: 114
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_highlight-soft_33_003147_1x100.png | size is: 127
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_highlight-soft_35_222222_1x100.png | size is: 113
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_highlight-soft_44_444444_1x100.png | size is: 117
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_highlight-soft_80_eeeeee_1x100.png | size is: 95
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-bg_loop_25_000000_21x21.png | size is: 235
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-icons_222222_256x240.png | size is: 4369
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-icons_4b8e0b_256x240.png | size is: 4369
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-icons_a83300_256x240.png | size is: 4369
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-icons_cccccc_256x240.png | size is: 4369
Finished curling http://jquery-ui.googlecode.com/svn/tags/1.7.3/themes/dark-hive/
↪images/ui-icons_ffffff_256x240.png | size is: 4369
Finished curling https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.
↪js | size is: 35951
Output file: ../../app/public/js/jquery.dataTables.min.js does not exist, putting␣
↪into place.
Making dir: ../../app/public/js/fancybox/
Output file: ../../app/public/js/fancybox//jquery.fancybox-buttons.js does not␣
↪exist. Putting file from zip into place.
Output file: ../../app/public/js/fancybox//fancybox_loading@2x.gif does not exist.
↪ Putting file from zip into place.
Output file: ../../app/public/js/fancybox//fancybox_loading.gif does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/fancybox//fancybox_buttons.png does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/fancybox//jquery.fancybox-thumbs.js does not␣
↪exist. Putting file from zip into place.
```

(continues on next page)

(continued from previous page)

```
Output file: ../../app/public/js/fancybox//jquery.fancybox-buttons.css does not␣
↪exist. Putting file from zip into place.
Output file: ../../app/public/js/fancybox//jquery.fancybox-thumbs.css does not␣
↪exist. Putting file from zip into place.
Output file: ../../app/public/js/fancybox//fancybox_sprite@2x.png does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/fancybox//jquery.fancybox.css does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/fancybox//jquery.fancybox-media.js does not␣
↪exist. Putting file from zip into place.
Output file: ../../app/public/js/fancybox//fancybox_overlay.png does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/fancybox//fancybox_sprite.png does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/fancybox//jquery.fancybox.js does not exist.␣
↪Putting file from zip into place.
Making dir: ../../app/public/js/flot/
Output file: ../../app/public/js/flot//jquery.flot.min.js does not exist. Putting␣
↪file from zip into place.
Output file: ../../app/public/js/flot//jquery.flot.selection.js does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/flot//jquery.flot.stack.js does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/flot//jquery.flot.time.js does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/flot//jquery.flot.tooltip.js does not exist.␣
↪Putting file from zip into place.
Output file: ../../app/public/js/flot/jquery.flot.axislabels.js does not exist,␣
↪putting into place.
Output file: ../../app/public/js//jMenu.jquery.min.js does not exist. Putting␣
↪file from zip into place.
Output file: ../../app/public/css//jmenu.css does not exist. Putting file from␣
↪zip into place.
Output file: ../../app/public/js/jquery-1.11.2.min.js does not exist, putting␣
↪into place.
Output file: ../../app/public/js/jquery-ui.min.js does not exist, putting into␣
↪place.
Output file: ../../app/public/css/jquery-ui.css does not exist, putting into␣
↪place.
Making dir: ../../app/public/css/images/
Output file: ../../app/public/css/images/ui-bg_flat_30_cccccc_40x100.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_flat_50_5c5c5c_40x100.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_glass_40_ffc73d_1x400.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_highlight-hard_20_0972a5_1x100.png␣
↪does not exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_highlight-soft_33_003147_1x100.png␣
↪does not exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_highlight-soft_35_222222_1x100.png␣
↪does not exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_highlight-soft_44_444444_1x100.png␣
↪does not exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_highlight-soft_80_eeeeee_1x100.png␣
↪does not exist, putting into place.
Output file: ../../app/public/css/images/ui-bg_loop_25_000000_21x21.png does not␣
↪exist, putting into place.
```

(continues on next page)

(continued from previous page)

```
Output file: ../../app/public/css/images/ui-icons_222222_256x240.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-icons_4b8e0b_256x240.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-icons_a83300_256x240.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-icons_cccccc_256x240.png does not␣
↪exist, putting into place.
Output file: ../../app/public/css/images/ui-icons_ffffff_256x240.png does not␣
↪exist, putting into place.
Output file: ../../app/public/js/bootstrap.min.js does not exist, putting into␣
↪place.
traffic_ops #
```

Upgrading Traffic Ops

To upgrade:

1. Enter the following command:`service traffic_ops stop`
2. Enter the following command:`yum upgrade traffic_ops`
3. See *Installing Traffic Ops* to run the post install.
4. Enter the following command:`service traffic_ops start`

3.1.2 Configuring Traffic Ops

Follow the steps below to configure the newly installed Traffic Ops Instance.

Installing the SSL Cert

By default, Traffic Ops runs as an SSL web server, and a certificate needs to be installed. TBD.

Content Delivery Networks

Parameters an profiles

Many of the settings for the different servers in a Traffic Control CDN are controlled by parameters in the parameter view of Traffic Ops. Parameters are grouped in profiles and profiles are assigned to a server. For a typical cache there are hundreds of configuration settings to apply. The Traffic Ops parameter view contains the defined settings. To make life easier, Traffic Ops allows for duplication, comparison, import and export of Profiles. Traffic Ops also has a "Global profile" - the parameters in this profile are going to be applied to all servers in the Traffic Ops instance, or apply to Traffic Ops themselves. These parameters are:

Name	Config file	Value
tm.url	global	The URL where this Traffic Ops instance is being served from.
tm.toolname	global	The name of the Traffic Ops tool. Usually "Traffic Ops". Used in the About screen and in the comments headers of the files generated.
tm.infourl	global	This is the "for more information go here" URL, which is visible in the About page.
tm.logourl	global	This is the URL of the logo for Traffic Ops and can be relative if the logo is under traffic_ops/app/public.
tm.instance_name	global	The name of the Traffic Ops instance. Can be used when multiple instances are active. Visible in the About page.
tm.traffic_mon_fwd_proxy	global	When collecting stats from Traffic Monitor, Traffic Ops uses this forward proxy to pull the stats through. This can be any of the MID tier caches, or a forward cache specifically deployed for this purpose. Setting this variable can significantly lighten the load on the Traffic Monitor system and it is recommended to set this parameter on a production system.
geolocation.polling.url	CRConfig.json	The location to get the GeoLiteCity database from.
geolocation6.polling.url	CRConfig.json	The location to get the IPv6 GeoLiteCity database from.

These parameters should be set to reflect the local environment.

After running the postinstall script, Traffic Ops has the following profiles pre-loaded:

Name	Description
EDGE1	The profile to be applied to the latest supported version of ATS, when running as an EDGE cache
TR1	The profile to be applied to the latest version of Traffic Router
TM1	The profile to be applied to the latest version of Traffic Monitor
MID1	The profile to be applied to the latest supported version of ATS, when running as an MID cache
RIAK_ALL	Riak profile for all CDNs to be applied to the Traffic Vault servers

..Note:: The Traffic Server profiles contain some information that is specific to the hardware being used (most notably the disk configuration), so some parameters will have to be changed to reflect your configuration. Future releases of Traffic Control will separate the hardware and software profiles so it is easier to "mix-and-match" different hardware configurations.

Below is a list of cache parameters that are likely to need changes from the default profiles shipped with Traffic Ops:

Name	Config file	Description
allow_ip	astats.config	This is a comma separated list of IPv4 CIDR blocks that will have access to the astats statistics on the caches. The Traffic Monitor IP addresses have to be included in this, if they are using IPv4 to monitor the caches.
allow_ip6	astats.config	This is a comma separated list of IPv6 CIDR blocks that will have access to the astats statistics on the caches. The Traffic Monitor IP addresses have to be included in this, if they are using IPv6 to monitor the caches.
Drive_Prefix	storage.config	JvD/Jeff to supply blurb
Drive_Letters	storage.config	JvD/Jeff to supply blurb
purge_allow_ip	ip_allow.config	The IP address that is allowed to "purge" content on the CDN through regex_revalidate
health.threshold.loadavg	rascal.properties	The Unix load average at which Traffic Router will stop sending traffic to this cache
health.threshold.\ availableBandwidthInKbps	rascal.properties	The amount of bandwidth that Traffic Router will try to keep available on the cache. For example: "">1500000" means stop sending new traffic to this cache when traffic is at 8.5Gbps on a 10Gbps interface.

Regions, Locations and Cache Groups

All servers have to have a *location*, which is their physical location. Each location is part of a *region*, and each region is part of a *division*. For Example, `Denver` could be a location in the `Mile High` region and that region could be part of the `West` division. Enter your divisions first in *Misc->Divisions*, then enter the regions in *Misc->Regions*, referencing the divisions entered, and finally, enter the physical locations in *Misc->Locations*, referencing the regions entered.

All servers also have to be part of a *cache group*. A cache group is a logical grouping of caches, that don't have to be in the same physical location (in fact, usually a cache group is spread across minimally 2 physical locations for redundancy purposes), but share geo coordinates for content routing purposes. JvD to add more.

Creating the CentOS Kickstart File

The kickstart file is a text file, containing a list of items, each identified by a keyword. You can create it by using the Kickstart Configurator application, or writing it from scratch. The Red Hat Enterprise Linux installation program also creates a sample kickstart file based on the options that you selected during installation. It is written to the file `/root/anaconda-ks.cfg`. This file is editable using most text editors that can save files as ASCII text.

To generate ISO, the CentOS Kickstart is necessary:

1. Create a kickstart file.
2. Create a boot media with the kickstart file or make the kickstart file available on the network.
3. Make the installation tree available.
4. Start the kickstart installation.

Create a ks.src file in the root of the selection location. See the example below:

```
mkdir newdir
cd newdir/
```

(continues on next page)

(continued from previous page)

```
cp -r ../centos65/* .
vim ks.src
vim isolinux/isolinux.cfg
cd vim osversions.cfg
vim osversions.cfg
```

This is a standard kickstart formatted file that the generate ISO process uses to create the kickstart (ks.cfg) file for the install. The generate ISO process uses the ks.src, overwriting any information set in the Generate ISO tab in Traffic Ops, creating ks.cfg.

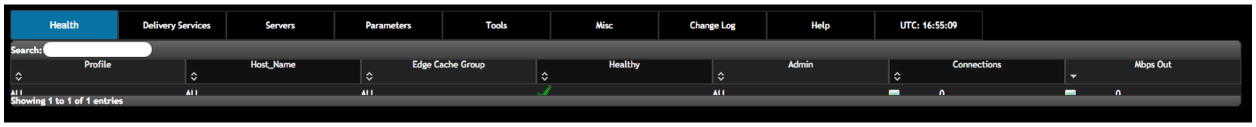
Note: Streamline your install folder for under 1GB, which assists in creating a CD.

See also:

For in-depth instructions, please see Kickstart Installation

3.1.3 Using Traffic Ops

The Traffic Ops Menu



The following tabs are available in the menu at the top of the Traffic Ops user interface.

- **Health**

  Information on the health of the system. Hover over this tab to get to the following options:

Option	Description
Table View	A real time view into the main performance indicators of the CDNs managed by Traffic Control. This view is sourced directly by the Traffic Monitor data and is updated every 10 seconds. This is the default screen of Traffic Ops. See *The Health Table* for details.
Graph View	A real graphical time view into the main performance indicators of the CDNs managed by Traffic Control. This view is sourced by the Traffic Monitor data and is updated every 10 seconds. On loading, this screen will show a history of 24 hours of data from Traffic Stats See *Graph View* for details.
Server Checks	A table showing the results of the periodic check extension scripts that are run. See *Server Checks*
Daily Summary	A graph displaying the daily peaks of bandwidth, overall bytes served per day, and overall bytes served since initial installation per CDN.

- **Delivery Services**

  The main Delivery Service table. This is where you Create/Read/Update/Delete Delivery Services of all types. There are currently no sub menus for this tab.

- **Servers**

The main Servers table. This is where you Create/Read/Update/Delete servers of all types. Click the main tab to get to the main table, and hover over to get these sub options:

Option	Description
Upload Server CSV	Bulk add of servers from a csv file. See *Bulk Upload Server*

- **Parameters**

  Parameters and Profiles can be edited here. Hover over the tab to get the following options:

Option	Description
Global Profile	The table of global parameters. See *Parameters an profiles*. This is where you Create/Read/Update/Delete parameters in the Global profile
All Cache Groups	The table of all parameters *that are assgined to a cachegroup* - this may be slow to pull up, as there can be thousands of parameters.
All Profiles	The table of all parameters *that are assgined to a profile* - this may be slow to pull up, as there can be thousands of parameters.
Select Profile	Select the parameter list by profile first, then get a table of just the parameters for that profile.
Orphaned Parameters	A table of parameters that are not associated to any profile of cache group. These parameters either should be deleted or associated with a profile of cache group.

- **Tools**

  Tools for working with Traffic Ops and it's servers. Hover over this tab to get the following options:

Option	Description
Generate ISO	Generate a bootable image for any of the servers in the Servers table (or any server for that matter). See *Generate ISO*
Queue Updates	Send Updates to the caches. See *Queue Updates and Snapshot CRConfig*
DB Dump	Backup the Database to a .sql file.
Snapshot CRConfig	Send updates to the Traffic Monitor / Traffic Router servers. See *Queue Updates and Snapshot CRConfig*
Invalidate Content	Invalidate or purge content from the CDN. See *Invalidate Content*
Generate DNSSEC keys	Neuman?

- **Misc**

  Miscellaneous editing options. Hover over this tab to get the following options:

Option	Description
Cache Groups	Create/Read/Update/Delete cache groups
Users	Create/Read/Update/Delete users
Profiles	Create/Read/Update/Delete profiles. See *Parameters and Profiles*
Networks(ASNs)	Create/Read/Update/Delete Autonomous System Numbers See *The Coverage Zone File and ASN Table*
Hardware	Get detailed hardware information (note: this should be moved to a Traffic Ops Extension)
Data Types	Create/Read/Update/Delete data types
Divisions	Create/Read/Update/Delete divisions
Regions	Create/Read/Update/Delete regions
Physical Locations	Create/Read/Update/Delete locations

- **ChangeLog**

  The Changelog table displays the changes that are being made to the Traffic Ops database through the Traffic Ops user interface. This tab will show the number of changes since you last visited this tab in (brackets) since the last time you visited this tab. There are currently no sub menus for this tab.

- **Help**

  Help for Traffic Ops and Traffic Control. Hover over this tab to get the following options:

Option	Description
About	Traffic Ops information, such as version, database information, etc
Release Notes	Release notes for the most recent releases of Traffic Ops
Logout	Logout from Traffic Ops

Health

The Health Table

The Health table is the default landing screen for Traffic Ops, it displays the status of the EDGE caches in a table form directly from Traffic Monitor (bypassing Traffic Stats), sorted by Mbps Out. The columns in this table are:

- **Profile**: the Profile of this server or ALL, meaning this row shows data for multiple servers, and the row shows the sum of all values.
- **Host Name**: the host name of the server or ALL, meaning this row shows data for multiple servers, and the row shows the sum of all values.
- **Edge Cache Group**: the edge cache group short name or ALL, meaning this row shows data for multiple servers, and the row shows the sum of all values.
- **Healthy**: indicates if this cache is healthy according to the Health Protocol. A row with ALL in any of the columns will always show a ✓, this column is valid only for individual EDGE caches.
- **Admin**: shows the administrative status of the server.
- **Connections**: the number of connections this cache (or group of caches) has open (`ats.proxy.process.http.current_client_connections` from ATS).
- **Mbps Out**: the bandwidth being served out if this cache (or group of caches)

Since the top line has ALL, ALL, ALL, it shows the total connections and bandwidth for all caches managed by this instance of Traffic Ops.

Graph View

The Graph View shows a live view of the last 24 hours of bits per seconds served and open connections at the edge in a graph. This data is sourced from Traffic Stats. If there are 2 CDNs configured, this view will show the statistis for both, and the graphs are stacked. On the left-hand side, the totals and immediate values as well as the percentage of total possible capacity are displayed. This view is update every 10 seconds.

Server Checks

Server Checks are ..

Daily Summary

Server

This view shows a table of all the servers in Traffic Ops. The table columns show the most important details of the server. The **IPAddrr** column is clickable to launch an `ssh://` link to this server. The icon will link to a Traffic Stats graph of this server for caches, and the will link to the server status pages for other server types.

Server Types

These are the types of servers that can be managed in Traffic Ops:

Name	Description
EDGE	Edge Cache
MID	Mid Tier Cache
ORG	Origin
CCR	Comcast Content Router
RASCAL	Rascal health polling & reporting
REDIS	Redis stats gateway (will be obsolete soon)
TOOLS_SERVER	Ops hosts for managment
RIAK	Riak keystore
SPLUNK	SPLUNK indexer search head etc
TRAFFIC_STATS	traffic_stats server
INFLUXDB	influxDb server

Bulk Upload Server

Delivery Service

The fields in the Delivery Service view are:

Name	Description
XML ID	A unique string that identifies this delivery service.
Content Routing Type	The type of content routing this delivery service will use. See *Delivery Service Types*.
Protocol	The protocol to serve this delivery service to the clients with: • 0 http • 1 https • 2 both http and https
DSCP Tag	The DSCP value to mark IP packets to the client with.
Signed URLs	Use Signed URLs? See *Token Based Authentication*.
Query String Handling	How to treat query strings: • 0 use in cache key and hand up to origin -this means each unique query string Is treated as a unique URL. • 1 Do not use in cache key, but pass up to origin - this means a 2 URLs that are the same except for the query string will match, and cache HIT, while the origin still sees original query string in the request. • 2 Drop at edge - this means a 2 URLs that are the same except for the query string will match, and cache HIT, while the origin will not see original query string in the request.
Geo Limit?	Some services are intended to be limited by geography. The possible settings are are: • None - Do not limit by geography. • CZF only - If the requesting IP is not in the Coverage Zone File, do not serve the request. • CZF + US - If the requesting IP is not in the Coverage Zone File or not in the United States, do not serve the request.
Bypass FQDN	(for HTTP routed delivery services only) This is the FQDN Traffic Router will redirect to (with the same path) when the max Bps or Max Tps for this deliveryservice are exceeded.
Bypass Ipv4	(For DNS routed delivery services only) This is the address to respond to A requests with when the the max Bps or Max Tps for this delivery service are exceeded.
Bypass IPv6	(For DNS routed delivery services only) This is the address to respond to AAAA requests with when the the max Bps or Max Tps for this delivery service are exceeded.
IPv6 Routing Enabled?	When set to yes, the Traffic Router will respond to AAAA DNS requests for the tr. and edge. names of this delivery service. Otherwise, only A records will be served.

Continued on next page

Table 1 – continued from previous page

Name	Description
Range Request Handling	(experimental) How to treat range requests: • 0 Do not cache (ranges requested from files taht are already cached due to a non range request will be a HIT) • 1 Use the background_fetch plugin. • 2 Use the cache_range_requests plugin.
Delivery Service DNS TTL	The Time To Live on the DNS record for the Traffic Router A and AAAA records (`tr.<deliveryservice>.<cdn-domain>`) for a HTTP delivery service *or* for the A and AAAAA records of the edge name (`edge.<deliveryservice>.<cdn-domain>`).
Origin Server Base URL	The Origin Server's base URL. This includes the protocol (http or https). Example: `http://movies.origin.com`
Use Multi Site Origin Feature	Enable the Multi Site Origin feature for this delivery service. See *Multi Site Origin*
CCR profile	The Traffic Router profile for this delivery service. See *CCR Profile or Traffic Router Profile*.
Maximum Bits per Second allowed globally	The maximum bits per second this delivery service can serve across all EDGE caches before traffic will be diverted to the bypass destination. For a DNS delivery service, the Bypass Ipv4 or Ipv6 will be used (depending on whether this was a A or AAAA request), and for HTTP delivery services the Bypass FQDN will be used.
Maximum Transactions per Second allowed globally	The maximum transactions per se this delivery service can serve across all EDGE caches before traffic will be diverted to the bypass destination. For a DNS delivery service, the Bypass Ipv4 or Ipv6 will be used (depending on whether this was a A or AAAA request), and for HTTP delivery services the Bypass FQDN will be used.
Geo Miss Default Latitude	Default Latitude for this delivery service. When client localization fails for bot Coverage Zone and Geo Lookup, this the client will be routed as if it was at this lat.
Geo Miss Default Longitude	Default Longitude for this delivery service. When client localization fails for bot Coverage Zone and Geo Lookup, this the client will be routed as if it was at this long.
Edge Header Rewrite Rules	Header Rewrite rules to apply for this delivery service at the EDGE tier. See *Header Rewrite Options and DSCP*.[1]
Mid Header Rewrite Rules	Header Rewrite rules to apply for this delivery service at the MID tier. See *Header Rewrite Options and DSCP*.[1]
Regex Remap Expression	Regex Remap rule to apply to this delivery service at the Edge tier. See ATS documentation on regex_remap.[1]
Cache URL expression	Cache URL rule to apply to this delivery service. See ATS documentation on cacheurl.[1]

Continued on next page

Table 1 – continued from previous page

Name	Description
Raw remap text	For HTTP and DNS deliveryservices, this will get added to the end of the remap line on the cache verbatim. For ANY_MAP deliveryservices this is the remap line.[1]
Long Description	Long description for this delivery service. To be consumed from the APIs by downstream tools (Portal).
Customer	Customer description for this delivery service. To be consumed from the APIs by downstream tools (Portal).
Service	Service description for this delivery service. To be consumed from the APIs by downstream tools (Portal).
Info URL	Info URL for this delivery service. To be consumed from the APIs by downstream tools (Portal).
Check Path	A path (ex: /crossdomain.xml) to verify the connection to the origin server with. This can be used by Check Extension scripts to do periodic health checks against the delivery service.
Origin Shield (Pipe Delimited String)	Experimental. Origin Shield string. See rl-org-shield
Active	When this is set to no Traffic Router will not serve DNS or HTTP responses for this delivery service.
Last Updated	(Read Only) The last time this delivery service was updated.
Number of edges assigned	(Read Only - change by clicking the **Server Assignments** button at the bottom) The number of EDGE caches assigned to this delivery service. See *Server Assignments*.
Number of static DNS entries	(Read Only - change by clicking the **Static DNS** button at the bottom) The number of static DNS entries for this delivery service. See *Static DNS Entries*.
Example delivery URL	(Read Only) An example of how the delivery URL may start. This could be multiple rows if multiple HOST_REGEXP entries have been entered.
Regular expressions for this delivery service	A subtable of the regular expressions to use when routing traffic for this delivery service. See *Delivery Service Regexp*.

Delivery Service Types

One of the most important settings when creating the delivery service is the selection of the delivery service *type*. This type determines the routing method and the primary storage for the delivery service.

[1] These fields are not validated by Traffic Ops to be correct syntactically, and can cause Traffic Server to not start if invalid. Please use with caution.

Name	Description
HTTP	HTTP Content Routing - The Traffic Router DNS auth server returns its own IP address on DNS queries, and the client gets redirected to a specific cache in the nearest cache group using HTTP 302. Use this for long sessions like HLS/HDS/Smooth live streaming, where a longer setup time is not a. problem.
DNS	DNS Content Routing - The Traffic Router DNS auth server returns an edge cache IP address to the client right away. The client will find the cache quickly but the Traffic Router can not route to a cache that already has this content in the cache group. Use this for smaller objects like web page images / objects.
HTTP_NO_CACHE	HTTP Content Routing, but the caches will not actually cache the content, they act as just proxies. The MID tier is bypassed.
HTTP_LIVE	HTTP Content routing, but where for “standard” HTTP content routing the objects are stored on disk, for this delivery service type the objects are stored on the RAM disks. Use this for linear TV. The MID tier is bypassed for this type.
HTTP_LIVE_NATNL	HTTP Content routing, same as HTTP_LIVE, but the MID tier is NOT bypassed.
DNS_LIVE_NATNL	DNS Content routing, ut where for “standard” DNS content routing the objects are stored on disk, for this delivery service type the objects are stored on the RAM disks. Use this for linear TV. The MID tier is NOT bypassed for this type.
DNS_LIVE	DNS Content routing, same as DNS_LIVE_NATIONAL, but the MID tier is bypassed.
ANY_MAP	ANY_MAP is not known to Traffic Router. For this deliveryservice, the “Raw remap text” field in the input form will be used as the remap line on the cache.

Note: Once created, the Traffic Ops user interface does not allow you to change the delivery service type; the drop down is greyed out. There are many things that can go wrong when changing the type, and it is safer to delete the delivery service, and recreate it.

Header Rewrite Options and DSCP

Most header manipulation and per-delivery service configuration overrides are done using the ATS Header Rewrite Plugin. Traffic Control allows you to enter header rewrite rules to be applied at the edge and at the mid level. The syntax used in Traffic Ops is the same as the one described in the ATS documentation, except for some special strings that will get replaced:

Traffic Ops Entry	Gets Replaced with
__RETURN__	A newline
__CACHE_IPV4__	The cache’s IPv4 address

The deliveryservice screen also allows you to set the DSCP value of traffic sent to the client. This setting also results in a header_rewrite rule to be generated and applied to at the edge.

Note: The DSCP setting in the UI is *only* for setting traffic towards the client, and gets applied *after* the initial TCP handshake is complete, and the HTTP request is received (before that the cache can’t determine what deliveryservice this request is for, and what DSCP to apply), so the DSCP feature can not be used for security settings - the TCP SYN-ACK is not going to be DSCP marked.

Token Based Authentication

Token based authentication or *signed URLs* is implemented using the Traffic Server `url_sig` plugin. To sign a URL at the signing portal take the full URL, without any query string, and add on a query string with the following parameters:

Client IP address The client IP address that this signature is valid for.

```
C=<client IP address>
```

Expiration The Expiration time (seconds since epoch) of this signature.

```
E=<expiration time in secs since unix epoch>
```

Algorithm The Algorithm used to create the signature. Only 1 (HMAC_SHA1) and 2 (HMAC_MD5) are supported at this time

```
A=<algorithm number>
```

Key index Index of the key used. This is the index of the key in the configuration file on the cache. The set of keys is a shared secret between the signing portal and the edge caches. There is one set of keys per reverse proxy domain (fqdn).

```
K=<key index used>
```

Parts Parts to use for the signature, always excluding the scheme (http://). parts0 = fqdn, parts1..x is the directory parts of the path, if there are more parts to the path than letters in the parts param, the last one is repeated for those. Examples:

> 1: use fqdn and all of URl path 0110: use part1 and part 2 of path only 01: use everything except the fqdn

```
P=<parts string (0's and 1's>
```

Signature The signature over the parts + the query string up to and including "S=".

```
S=<signature>
```

See also:

The url_sig README.

Generate URL Sig Keys

To generate a set of random signed url keys for this delivery service and store them in Traffic Vault, click the **Generate URL Sig Keys** button at the bottom of the delivery service details screen.

Multi Site Origin

Note: The Multi Site Origin feature is based upon a feature n ATS that has yet to be submitted to Traffic Server upstream, until it is, set this to 0.

Normally, the mid servers are not aware of any redundancy at the origin layer. With Multi Site Origin enabled this changes - Traffic Server (and Traffic Ops) are now made aware of the fact there are multiple origins, and can be configured to do more advanced failover and loadbalancing actions.

With This feature enabled, origin servers (or origin server VIP names for a site) are going to be entered as servers in to the Traiffic Ops UI. Server type is With This feature enabled, origin servers (or origin server VIP names for a site) are going to be entered as servers in to the Traiffic Ops UI. Server type is “”

Parameters in the Origin profile that influence this feature:

Name	Default	Description
CONFIG proxy.config.http.parent_proxy_routing_enable	INT 1	enable parent selection. This is a required setting.
CONFIG proxy.config.url_remap.remap_required	INT 1	required for parent selection.
CONFIG proxy.config.http.parent_proxy.per_parent_connect_attempts	INT 5	maximum of 5 connection attempts per parent (parent.config list) within a transaction.
CONFIG proxy.config.http.parent_proxy.total_connect_attempts	INT 10	maximum of 10 total connection attempts within a transaction.
CONFIG proxy.config.http.parent_origin.simple_retry_enabled	INT 1	enables simple retry.
CONFIG proxy.config.http.parent_origin.simple_retry_response_codes	STRING 404	the response code that invokes simple retry. May be a comman separated list of response codes.
CONFIG proxy.config.http.parent_origin.dead_server_retry_response_codes	STRING 503	the response code that invokes dead server retry. May be a comma separated list of response codes
CONFIG proxy.config.http.parent_origin.dead_server_retry_enabled	INT 1	enables dead server retry.
CONFIG proxy.config.diags.debug.enabled	INT 1	enable debugging for testing only

CCR Profile or Traffic Router Profile

Name	Config_file	Description
location	dns.zone	Location to store the DNS zone files in the local file system of Traffic Router.
location	http-log4j.properties	Location to find the log4j.properties file for Traffic Router.
location	dns-log4j.properties	Location to find the dns-log4j.properties file for Traffic Router.
location	geolocation.properties	Location to find the log4j.properties file for Traffic Router.
CDN_name	rascal-config.txt	The human readable name of the CDN for this profile.
CoverageZoneJsonURL	CRConfig.xml	The location (URL) to retrieve the coverage zone map file in JSON format from.
geolocation.polling.url	CRConfig.json	The location (URL) to retrieve the geo database file from.
geolocation.polling.interval	CRConfig.json	How often to refresh the coverage geo location database in ms
coveragezone.polling.interval	CRConfig.json	How often to refresh the coverage zone map in ms
coveragezone.polling.url	CRConfig.json	The location (URL) to retrieve the coverage zone map file in XML format from.
domain_name	CRConfig.json	The top level domain of this Traffic Router instance.
tld.ttls.AAAA	CRConfig.json	The Time To Live (TTL) the Traffic Router DNS Server will respond with on AAAA records.
tld.ttls.A	CRConfig.json	The TTL the Traffic Router DNS Server will respond with on A records.
tld.soa.expire	CRConfig.json	The value for the expire field the Traffic Router DNS Server will respond with on Start of Authority (SOA) records.
tld.soa.minimum	CRConfig.json	The value for the minimum field the Traffic Router DNS Server will respond with on SOA records.
tld.soa.admin	CRConfig.json	The DNS Start of Authority admin.
tld.soa.retry	CRConfig.json	The value for the retry field the Traffic Router DNS Server will respond with on SOA records.
tld.soa.refresh	CRConfig.json	The TTL the Traffic Router DNS Server will respond with on A records.
tld.ttls.NS	CRConfig.json	The TTL the Traffic Router DNS Server will respond with on NS records.
tld.ttls.SOA	CRConfig.json	The TTL the Traffic Router DNS Server will respond with on SOA records.
api.port	server.xml	The TCP port Traffic Router listens on for API (REST) access.
api.cache-control.max-age	CRConfig.json	The value of the `Cache-Control: max-age=` header in the API responses of Traffic Router.

Delivery Service Regexp

This table defines how requests are matched to the delivery service. There are 3 type of entries possible here:

Name	Description	DS Type	Status
HOST_REGEXP	This is the regular expresion to match the host part of the URL.	DNS and HTTP	Supported
PATH_REGEXP	This is the regular expresion to match the path part of the URL.	HTTP	Beta
HEADER_REGEXP	This is the regular expresion to match on any header in the request.	HTTP	Beta

The **Order** entry defines the order in which the regular expressions get evaluated. To support `CNAMES` from domains outside of the Traffic Control top level DNS domain, enter multiple `HOST_REGEXP` lines.

Example: Example foo.

Note: In most cases is is sufficient to have just one entry in this table that has a `HOST_REGEXP` Type, and Order `0`. For the *movies* delivery service in the Kabletown CDN, the entry is simply single `HOST_REGEXP` set to `.*\.movies\..*`. This will match every url that has a hostname that ends with `movies.cdn1.kabletown.net`, since `cdn1.kabletown.net` is the Kabletown CDN's DNS domain.

Static DNS Entries

Static DNS entries allow you to create other names *under* the delivery service domain. You can enter any valid hostname, and create a CNAME, A or AAAA record for it by clicking the **Static DNS** button at the bottom of the delivery service details screen.

Server Assignments

Click the **Server Assignments** button at the bottom of the screen to assign servers to this delivery service. Servers can be selected by drilling down in a tree, starting at the profile, then the cache group, and then the individual servers. Traffic Router will only route traffic for this delivery service to servers that are assigned to it.

The Coverage Zone File and ASN Table

The Coverage Zone File (CZF) should contain a cachegroup name to network prefix mapping in the form:

```
{
  "coverageZones": {
    "cache-group-01": {
      "network6": [
        "1234:5678::\/64",
        "1234:5679::\/64"
      ],
      "network": [
        "192.168.8.0\/24",
        "192.168.9.0\/24"
      ]
    }
    "cache-group-02": {
      "network6": [
        "1234:567a::\/64",
        "1234:567b::\/64"
```

(continues on next page)

(continued from previous page)

```
      ],
      "network": [
        "192.168.4.0\/24",
        "192.168.5.0\/24"
      ]
    }
  }
}
```

The CZF is an input to the Traffic Control CDN, and as such does not get generated by Traffic Ops, but rather, it gets consumed by Traffic Router. Some popular IP management systems output a very similar file to the CZF but in stead of a cachegroup an ASN will be listed. Traffic Ops has the “Networks (ASNs)” view to aid with the conversion of files like that to a Traffic Control CZF file; this table is not used anywhere in Traffic Ops, but can be used to script the conversion using the API.

The script that generates the CZF file is not part of Traffic Control, since it is different for each situation.

Parameters and Profiles

Parameters are shared between profiles if the set of `{ name, config_file, value }` is the same. To change a value in one profile but not in others, the parameter has to be removed from the profile you want to change it in, and a new parameter entry has to be created (**Add Parameter** button at the bottom of the Parameters view), and assigned to that profile. It is easy to create new profiles from the **Misc > Profiles** view - just use the **Add/Copy Profile** button at the bottom of the profile view to copy an existing profile to a new one. Profiles can be exported from one system and imported to another using the profile view as well. It makes no sense for a parameter to not be assigned to a single profile - in that case it really has no function. To find parameters like that use the **Parameters > Orphaned Parameters** view. It is easy to create orphaned parameters by removing all profiles, or not assigning a profile directly after creating the parameter.

See also:

Parameters an profiles in the *Configuring Traffic Ops* section.

Tools

Generate ISO

Queue Updates and Snapshot CRConfig

When changing delivery services special care has to be taken so that Traffic Router will not send traffic to caches for delivery services that the cache doesn’t know about yet. In general, when adding delivery services, or adding servers to a delivery service, it is best to update the caches before updating Traffic Router and Traffic Monitor. When deleting delivery services, or deleting server assignments to delivery services, it is best to update Traffic Router and Traffic Monitor first and then the caches. Updating the cache configuration is done through the *Queue Updates* menu, and updating Traffic Monitor and Traffic Router config is done through the *Snapshot CRConfig* menu.

Queue Updates

Every 15 minutes the caches will run a *syncds* to get all changes needed from Traffic Ops. The files that will be updated by the syncds job are:

- records.config

- remap.config
- parent.config
- cache.config
- hosting.config
- url_sig_(.*).config
- hdr_rw_(.*).config
- regex_revalidate.config
- ip_allow.config

A cache will only get updated when the update flag is set for it. To set the update flag, use the *Queue Updates* menu - here you can schedule updates for a whole CDN or a cache group:

1. Click **Tools > Queue Updates**.
2. Select the CDN to queueu uodates for, or All.
3. Select the cache group to queue updates for, or All
4. Click the **Queue Updates** button.
5. When the Queue Updates for this Server? (all) window opens, click **OK**.

To schedule updates for just one cache, use the "Server Checks" page, and click the ✓ in the *UPD* column. The UPD column of Server Checks page will change show a 🕒 when updates are pending for that cache.

Snapshot CRConfig

Every 60 seconds Traffic Monitor will check with Traffic Ops to see if a new CRConfig snapshot was made. If there is a new CRCOnfig, it will apply it to both Traffic Monitor and Traffic Router. See rl-crconfig for more information on the CRConfig file. To create a new snapshot, use the *Tools > Snapshot CRConfig* menu:

1. Click **Tools > Snapshot CRConfig**.
2. Verify the selection of the correct CDN from the Choose CDN drop down and click **Diff CRConfig**. On initial selection of this, the CRConfig Diff window says the following:

   There is no existing CRConfig for [cdn] to diff against... Is this the first snapshot??? If you are not sure why you are getting this message, please do not proceed! To proceed writing the snapshot anyway click the 'Write CRConfig' button below.

   If there is an older version of the CRConfig, a window will pop up showing the differences between the active CRConfig and the CRConfig about to be written.
3. Click **Write CRConfig**.
4. When the This will push out a new CRConfig.json. Are you sure? window opens, click **OK**.
5. The Successfully wrote CRConfig.json! window opens, click **OK**.

Invalidate Content

Invalidating content on the CDN is sometimes necessary when the origin was mis configured and something is cached in the CDN caches that needs to be removed. Given the size of a typical Traffic Control CDN and the amount of content that can be cached in it, removing the content from all the caches may take a long time. To speed up content invalidation, Traffic Ops will not try to remove the content from the caches, but it makes the content in accessible using the *regex_revalidate* ATS plugin. This forces a *revalidation* of the content, rather than a new get.

Note: This method forces a HTTP *revalidation* of the content, and not a new *GET* - the origin needs to support revalidation according to the HTTP/1.1 specification, and send a `200 OK` or `304 Not Modified` as applicable.

To invalidate content:

1. Click **Tools > Invalidate Content**
2. Fill out the form fields:

- Select the **Delivery Service**
- Enter the **Path Regex** - this should be a PCRE compatible regular expression for the path to match for forcing the revalidation. Be careful to only match on the content you need to remove - revalidation is an expensive operation for many origins, and a simple `/.*` can cause an overload condition of the origin.
- Enter the **Time To Live** - this is how long the revalidation force will be active for. It usually makes sense to make this the same as the `Cache-Control` header from the origin sets the object time to live in cache (by `max-age` or `Expires`). Entering a longer TTL here will make the caches do unnecessary work.
- Enter the **Start Time** - this is the start time when the force revalidation will be made active. Is pre populated with the current time, leave as is to schedule ASAP.

3. Click the **Submit** button.

Generate DNSSEC Keys

TBD

3.1.4 Managing Traffic Ops Extensions

3.1.5 Traffic Monitor Administration

Installing Traffic Monitor

The following are requirements to ensure an accurate set up:

- CentOS 6
- 4 vCPUs
- 8GB RAM
- Successful install of Traffic Ops
- Tomcat
- Administrative access to the Traffic Ops
- Physical address of the site
- perl-JSON
- perl-WWW-Curl

1. Enter the Traffic Monitor server into Traffic Ops
2. Make sure the FQDN of the Traffic Monitor is resolvable in DNS.
3. Install Traffic Monitor and perl mods: `sudo yum -y install traffic_monitor perl-JSON perl-WWW-Curl`
4. Take the config from Traffic Ops - run : `sudo /opt/traffic_monitor/bin/traffic_monitor_config.pl`

   Sample output:

```
traffic_mon # /opt/traffic_monitor/bin/traffic_monitor_config.pl https://
↪traffic-ops.cdn.kabletown.net admin:password prompt
DEBUG: traffic_ops selected: https://traffic-ops.cdn.kabletown.net
DEBUG: traffic_ops login: admin:kl0tevax
DEBUG: Config write mode: prompt
DEBUG: Found profile from traffic_ops: RASCAL_CDN
DEBUG: Found CDN name from traffic_ops: kabletown_cdn
DEBUG: Found location for rascal-config.txt from traffic_ops: /opt/
↪traffic_monitor/conf
WARN: Param not in traffic_ops: allow.config.edit                          ␣
↪description: Allow the running configuration to be edited through the␣
↪UI                                                                  Using␣
↪default value of: false
WARN: Param not in traffic_ops: default.accessControlAllowOrigin           ␣
↪description: The value for the header: Access-Control-Allow-Origin for␣
↪published jsons... should be narrowed down to TMs                  Using␣
↪default value of: *
WARN: Param not in traffic_ops: default.connection.timeout                 ␣
↪description: Default connection time for all queries (cache, peers, TM)␣
↪                                                                  Using␣
↪default value of: 2000
WARN: Param not in traffic_ops: hack.forceSystemExit                       ␣
↪description: Call System.exit on shutdown                                 ␣
↪                                                                  Using␣
↪default value of: false
WARN: Param not in traffic_ops: hack.peerOptimistic                        ␣
↪description: The assumption of a caches availability when unknown by␣
↪peers                                                                Using␣
↪default value of: true
WARN: Param not in traffic_ops: hack.publishDsStates                       ␣
↪description: If true, the delivery service states will be included in␣
↪the CrStates.json                                                   Using␣
↪default value of: true
WARN: Param not in traffic_ops: health.ds.interval                         ␣
↪description: The polling frequency for calculating the deliveryService␣
↪states                                                             Using␣
↪default value of: 1000
WARN: Param not in traffic_ops: health.ds.leniency                         ␣
↪description: The amount of time before the deliveryService disregards␣
↪the last update from a non-responsive cache                         Using␣
↪default value of: 30000
WARN: Param not in traffic_ops: health.event-count                         ␣
↪description: The number of historical events that will be kept           ␣
↪                                                                  Using␣
↪default value of: 200
WARN: Param not in traffic_ops: health.polling.interval                    ␣
↪description: The polling frequency for getting the states from caches    ␣
↪                                                                  Using␣
↪default value of: 5000
```

(continues on next page)

(continued from previous page)

```
WARN: Param not in traffic_ops: health.startupMinCycles                         ␣
↪description: The number of query cycles that must be completed before␣
↪this Traffic Monitor will start reporting                           Using␣
↪default value of: 2
WARN: Param not in traffic_ops: health.timepad                                  ␣
↪description: A delay between each separate cache query                        ␣
↪                                                                  Using␣
↪default value of: 10
WARN: Param not in traffic_ops: peers.polling.interval                          ␣
↪description: Polling frequency for getting states from peer monitors          ␣
↪                                                                  Using␣
↪default value of: 5000
WARN: Param not in traffic_ops: peers.polling.url                               ␣
↪description: The url for current, unfiltered states from peer monitors        ␣
↪                                                                  Using␣
↪default value of: http://${hostname}/publish/CrStates?raw
WARN: Param not in traffic_ops: peers.threadPool                                ␣
↪description: The number of threads given to the pool for querying peers␣
↪                                                                  Using␣
↪default value of: 1
WARN: Param not in traffic_ops: tm.auth.url                                     ␣
↪description: The url for the authentication form                              ␣
↪                                                                  Using␣
↪default value of: https://${tmHostname}/login
WARN: Param not in traffic_ops: tm.crConfig.json.polling.url                    ␣
↪description: Url for the cr-config (json)                                     ␣
↪                                                                  Using␣
↪default value of: https://${tmHostname}/CRConfig-Snapshots/${cdnName}/
↪CRConfig.json
WARN: Param not in traffic_ops: tm.healthParams.polling.url                     ␣
↪description: The url for the heath params (json)                              ␣
↪                                                                  Using␣
↪default value of: https://${tmHostname}/health/${cdnName}
WARN: Param not in traffic_ops: tm.polling.interval                             ␣
↪description: The polling frequency for getting updates from TM                ␣
↪                                                                  Using␣
↪default value of: 10000
DEBUG: allow.config.edit needed in config, but does not exist in config␣
↪on disk.
DEBUG: cdnName value on disk () does not match value needed in config␣
↪(kabletown_cdn).
DEBUG: default.accessControlAllowOrigin needed in config, but does not␣
↪exist in config on disk.
DEBUG: default.connection.timeout needed in config, but does not exist in␣
↪config on disk.
DEBUG: hack.forceSystemExit needed in config, but does not exist in␣
↪config on disk.
DEBUG: hack.peerOptimistic needed in config, but does not exist in config␣
↪on disk.
DEBUG: hack.publishDsStates needed in config, but does not exist in␣
↪config on disk.
DEBUG: health.ds.interval needed in config, but does not exist in config␣
↪on disk.
DEBUG: health.ds.leniency needed in config, but does not exist in config␣
↪on disk.
DEBUG: health.startupMinCycles needed in config, but does not exist in␣
↪config on disk.
```

(continues on next page)

(continued from previous page)

```
DEBUG: health.timepad value on disk (20) does not match value needed in␣
↪config (10).
DEBUG: peers.polling.interval needed in config, but does not exist in␣
↪config on disk.
DEBUG: peers.threadPool needed in config, but does not exist in config on␣
↪disk.
DEBUG: tm.auth.password value on disk () does not match value needed in␣
↪config (kl0tevax).
DEBUG: tm.auth.username value on disk () does not match value needed in␣
↪config (admin).
DEBUG: tm.hostname value on disk () does not match value needed in config␣
↪(traffic-ops.cdn.kabletown.net).
DEBUG: Proposed traffic_monitor_config:
{
   "traffic_monitor_config":{
      "default.accessControlAllowOrigin":"*",
      "health.startupMinCycles":"2",
      "tm.auth.password":"kl0tevax",
      "tm.auth.url":"https://${tmHostname}/login",
      "tm.healthParams.polling.url":"https://${tmHostname}/health/$
↪{cdnName}",
      "allow.config.edit":"false",
      "tm.crConfig.json.polling.url":"https://${tmHostname}/CRConfig-
↪Snapshots/${cdnName}/CRConfig.json",
      "tm.auth.username":"admin",
      "peers.polling.url":"http://${hostname}/publish/CrStates?raw",
      "health.timepad":"10",
      "hack.publishDsStates":"true",
      "default.connection.timeout":"2000",
      "health.ds.interval":"1000",
      "peers.polling.interval":"5000",
      "hack.forceSystemExit":"false",
      "health.ds.leniency":"30000",
      "cdnName":"kabletown_cdn",
      "peers.threadPool":"1",
      "tm.polling.interval":"10000",
      "health.polling.interval":"5000",
      "health.event-count":"200",
      "hack.peerOptimistic":"true",
      "tm.hostname":"traffic-ops.cdn.kabletown.net"
   }
}
----------------------------------------------
----OK to write this config to disk? (Y/n) [n]y
----------------------------------------------
----------------------------------------------
----OK to write this config to disk? (Y/n) [n]Y
----------------------------------------------
DEBUG: Writing /opt/traffic_monitor/conf/traffic_monitor_config.js
traffic_mon #
```

5. Start Tomcat: `sudo service tomcat start`

```
Using CATALINA_BASE: /opt/tomcat
Using CATALINA_HOME: /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME: /usr
```

(continues on next page)

(continued from previous page)

```
Using CLASSPATH:/opt/tomcat/bin/bootstrap.jar
Using CATALINA_PID:/var/run/tomcat/tomcat.pid
Starting tomcat [ OK ]
```

6. Verify Traffic Monitor is running by pointing your browser to port 80 on the Traffic Monitor host.

Configuring Traffic Monitor

Configuration Overview

Traffic Monitor is configured using its JSON configuration file, `traffic_monitor_config.js`. Specify the URL, username, password, and CDN name for the instance of Traffic Ops for which this Traffic Monitor is a member, and start the software. Once started with the correct configuration, Traffic Monitor downloads its configuration from Traffic Ops and begins polling caches. Once a configurable number of polling cycles completes, health protocol state is available via RESTful JSON endpoints.

Troubleshooting and log files

Traffic Monitor log files are in `/opt/traffic_monitor/var/log/`, and tomcat log files are in `/opt/tomcat/logs/`.

3.1.6 Traffic Router Administration

Installing Traffic Router

The following are requirements to ensure an accurate set up:

- CentOS 6
- 4 vCPUs
- 8GB RAM
- Successful install of Traffic Ops
- Successful install of Traffic Monitor
- Administrative access to the Traffic Ops
- Physical address of the site
- perl-JSON
- perl-WWW-Curl

1. Enter the Traffic Router server into Traffic Ops.
2. Make sure the FQDN of the Traffic Monitor is resolvable in DNS.
3. Install a traffic router: `sudo yum install traffic_router`.
4. Edit `/opt/traffic_router/conf/traffic_monitor.properties` and put in the correct online Traffic Monitor(s) for your CDN.

   Example:

```
# list of ips or hostnames delimited by semicolon (;)
traffic_monitor.bootstrap.hosts=traffic-mon-01.cdn.kabletown.net:80;

# Instead of using the traffic_monitor.bootstrap.hosts property as a␣
↪bootstrap
# source before switching to ONLINE Monitors in the TrConfig, always
# use the hosts listed for TrConfig and TrStates. Defaults to false.
traffic_monitor.bootstrap.local = false

# traffic_monitor.properties: url that should normally point to this file
traffic_monitor.properties=file:/opt/traffic_router/conf/traffic_monitor.
↪properties

# Frequency for reloading this file
# traffic_monitor.properties.reload.period=60000
```

5. Start Tomcat: `sudo service tomcat start`, and test lookups with dig and curl against that server.
6. Snapshot CRConfig

- This instantly associates production traffic on the servers. They need to be online when you change the DNS records.

7. Add the correct DNS entries to the SOA records for the CDN on which you are working.
8. Add the servers to the NS and SOA records for your domain.

Configuring Traffic Router

1. From **Misc > Profiles**, verify the following:

- The Traffic Router information.
- The profile is set correctly.
- The Status is set to OFFLINE.

2. Verify the functionality of the DNS entry for the Traffic Router.
3. Click **Tools > Generate ISO**.
4. Complete the necessary fields.
5. Click **Download ISO**.

Troubleshooting and log files

Traffic Router log files are in `/opt/traffic_router/var/log/`, and tomcat log files are in `/opt/tomcat/logs/`.

3.1.7 Traffic Stats Administration

Installing Traffic Stats

Configuring Traffic Stats

3.1.8 Traffic Server Administration

Installing Traffic Server

1. Select **Servers**.
2. Scroll to the bottom of the page and click **Add Server**.
3. Complete the Required Info: section.
4. Click **Submit**.
5. Click **Save**.

Configuring Traffic Server

All of the Traffic Server application configuration files are generated by Traffic Ops and installed by way of the traffic_ops_ort.pl script.

traffic_ops_ort.pl The traffic_ops_ort.pl should be installed on all caches (by puppet or other non Traffic Ops means), usually in /opt/ort. It is used to do initial install of the config files when the cache is being deployed, and to keep the config files up to date when the cache is already in service. The usage message of the script is shown below:

```
$  /opt/ort/traffic_ops_ort.pl
Mon Mar  9 18:38:01 UTC 2015
Version of this script: 0.46b
====-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-====
Usage: ./traffic_ops_ort.pl <Mode> <Log_Level> <Traffic_Ops_URL> <Traffic_Ops_
↪Login>
    <Mode> = interactive - asks questions during config process.
    <Mode> = report - prints config differences and exits.
    <Mode> = badass - attempts to fix all config differences that it can.
    <Mode> = syncds - syncs delivery services with what is configured in Traffic␣
↪Ops.

    <Log_Level> => ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, NONE

    <Traffic_Ops_URL> = URL to 12 monkeys host. Example: https://trafficops.
↪company.net

    <Traffic_Ops_Login> => Example: 'username:password'
====-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-====
$
```

For initial configuration or when major changes (like a Profile change) need to be made, run the script in “badass mode”. All required rpm packages will be installed, all Traffic Server config files will be fetched and installed, and (if needed) the Traffic Server application will be restarted. Example run below:

```
run here
```

For "every day changes" such as adding deliveryservices or changing records.config parameters caches should run the script in "syncds" mode out of cron. Example crontab entry:

```
*/15 * * * * /opt/ort/traffic_ops_ort.pl syncds warn https://traffops.kabletown.
↪net admin:password > /tmp/ort/syncds.log 2>&1
```

Note: <disclaimer on what is "hot changeable" here>

3.1.9 Traffic Vault Administration

Installing Traffic Vault

In order to successfully store private keys you will need to install Riak. The latest version of Riak can be downloaded on the Riak website. The installation instructions for Riak can be found here.

Production is currently running version 2.0.5 of Riak, but the latest version should suffice.

Configuring Traffic Vault

The following steps were taken to configure Riak in our environments.

Riak configuration file configuration

The following steps need to be performed on each Riak server in the cluster:

- Log into riak server as root
- cd to /etc/riak/
- **Update the following in riak.conf to reflect your IP:**
  - nodename = riak@a-host.sys.kabletown.net
  - listener.http.internal = a-host.sys.kabletown.net:8098 (can be 80 - This endpoint will not work with sec enabled)
  - listener.protobuf.internal = a-host.sys.kabletown.net:8087 (can be different port if you want)
  - listener.https.internal = a-host.sys.kabletown.net:8088 (can be 443)
- **Updated the following conf file to point to your cert files**
  - ssl.certfile = /etc/riak/certs/server.crt
  - ssl.keyfile = /etc/riak/certs/server.key
  - ssl.cacertfile = /etc/pki/tls/certs/ca-bundle.crt
- **Add a line at the bottom of the config for tlsv1**
  - tls_protocols.tlsv1 = on
- **Once the config file has been updated restart riak**
  - `/etc/init.d/riak restart`
- **Validate server is running by going to the following URL:**
  - https://<serverHostname>:8088/ping

Riak-admin configuration

Riak-admin is a command line utility that needs to be run as root on a server in the riak cluster.

Assumptions:

- Riak 2.0.2 or greater is installed
- SSL Certificates have been generated (signed or self-signed)
- Root access to riak servers

Add admin user and riakuser to riak

- Admin user will be a super user
- Riakuser will be the application user

Login to one of the riak servers in the cluster as root (any will do)

1. Enable security

   ```
   riak-admin security enable
   ```

2. Add groups

   ```
   riak-admin security add-group admins
   ```

   ```
   riak-admin security add-group keysusers
   ```

3. Add users

   Note: username and password should be stored in /opt/traffic_ops/app/conf/<environment>/riak.conf

   ```
   riak-admin security add-user admin password=<AdminPassword>
   groups=admins
   ```

   ```
   riak-admin security add-user riakuser
   password=<RiakUserPassword> groups=keysusers
   ```

4. Grant access for admin and riakuser

   ```
   riak-admin security add-source riakuser 0.0.0.0/0 password
   ```

   ```
   riak-admin security add-source admin 0.0.0.0/0 password
   ```

5. Grant privs to admins for everything

   ```
   riak-admin security grant riak_kv.list_buckets,riak_kv.
   list_keys,riak_kv.get,riak_kv.put,riak_kv.delete on any to
   admins
   ```

6. Grant privs to keysuser for ssl, dnssec, and url_sig_keys buckets only

   ```
   riak-admin security grant riak_kv.get,riak_kv.put,riak_kv.
   delete on default ssl to keysusers
   ```

   ```
   riak-admin security grant riak_kv.get,riak_kv.put,riak_kv.
   delete on default dnssec to keysusers
   ```

   ```
   riak-admin security grant riak_kv.get,riak_kv.put,riak_kv.
   delete on default url_sig_keys to keysusers
   ```

See also:

For more information on security in Riak, see the Riak Security documentation.

See also:

For more information on authentication and authorization in Riak, see the Riak Authentication and Authorization documentation.

Traffic Ops Configuration

There are a couple conifgurations that are necessary in Traffic Ops.

1. **Database Updates**
    - A new profile for Riak needs to be added to the profile table
    - A new type of Riak needs to be added to the type table
    - The servers in the Riak cluster need to be added to the server table

    Note: profile and type data should be pre-loaded by seeds sql script.

2. **Configuration updates**
    - /opt/traffic_ops/app/conf/<environment>/riak.conf needs to be updated to reflect the correct username and password for accessing riak.

CHAPTER 4

Developer's Guide

A guide to the various internal and external APIs, and a introduction for the Traffic Control developer.

4.1 Developer's Guide

Use this guide to start developing applications that consume the Traffic Control APIs, to create extensions to Traffic Ops, or work on Traffic Control itself.

4.1.1 Traffic Ops

Introduction

Traffic Ops uses a MySql or Postgres database to store the configuration information, and the Mojolicious framework to generate the user interface and REST APIs.

Software Requirements

To work on Traffic Ops you need a *nix (MacOS and Linux are most commonly used) environment that has the following installed:

- Carton 1.0.12
- Go 1.4
- Perl 5.10.1
- Git
- MySQL 5.1.52

Traffic Ops Project Tree Overview

/opt/traffic_ops/app

- bin/ - Directory for scripts, cronjobs, etc.
- conf/
  - /development - Development (local) specific config files.
  - /misc - Miscellaneous config files.
  - /production - Production specific config files.
  - /test - Test (unit test) specific config files.
- db/ - Database related area.
  - /migrations - Database Migration files.
- lib/
  - /API - Mojo Controllers for the /API area of the application.
  - /Common - Common Code between both the API and UI areas.
  - /Extensions
  - Fixtures/ - Test Case fixture data for the 'to_test' database. * /Integration - Integration Tests.
  - /MojoPlugins - Mojolicious Plugins for Common Controller Code.
  - Schema/ - Database Schema area. * /Result - DBIx ORM related files.
  - /Test - Common Test.
  - /UI - Mojo Controllers for the Traffic Ops UI itself.
  - Utils/ * /Helper - Common utilities for the Traffic Ops application.
- log/ - Log directory where the development and test files are written by the app.
- public/
- css/ - Stylesheets.
- images/ - Images.
- js/ - Javascripts
- script/ - Mojo Bootstrap scripts.
- t/ - Unit Tests for the UI.
- api/ - Unit Tests for the API.
- t_integration/ - High level tests for Integration level testing.
- templates/ - Mojo Embedded Perl (.ep) files for the UI.

Perl Formatting Conventions

Perl tidy is for use in code formatting. See the following config file for formatting conventions.

```
edit a file called $HOME/.perltidyrc

l = 156
et=4
t
ci=4
st
se
vt=0
cti=0
pt=1
bt=1
sbt=1
bbt=1
nsfs
nolq
otr
aws
wls="= + - / * ."
wrs=\"= + - / * .\"
wbb =% + - * / x != == >= <= =~ < > | & **= += *= &= <<= &&= -= /= |= + >>= ||= .= %=␣
↪^= x=
```

Database Management

The admin.pl script is for use in managing the Traffic Ops database tables. Below is an example of its usage.

`$ db/admin.pl`

Usage: db/admin.pl [–env (development|test|production)] [arguments]

Example: `db/admin.pl --env=test reset`

Purpose: This script is used to manage the database. The environments are defined in the dbconf.yml, as well as the database names.

Arguments	Description
create	Execute db 'create' the database for the current environment.
down	Roll back a single migration from the current version.
drop	Execute db 'drop' on the database for the current environment.
redo	Roll back the most recently applied migration, then run it again.
reset	Execute db drop, create, load_schema, migrate on the database for the current environment.
seed	Execute SQL from db/seeds.sql for loading static data.
setup	Execute db drop, create, load_schema, migrate, seed on the database for the current environment.
status	Print the status of all migrations.
upgrade	Execute migrate then seed on the database for the current environment.

Installing The Developer Environment

To install the Traffic Ops Developer environment:

1. Clone the traffic_control repository from github.com.
2. Install the local dependencies using Carton (cpanfile).

```
$ cd traffic_ops/app
$ carton
```

3. Set up a user in MySQL.

   Example:

```
master $ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 305
Server version: 5.6.19 Homebrew

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input␣
↪statement.

mysql> create user ‘to_user’@’localhost’;
mysql> grant all on to_development.* to 'to_user'@'localhost' identified by
↪'twelve';
mysql> grant all on to_test.* to 'to_user'@'localhost' identified by 'twelve
↪';
mysql> grant all on to_integration.* to 'to_user'@'localhost' identified by
↪'twelve';
```

4. Enter `db/admin.pl --env=<enviroment name> setup` to set up the traffic_ops database(s).

   - Unit test database: `$ db/admin.pl --env=test setup`
   - Development database: `$ db/admin.pl --env=development setup`
   - Integration database: `$ db use db/admin.pl --env=integration setup`

   The database schema should look like this:

```
master $ db/admin.pl --env=development setup
Using database.conf: conf/development/database.conf
Using database.conf: conf/development/database.conf
Using database.conf: conf/development/database.conf
Using database.conf: conf/development/database.conf
Using database.conf: conf/development/database.conf
Using database.conf: conf/development/database.conf
Executing 'drop database to_development'
Executing 'create database to_development'
Creating database tables...
Warning: Using a password on the command line interface can be insecure.
Migrating database...
goose: migrating db environment 'development', current version: 0, target:␣
↪20150210100000
OK    20141222103718_extension.sql
OK    20150108100000_add_job_deliveryservice.sql
OK    20150205100000_cg_location.sql
OK    20150209100000_cran_to_asn.sql
OK    20150210100000_ds_keyinfo.sql
```

(continues on next page)

(continued from previous page)

```
Seeding database...
Warning: Using a password on the command line interface can be insecure.
```

5. (Optional) To load temporary data into the tables: `$ perl bin/db/setup_kabletown.pl`
6. To start Traffic Ops, enter `$ bin/start.sh`

   The local Traffic Ops instance uses an open source framework called morbo, starting following the start command execution.

   Start up success includes the following:

```
[2015-02-24 10:44:34,991] [INFO] Listening at "http://*:3000".

Server available at http://127.0.0.1:3000.
```

7. Using a browser, navigate to the given address: `http://127.0.0.1:3000`
8. For the initial log in:

- User name: admin
- Password: password

9. Change the log in information.

Test Cases

Use prove to execute test cases. Execute after a carton install:

- To run the Unit Tests: `$ local/bin/prove -qrp t/`
- To run the Integration Tests: `$ local/bin/prove -qrp t_integration/`

The KableTown CDN example

The integration tests will load an example CDN with most of the features of Traffic Control being used. This is mostly for testing purposes, but can also be used as an example of how to configure certain features. To load the KableTown CDN example and access it:

1. Run the integration tests
2. Start morbo against the integration database: `export MOJO_MODE=integration; ./bin/start.sh`
3. Using a browser, navigate to the given address: `http://127.0.0.1:3000`
4. For the initial log in:

- User name: admin
- Password: password

Extensions

Traffic Ops Extensions are a way to enhance the basic functionality of Traffic Ops in a custom manner. There are three types of extensions:

1. Check Extensions

These allow you to add custom checks to the "Health->Server Checks" view.

2. Configuration Extensions

   These allow you to add custom configuration file generators.

3. Data source Extensions

   These allow you to add statistic sources for the graph views and APIs.

Extensions are managed using the $TO_HOME/bin/extensions command line script. For more information see *Managing Traffic Ops Extensions*.

Check Extensions

In other words, check extensions are scripts that, after registering with Traffic Ops, have a column reserved in the "Health->Server Checks" view and that usually run periodically out of cron.

It is the responsibility of the check extension script to iterate over the servers it wants to check and post the results. A check extension can have a column of ✓'s and ✗'s (CHECK_EXTENSION_BOOL) or a column that shows a number (CHECK_EXTENSION_NUM). A simple example of a check extension of type CHECK_EXTENSION_NUM that will show 99.33 for all servers of type EDGE is shown below:

```
Script here.
```

Check Extension scripts are located in the $TO_HOME/bin/checks directory.

Currently, the following Check Extensions are available and installed by default:

Cache Disk Usage Check - CDU This check shows how much of the available total cache disk is in use. A "warm" cache should show 100.00.

Cache Hit Ratio Check - CHR The cache hit ratio for the cache in the last 15 minutes (the interval is determined by the cron entry).

DiffServe CodePoint Check - DSCP Checks if the returning traffic from the cache has the correct DSCP value as assigned in the delivery service. (Some routers will overwrite DSCP)

Maximum Transmission Check - MTU Checks if the Traffic Ops host (if that is the one running the check) can send and receive 8192 size packets to the `ip_address` of the server in the server table.

Operational Readiness Check - ORT See *Configuring Traffic Server* for more information on the ort script. The ORT column shows how many changes the traffic_ops_ort.pl script would apply if it was run. The number in this column should be 0.

Ping Check - 10G, ILO, 10G6, FQDN The bin/checks/ToPingCheck.pl is to check basic IP connectivity, and in the default setup it checks IP connectivity to the following:

10G Is the `ip_address` (the main IPv4 address) from the server table pingable?

ILO Is the `ilo_ip_address` (the lights-out-mangement IPv4 address) from the server table pingable?

10G6 Is the `ip6_address` (the main IPv6 address) from the server table pingable?

FQDN Is the Fully Qualified Domain name (the concatenation of `host_name` and `.` and `domain_name` from the server table) pingable?

Traffic Router Check - RTR

Configuration Extensions

NOTE: Config Extensions are Beta at this time.

Data source Extensions

NOTE: Data source Extensions are Beta at this time.

API

The Traffic Ops API provides programmatic access to read and write CDN data providing authorized API consumers with the ability to monitor CDN performance and configure CDN settings and parameters.

Response Structure

All successful responses have the following structure:

```
{
  "response": <JSON object with main response>,
  "version": "1.1"
}
```

To make the documentation easier to read, only the `<JSON object with main response>` is documented, even though the response and version fields are always present.

Using API Endpoints

1. Authenticate with your Traffic Portal or Traffic Ops user account credentials.
2. Upon successful user authentication, note the mojolicious cookie value in the response headers.
3. Pass the mojolicious cookie value, along with any subsequent calls to an authenticated API endpoint.

Example:

```
[jvd@laika ~]$ curl -H "Accept: application/json" http://localhost:3000/api/1.1/usage/
↪asns.json
{"version":"1.1","alerts":[{"level":"error","text":"Unauthorized, please log in."}]}
[jvd@laika ~]$
[jvd@laika ~]$ curl -v -H "Accept: application/json" -v -X POST --data '{ "u":"admin",
↪ "p":"secret_passwd" }' http://localhost:3000/api/1.1/user/login
* Hostname was NOT found in DNS cache
*   Trying ::1...
* connect to ::1 port 3000 failed: Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> POST /api/1.1/user/login HTTP/1.1
> User-Agent: curl/7.37.1
> Host: localhost:3000
> Accept: application/json
> Content-Length: 32
> Content-Type: application/x-www-form-urlencoded
>
```

(continues on next page)

(continued from previous page)

```
* upload completely sent off: 32 out of 32 bytes
< HTTP/1.1 200 OK
< Connection: keep-alive
< Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
< Access-Control-Allow-Origin: http://localhost:8080
< Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
< Set-Cookie: mojolicious=eyJleHBpcmVzIjoxNDI5NDAyMjAxLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪f990d03b7180b1ece97c3bb5ca69803cd6a79862; expires=Sun, 19 Apr 2015 00:10:01 GMT;␣
↪path=/; HttpOnly
< Content-Type: application/json
< Date: Sat, 18 Apr 2015 20:10:01 GMT
< Access-Control-Allow-Credentials: true
< Content-Length: 81
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
* Server Mojolicious (Perl) is not blacklisted
< Server: Mojolicious (Perl)
<
* Connection #0 to host localhost left intact
{"alerts":[{"level":"success","text":"Successfully logged in."}],"version":"1.1"}
[jvd@laika ~]$

[jvd@laika ~]$ curl -H'Cookie:␣
↪mojolicious=eyJleHBpcmVzIjoxNDI5NDAyMjAxLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪f990d03b7180b1ece97c3bb5ca69803cd6a79862;' -H "Accept: application/json" http://
↪localhost:3000/api/1.1/asns.json
{"response":{"asns":[{"lastUpdated":"2012-09-17 15:41:22", .. asn data deleted ..   ,
↪"version":"1.1"}
[jvd@laika ~]$
```

API Errors

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.
`version`	string	

The 3 most common errors returned by Traffic Ops are:

401 Unauthorized When you don't supply the right cookie, this is the response.

```
[jvd@laika ~]$ curl -v -H "Accept: application/json" http://localhost:3000/api/1.
↪1/usage/asns.json
* Hostname was NOT found in DNS cache
*   Trying ::1...
* connect to ::1 port 3000 failed: Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /api/1.1/usage/asns.json HTTP/1.1
> User-Agent: curl/7.37.1
> Host: localhost:3000
> Accept: application/json
```

(continues on next page)

(continued from previous page)

```
>
< HTTP/1.1 401 Unauthorized
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Content-Length: 84
* Server Mojolicious (Perl) is not blacklisted
< Server: Mojolicious (Perl)
< Connection: keep-alive
< Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
< Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
< Access-Control-Allow-Origin: http://localhost:8080
< Date: Sat, 18 Apr 2015 20:36:12 GMT
< Content-Type: application/json
< Access-Control-Allow-Credentials: true
<
* Connection #0 to host localhost left intact
{"version":"1.1","alerts":[{"level":"error","text":"Unauthorized, please log in."}
↪]}
[jvd@laika ~]$
```

404 Not Found When the resource (path) is non existant Traffic Ops returns a 404:

```
[jvd@laika ~]$ curl -v -H'Cookie:␣
↪mojolicious=eyJleHBpcmVzIjoxNDI5NDAyMjAxLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪f990d03b7180b1ece97c3bb5ca69803cd6a79862;' -H "Accept: application/json" http://
↪localhost:3000/api/1.1/asnsjj.json
* Hostname was NOT found in DNS cache
*   Trying ::1...
* connect to ::1 port 3000 failed: Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /api/1.1/asnsjj.json HTTP/1.1
> User-Agent: curl/7.37.1
> Host: localhost:3000
> Cookie: mojolicious=eyJleHBpcmVzIjoxNDI5NDAyMjAxLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪f990d03b7180b1ece97c3bb5ca69803cd6a79862;
> Accept: application/json
>
< HTTP/1.1 404 Not Found
* Server Mojolicious (Perl) is not blacklisted
< Server: Mojolicious (Perl)
< Content-Length: 75
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Content-Type: application/json
< Date: Sat, 18 Apr 2015 20:37:43 GMT
< Access-Control-Allow-Credentials: true
< Set-Cookie:␣
↪mojolicious=eyJleHBpcmVzIjoxNDI5NDAzODYzLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪8a5a61b91473bc785d4073fe711de8d2c63f02dd; expires=Sun, 19 Apr 2015 00:37:43 GMT;
↪ path=/; HttpOnly
< Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
< Connection: keep-alive
< Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
< Access-Control-Allow-Origin: http://localhost:8080
<
* Connection #0 to host localhost left intact
{"version":"1.1","alerts":[{"text":"Resource not found.","level":"error"}]}
[jvd@laika ~]$
```

500 Internal Server Error When you are asking for a correct path, but the database doesn't match, it returns a 500:

```
[jvd@laika ~]$ curl -v -H'Cookie:␣
↪mojolicious=eyJleHBpcmVzIjoxNDI5NDAyMjAxLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪f990d03b7180b1ece97c3bb5ca69803cd6a79862;' -H "Accept: application/json" http://
↪localhost:3000/api/1.1/servers/hostname/jj/details.json
* Hostname was NOT found in DNS cache
*   Trying ::1...
* connect to ::1 port 3000 failed: Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /api/1.1/servers/hostname/jj/details.json HTTP/1.1
> User-Agent: curl/7.37.1
> Host: localhost:3000
> Cookie: mojolicious=eyJleHBpcmVzIjoxNDI5NDAyMjAxLCJhdXRoX2RhdGEiOiJhZG1pbiJ9--
↪f990d03b7180b1ece97c3bb5ca69803cd6a79862;
> Accept: application/json
>
< HTTP/1.1 500 Internal Server Error
* Server Mojolicious (Perl) is not blacklisted
< Server: Mojolicious (Perl)
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Content-Length: 93
< Set-Cookie:␣
↪mojolicious=eyJhdXRoX2RhdGEiOiJhZG1pbiIsImV4cGlyZXMiOjE0Mjk0MDQzMDZ9--
↪1b08977e91f8f68b0ff5d5e5f6481c76ddfd0853; expires=Sun, 19 Apr 2015 00:45:06 GMT;
↪ path=/; HttpOnly
< Content-Type: application/json
< Date: Sat, 18 Apr 2015 20:45:06 GMT
< Access-Control-Allow-Credentials: true
< Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
< Connection: keep-alive
< Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
< Access-Control-Allow-Origin: http://localhost:8080
<
* Connection #0 to host localhost left intact
{"alerts":[{"level":"error","text":"An error occurred. Please contact your␣
↪administrator."}]}
[jvd@laika ~]$
```

The rest of the API documentation will only document the `200 OK` case, where no errors have occured.

API 1.1 Reference

ASN

GET /api/1.1/asns.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`asns`	array	A collection of asns
`>lastUpdated`	string	The Time / Date this server entry was last updated
`>id`	string	Local unique identifier for the ASN
`>asn`	string	Autonomous System Numbers per APNIC for identifying a service provider.
`>cachegroup`	string	Related cachegroup name

Response Example

```
{
 "response": {
    "asns": [
       {
          "lastUpdated": "2012-09-17 21:41:22",
          "id": "27",
          "asn": "7015",
          "cachegroup": "us-ma-woburn"
       },
       {
          "lastUpdated": "2012-09-17 21:41:22",
          "id": "28",
          "asn": "7016",
          "cachegroup": "us-pa-pittsburgh"
       }
    ]
 },
 "version": "1.1"
}
```

Cache Group

GET /api/1.1/cachegroups.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`longitude`	string	Longitude for the Cache Group
`parentCachegroupId`	string	Identifier that refers to the ‘id’ field of different Cache Group entry.
`lastUpdated`	string	The Time / Date this entry was last updated
`typeName`	string	The name of the type of Cache Group entry
`name`	string	The name of the Cache Group entry
`typeId`	string	Unique identifier for the ‘Type’ of Cache Group entry
`latitude`	string	Latitude for the Cache Group
`id`	string	Local unique identifier for the Cache Group
`shortName`	string	Abbreviation of the Cache Group Name

Response Example

```
{
 "response": [
    {
       "longitude": "0",
       "parentCachegroupId": null,
       "lastUpdated": "2012-09-25 20:27:28",
       "typeName": "MID_LOC",
       "name": "dc-chicago",
       "parentCachegroupName": null,
       "typeId": "4",
       "latitude": "0",
       "id": "21",
       "shortName": "dcchi"
    },
    {
       "longitude": "0",
       "parentCachegroupId": null,
       "lastUpdated": "2012-09-25 20:32:03",
       "typeName": "MID_LOC",
       "name": "dc-cmc",
       "parentCachegroupName": null,
       "typeId": "4",
       "latitude": "0",
       "id": "22",
       "shortName": "dccmc"
    }
 ],
 "version": "1.1"
}
```

GET /api/1.1/cachegroups/trimmed.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`name`	string	

Response Example

```
 {
  "response": [
     {
        "name": "dc-chicago"
     },
     {
        "name": "dc-cmc"
     }
  ],
  "version": "1.1"
}
```

GET /api/1.1/cachegroup/:parameter_id/parameter.json

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`parameter_id`	yes	

Response Properties

Parameter	Type	Description
`cachegroups`	array	
`>name`	string	
`>id`	string	

Response Example

```
{
 "response": {
    "cachegroups": [
       {
          "name": "dc-chicago",
          "id": "21"
       },
       {
          "name": "dc-cmc",
          "id": "22"
       }
    ]
 },
 "version": "1.1"
}
```

GET /api/1.1/cachegroupparameters.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`cachegroupParameters`	array	A collection of cache group parameters.
`>parameter`	string	
`>last_updated`	string	
`>cachegroup`	string	

Response Example

```
{
 "response": {
    "cachegroupParameters": [
       {
```

(continues on next page)

(continued from previous page)

```
            "parameter": "379",
            "last_updated": "2013-08-05 18:49:37",
            "cachegroup": "us-ca-sanjose"
         },
         {
            "parameter": "380",
            "last_updated": "2013-08-05 18:49:37",
            "cachegroup": "us-ca-sanjose"
         },
         {
            "parameter": "379",
            "last_updated": "2013-08-05 18:49:37",
            "cachegroup": "us-ma-woburn"
         }
      ]
 },
 "version": "1.1"
}
```

GET /api/1.1/cachegroups/:parameter_id/parameter/available.json

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`parameter_id`	yes	

Response Properties

Parameter	Type	Description
`name`		
`id`		

Response Example

```
{
 "response": [
    {
       "name": "dc-chicago",
       "id": "21"
    },
    {
       "name": "dc-cmc",
       "id": "22"
    }
 ],
 "version": "1.1"
}
```

CDN

Health

GET /api/1.1/cdns/health.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
totalOnline	int	Total number of online caches across all CDNs.
totalOffline	int	Total number of offline caches across all CDNs.
cachegroups	array	A collection of cache groups.
>online	int	The number of online caches for the cache group
>offline	int	The number of offline caches for the cache group.
>name	string	Cache group name.

Response Example

```
{
 "response": {
    "totalOnline": 148,
    "totalOffline": 0,
    "cachegroups": [
       {
          "online": 8,
          "offline": 0,
          "name": "us-co-denver"
       },
       {
          "online": 7,
          "offline": 0,
          "name": "us-de-newcastle"
       }
    ]
 },
 "version": "1.1"
}
```

GET /api/1.1/cdns/:name/health.json

Retrieves the health of all locations (cache groups) for a given CDN.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
name	yes	

Response Properties

Parameter	Type	Description
`totalOnline`	int	Total number of online caches across the specified CDN.
`totalOffline`	int	Total number of offline caches across the specified CDN.
`cachegroups`	array	A collection of cache groups.
`>online`	int	The number of online caches for the cache group
`>offline`	int	The number of offline caches for the cache group.
• `name`	string	Cache group name.

Response Example

```
{
 "response": {
    "totalOnline": 148,
    "totalOffline": 0,
    "cachegroups": [
       {
          "online": 8,
          "offline": 0,
          "name": "us-co-denver"
       },
       {
          "online": 7,
          "offline": 0,
          "name": "us-de-newcastle"
       }
    ]
 },
 "version": "1.1"
}
```

GET /api/1.1/cdns/usage/overview.json

Retrieves the high-level CDN usage metrics.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`currentGbps`	number	
`tps`	int	
`maxGbps`	int	

Response Example

```
{
 "response": {
    "currentGbps": 149.368167,
    "tps": 36805,
    "maxGbps": 3961
 },
 "version": "1.1"
}
```

GET /api/1.1/cdns/capacity.json

Retrieves the aggregate capacity percentages of all locations (cache groups) for a given CDN.

Response Properties

Parameter	Type	Description
`availablePercent`	number	
`unavailablePercent`	number	
`utilizedPercent`	number	
`maintenancePercent`	number	

Response Example

```
{
 "response": {
    "availablePercent": 89.0939840205533,
    "unavailablePercent": 0,
    "utilizedPercent": 10.9060020300395,
    "maintenancePercent": 0.0000139494071146245
 },
 "version": "1.1"
}
```

Routing

GET /api/1.1/cdns/routing.json

Authentication Required: Yes

Retrieves the aggregate routing percentages of all locations (cache groups) for a given CDN.

Response Properties

Parameter	Type	Description
`staticRoute`	number	Used pre-configured DNS entries.
`miss`	number	No location available for client IP.
`geo`	number	Used 3rd party geo-IP mapping.
`err`	number	Error localizing client IP.
`cz`	number	Used Coverage Zone geo-IP mapping.
`dsr`	number	Overflow traffic sent to secondary CDN.

Response Example

```
{
  "response": {
     "staticRoute": 0,
     "miss": 0,
     "geo": 37.8855391018869,
     "err": 0,
     "cz": 62.1144608981131,
     "dsr": 0
  },
  "version": "1.1"
 }
```

Metrics

GET /api/1.1/cdns/metric_types/:metric/start_date/:start/end_date/:end.json

Authentication Required: Yes

Retrieves edge metrics of one or all locations (cache groups).

Request Route Parameters

Name	Required	Description
`metric_type`	yes	ooff, origin_tps
`start`	yes	UNIX time, yesterday, now
`end`	yes	UNIX time, yesterday, now

Response Properties

Parameter	Type	Description
`stats`	hash	
`>count`	string	
`>98thPercentile`	string	
`>min`	string	
`>max`	string	
`>5thPercentile`	string	
`>95thPercentile`	string	
`>mean`	string	
`>sum`	string	
`data`	array	
`>time`	int	
`>value`	number	
`label`	string	

Response Example

```
{
 "response": [
    {
```

(continues on next page)

(continued from previous page)

```
        "stats": {
           "count": 1,
           "98thPercentile": 1668.03,
           "min": 1668.03,
           "max": 1668.03,
           "5thPercentile": 1668.03,
           "95thPercentile": 1668.03,
           "mean": 1668.03,
           "sum": 1668.03
        },
        "data": [
           [
              1425135900000,
              1668.03
           ],
           [
              1425136200000,
              null
           ]
        ],
        "label": "Origin TPS"
     }
  ],
  "version": "1.1"
}
```

Domains

GET /api/1.1/cdns/domains.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`profileId`	string	
`parameterId`	string	
`profileName`	string	
`profileDescription`	string	
`domainName`	string	

Response Example

```
{
  "response": [
     {
        "profileId": "5",
        "parameterId": "404",
        "profileName": "CR_FOO",
        "profileDescription": "Comcast Content Router for foo.domain.net",
```

(continues on next page)

(continued from previous page)

```
            "domainName": "foo.domain.net"
        },
        {
            "profileId": "8",
            "parameterId": "405",
            "profileName": "CR_BAR",
            "profileDescription": "Comcast Content Router for bar.domain.net",
            "domainName": "bar.domain.net"
        }
    ],
    "version": "1.1"
}
```

Topology

GET /api/1.1/cdns/:cdn_name/configs.json

Retrieves CDN config information.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`cdn_name`	yes	Your cdn name or, all

Response Properties

Parameter	Type	Description
`id`	string	
`value`	string	
`name`	string	
`config_file`	string	

Response Example

```
TBD
```

GET /api/1.1/cdns/:name/configs/monitoring.json

Retrieves CDN monitoring information.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`name`	yes	

Response Properties

Parameter	Type	Description
`trafficServers`	array	A collection of Traffic Servers.
`>profile`	string	
`>ip`	string	
`>status`	string	
`>cacheGroup`	string	
`>ip6`	string	
`>port`	int	
`>hostName`	string	
`>fqdn`	string	
`>interfaceName`	string	
`>type`	string	
`>hashId`	string	
`cacheGroups`	array	A collection of cache groups.
`>coordinates`	hash	
`>>longitude`	number	
`>>latitude`	number	
`>name`	string	
`config`	hash	
`>hack.ttl`	int	
`>tm.healthParams.polling.url`	string	
`>tm.dataServer.polling.url`	string	
`>health.timepad`	int	
`>tm.polling.interval`	int	
`>health.threadPool`	int	
`>health.polling.interval`	int	
`>health.event-count`	int	
`>tm.crConfig.polling.url`	number	
`>CDN_name`	number	
`trafficMonitors`	array	A collection of Traffic Monitors.
`>profile`	string	
`>location`	string	
`>ip`	string	
`>status`	string	
`>ip6`	string	
`>port`	int	
`>hostName`	string	
`>fqdn`	string	
`deliveryServices`	array	A collection of delivery services.
`>xmlId`	string	
`>totalTpsThreshold`	int	
`>status`	string	
`>totalKbpsThreshold`	int	
`profiles`	array	A collection of profiles.
`>parameters`	hash	

Continued on next page

Table 1 – continued from previous page

Parameter	Type	Description
`>>health.connection.timeout`	int	
`>>health.polling.url`	string	
`>>health.threshold.queryTime`	int	
`>>history.count`	int	
`>>health.threshold.availableBandwidthInKbps`	string	
`>>health.threshold.loadavg`	string	
`>name`	string	
`>type`	string	

Response Example

```
TBD
```

GET /api/1.1/cdns/:name/configs/routing.json

Retrieves CDN routing information.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`name`	yes	

Response Properties

Parameter	Type	Description
`trafficServers`	array	A collection of Traffic Servers.
`>profile`	string	
`>ip`	string	
`>status`	string	
`>cacheGroup`	string	
`>ip6`	string	
`>port`	int	
`>deliveryServices`	array	
`>>xmlId`	string	
`>>remaps`	array	
`>>hostName`	string	
`>fqdn`	string	
`>interfaceName`	string	
`>type`	string	
`>hashId`	string	
`stats`	hash	
`>trafficOpsPath`	string	
`>cdnName`	string	
`>trafficOpsVersion`	string	

Continued on next page

Table 2 – continued from previous page

Parameter	Type	Description
`>trafficOpsUser`	string	
`>date`	int	
`>trafficOpsHost`	string	
`cacheGroups`	array	A collection of cache groups.
`>coordinates`	hash	
`>>longitude`	number	
`>>latitude`	number	
`>name`	string	
`config`	hash	
`>tld.soa.admin`	string	
`>tcoveragezone.polling.interval`	int	
`>geolocation.polling.interval`	int	
`>tld.soa.expire`	int	
`>coveragezone.polling.url`	string	
`>tld.soa.minimum`	int	
`>geolocation.polling.url`	string	
`>domain_name`	string	
`>tld.ttls.AAAA`	int	
`>tld.soa.refresh`	int	
`>tld.ttls.NS`	int	
`>tld.ttls.SOA`	int	
`>geolocation6.polling.interval`	int	
`>tld.ttls.A`	int	
`>tld.soa.retry`	int	
`>geolocation6.polling.url`	string	
`trafficMonitors`	array	A collection of Traffic Monitors.
`>profile`	string	
`>location`	string	
`>ip`	string	
`>status`	string	
`>ip6`	string	
`>port`	int	
`>hostName`	string	
`>fqdn`	string	
`deliveryServices`	array	A collection of delivery services.
`>xmlId`	string	
`>ttl`	int	
`>geoEnabled`	string	
`>coverageZoneOnly`	boolean	
`>matchSets`	array	
`>>protocol`	string	
`>>matchList`	array	
`>>>regex`	string	
`>>>matchType`	string	
`>bypassDestination`	hash	
`>>maxDnsIpsForLocation`	int	
`>>ttl`	int	
`>>type`	string	
`>ttls`	hash	

Continued on next page

Table 2 – continued from previous page

Parameter	Type	Description
`>>A`	int	
`>>SOA`	int	
`>>NS`	int	
`>>AAAA`	int	
`>missCoordinates`	hash	
`>>longitude`	number	
`>>latitude`	number	
`>soa`	hash	
`>>admin`	string	
`>>retry`	int	
`>>minimum`	int	
`>>refresh`	int	
`>>expire`	int	
`trafficRouters`	hash	
`>profile`	int	
`>location`	string	
`>ip`	string	
`>status`	string	
`>ip6`	string	
`>port`	int	
`>hostName`	string	
`>fqdn`	string	
`>apiPort`	int	

Response Example

:: TBD

DNSSEC Keys

GET /api/1.1/cdns/name/:name/dnsseckeys.json

Gets a list of dnsseckeys for CDN and all associated Delivery Services. Before returning response to user, check to make sure keys aren't expired. If they are expired, generate new ones. Before returning response to user, make sure dnssec keys for all delivery services exist. If they don't exist, create them.

Authentication Required: Yes

Role Required: Admin

Request Route Parameters

Name	Required	Description
`name`	yes	

Response Properties

Parameter	Type	Description
`cdn name/ds xml_id`	string	identifier for ds or cdn
`>zsk/ksk`	array	collection of zsk/ksk data
`>>ttl`	string	time-to-live for dnssec requests
`>>inceptionDate`	string	epoch timestamp for when the keys were created
`>>expirationDate`	string	epoch timestamp representing the expiration of the keys
`>>private`	string	encoded private key
`>>public`	string	encoded public key
`>>name`	string	domain name
`version`	string	API version

Response Example

```
{
  "response": {
    "cdn1": {
      "zsk": {
        "ttl": "60",
        "inceptionDate": "1426196750",
        "private": "zsk private key",
        "public": "zsk public key",
        "expirationDate": "1428788750",
        "name": "foo.kabletown.com."
      },
      "ksk": {
        "name": "foo.kabletown.com.",
        "expirationDate": "1457732750",
        "public": "ksk public key",
        "private": "ksk private key",
        "inceptionDate": "1426196750",
        "ttl": "60"
      }
    },
    "ds-01": {
      "zsk": {
        "ttl": "60",
        "inceptionDate": "1426196750",
        "private": "zsk private key",
        "public": "zsk public key",
        "expirationDate": "1428788750",
        "name": "ds-01.foo.kabletown.com."
      },
      "ksk": {
        "name": "ds-01.foo.kabletown.com.",
        "expirationDate": "1457732750",
        "public": "ksk public key",
        "private": "ksk private key",
        "inceptionDate": "1426196750"
      }
    },
    ... repeated for each ds in the cdn
  },
  "version": "1.1"
}
```

GET /api/1.1/cdns/name/:name/dnsseckeys/delete.json

Delete dnssec keys for a cdn and all associated delivery services.

Authentication Required: Yes

Role Required: Admin

Request Route Parameters

Name	Required	Description
`name`	yes	name of the CDN for which you want to delete dnssec keys

Response Properties

Parameter	Type	Description
`response`	string	success response

Response Example

```
{
  "version": "1.1",
  "response": "Successfully deleted dnssec keys for <cdn>"
}
```

POST /api/1.1/deliveryservices/dnsseckeys/generate

Generates zsk and ksk keypairs for a cdn and all associated delivery services.

Authentication Required: Yes

Role Required: Admin

Request Properties

Parameter	Type	Description
`key`	string	name of the cdn
`name`	string	domain name of the cdn
`ttl`	string	time to live
`kskExpirationDays`	string	Expiration (in days) for the key signing keys
`zskExpirationDays`	string	Expiration (in days) for the zone signing keys

Request Example

```
{
  "key": "cdn1",
  "name" "ott.kabletown.com",
  "ttl": "60",
  "kskExpirationDays": "365",
```

(continues on next page)

(continued from previous page)

```
  "zskExpirationDays": "90"
}
```

Response Properties

Parameter	Type	Description
`response`	string	response string
`version`	string	API version

Response Example

```
{
  "version": "1.1",
  "response": "Successfully created dnssec keys for cdn1"
}
```

Change Logs

GET /api/1.1/logs.json

Response Properties

Parameter	Type	Description
`ticketNum`	string	Optional field to cross reference with any bug tracking systems
`level`	string	Log categories for each entry, examples: 'UICHANGE', 'OPER', 'APICHANGE'.
`lastUpdated`	string	Local unique identifier for the Log
`user`	string	Current user who made the change that was logged
`id`	string	Local unique identifier for the Log entry
`message`	string	Log detail about what occurred

Response Example

```
{
 "response": [
    {
       "ticketNum": null,
       "level": "OPER",
       "lastUpdated": "2015-02-04 22:59:13",
       "user": "mtorlu9137e",
       "id": "22661",
       "message": "Snapshot CRConfig created."
    },
    {
       "ticketNum": null,
       "level": "APICHANGE",
       "lastUpdated": "2015-02-03 17:04:20",
       "user": "admin",
       "id": "22658",
       "message": "Update server odol-atsec-nyc-23.kbaletown.net␣
↪status=REPORTED"
    },
```

(continues on next page)

(continued from previous page)

```
 ],
 "version": "1.1"
}
```

GET /api/1.1/logs/:days/days.json

Request Route Parameters

Name	Required	Description
`days`	yes	Number of days.

Response Properties

Parameter	Type	Description
`ticketNum`	string	
`level`	string	
`lastUpdated`	string	
`user`	string	
`id`	string	
`message`	string	

Response Example

```
{
 "response": [
    {
       "ticketNum": null,
       "level": "OPER",
       "lastUpdated": "2015-02-04 22:59:13",
       "user": "mtorlu9137e",
       "id": "22661",
       "message": "Snapshot CRConfig created."
    },
    {
       "ticketNum": null,
       "level": "APICHANGE",
       "lastUpdated": "2015-02-03 17:04:20",
       "user": "admin",
       "id": "22658",
       "message": "Update server odol-atsec-nyc-23.kabletown.net␣
↪status=REPORTED"
    }
 ],
 "version": "1.1"
}
```

GET /api/1.1/logs/newcount.json

Response Properties

Parameter	Type	Description
`newLogcount`	string	

Response Example

```
{
 "response": {
    "newLogcount": 0
 },
 "version": "1.1"
}
```

Delivery Service

GET /api/1.1/deliveryservices.json

Retrieves all delivery services. See also Using Traffic Ops - Delivery Service.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`active`	bool	true if active, false if inactive (inact).
`cacheurl`	string	Cache URL rule to apply to this delivery service.
`protocol`	string	• 0: serve with http:// at EDGE • 1: serve with https:// at EDGE • 2: serve with both http:// and https:// at EDGE
`ccrDnsTtl`	string	The TTL of the DNS response for A or AAAA queries requesting the IP address of the tr. host.
`checkPath`	string	The path portion of the URL to check this deliveryservice for health.
`dnsBypassIp`	string	The IPv4 IP to use for bypass on a DNS deliveryservice - bypass starts when serving more than the globalMaxMbps traffic on this deliveryservice.

Continued on next page

Table 3 – continued from previous page

Parameter	Type	Description
`dnsBypassIp6`	string	The IPv6 IP to use for bypass on a DNS deliveryservice - bypass starts when serving more than the globalMaxMbps traffic on this deliveryservice.
`dnsBypassTtl`	string	The TTL of the DNS bypass response.
`dscp`	string	The Differentiated Services Code Point (DSCP) with which to mark downstream (EDGE -> customer) traffic.
`edgeHeaderRewrite`	string	The EDGE header rewrite actions to perform.
`geoLimit`	string	• 0: None - no limitations • 1: Only route on CZF file hit • 2: Only route on CZF hit or when from USA Note that this does not prevent access to content or makes content secure; it just prevents routing to the content by Traffic Router.
`globalMaxMbps`	string	The maximum global bandwidth allowed on this deliveryservice. If exceeded, the traffic routes to the dnsByPassIp* for DNS deliveryservices and to the httpBypassFqdn for HTTP deliveryservices.
`globalMaxTps`	string	The maximum global transactions per second allowed on this deliveryservice. When this is exceeded traffic will be sent to the dnsByPassIp* for DNS deliveryservices and to the httpBypassFqdn for HTTP deliveryservices
`headerRewrite`	string	The EDGE header rewrite actions to perform.
`httpBypassFqdn`	string	The HTTP destination to use for bypass on an HTTP deliveryservice - bypass starts when serving more than the globalMaxMbps traffic on this deliveryservice.
`id`	string	The deliveryservice id (database row number).

Continued on next page

Table 3 – continued from previous page

Parameter	Type	Description
`infoUrl`	string	Use this to add a URL that points to more information about that deliveryservice.
`ipv6RoutingEnabled`	bool	false: send IPv4 address of Traffic Router to client on HTTP type del.
`longDesc`	string	Description field 1.
`longDesc1`	string	Description field 2.
`longDesc2`	string	Description field 2.
`matchList`	array	Array of matchList hashes.
`>>type`	string	The type of MatchList (one of :ref:to-api-types use_in_table='regex').
`>>setNumber`	string	The set Number of the matchList.
`>>pattern`	string	The regexp for the matchList.
`maxDnsAnswers`	string	The maximum number of IPs to put in a A/AAAA response for a DNS deliveryservice (0 means all available).
`missLat`	string	The latitude to use when the client cannot be found in the CZF or the Geo lookup.
`missLong`	string	The longitude to use when the client cannot be found in the CZF or the Geo lookup.
`midHeaderRewrite`	string	The MID header rewrite actions to perform.
`multiSiteOrigin`	string	Is the Multi Site Origin feature enabled for this delivery service. See rl-mulit-site-origin
`orgServerFqdn`	string	The origin server base URL (FQDN when used in this instance, includes the protocol (http:// or https://) for use in retrieving content from the origin server.
`profileDescription`	string	The description of the Traffic Router Profile with which this deliveryservice is associated.
`profileName`	string	The name of the Traffic Router Profile with which this deliveryservice is associated.

Continued on next page

Table 3 – continued from previous page

Parameter	Type	Description
qstringIgnore	string	• 0: no special query string handling; it is for use in the cache-key and pass up to origin. • 1: ignore query string in cache-key, but pass it up to parent and or origin. • 2: drop query string at edge, and do not use it in the cache-key.
regexRemap	string	Regex Remap rule to apply to this delivery service at the Edge tier.
remapText	string	Additional raw remap line text.
signed	bool	• false: token based auth (see :ref:token-based-auth) is not enabled for this deliveryservice. • true: token based auth is enabled for this delivery-service.
rangeRequestHandling	string	How to treat range requests: • 0 Do not cache (ranges requested from files taht are already cached due to a non range request will be a HIT) • 1 Use the background_fetch plugin. • 2 Use the cache_range_requests plugin.
type	string	The type of this deliveryservice (one of :ref:to-api-types use_in_table='deliveryservice').
xmlId	string	Unique string that describes this deliveryservice.

Response Example

```
{
  "response": [
    {
      "active": true,
      "cacheurl": null,
      "protocol": "0",
      "ccrDnsTtl": "3600",
```

(continues on next page)

(continued from previous page)

```
      "checkPath": "/crossdomain.xml",
      "dnsBypassIp": "",
      "dnsBypassIp6": null,
      "dnsBypassTtl": null,
      "dscp": "40",
      "geoLimit": "0",
      "globalMaxMbps": "0",
      "globalMaxTps": "0",
      "headerRewrite": "add-header X-Powered-By: KABLETOWN [L]",
      "edgeHeaderRewrite": "add-header X-Powered-By: KABLETOWN [L]",
      "midHeaderRewrite": null,
      "httpBypassFqdn": "",
      "rangeRequestHandling": "0",
      "id": "12",
      "infoUrl": "",
      "ipv6RoutingEnabled": false,
      "longDesc": "long_desc",
      "longDesc1": "long_desc_1",
      "longDesc2": "long_desc_2",
      "matchList": [
        {
          "type": "HOST_REGEXP",
          "setNumber": "0",
          "pattern": ".*\\.images\\..*"
        }
      ],
      "maxDnsAnswers": "0",
      "missLat": "41.881944",
      "missLong": "-87.627778",
      "orgServerFqdn": "http://cdl.origin.kabletown.net",
      "profileDescription": "Comcast Content Router for cdn2.comcast.net",
      "profileName": "EDGE_CDN2",
      "qstringIgnore": "0",
      "remapText": null,
      "regexRemap": null,
      "signed": true,
      "type": "HTTP",
      "xmlId": "cdl-c2"
    },
    { .. },
    { .. }
  ],
  "version": "1.1"
}
```

GET /api/1.1/deliveryservices/:id.json

Retrieves a specific delivery service. See also Using Traffic Ops - Delivery Service.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`active`	bool	true if active, false if inactive (inact).
`cacheurl`	string	Cache URL rule to apply to this delivery service.
`protocol`	string	• 0: serve with http:// at EDGE • 1: serve with https:// at EDGE • 2: serve with both http:// and https:// at EDGE
`ccrDnsTtl`	string	The TTL of the DNS response for A or AAAA queries requesting the IP address of the tr. host.
`checkPath`	string	The path portion of the URL to check this deliveryservice for health.
`dnsBypassIp`	string	The IPv4 IP to use for bypass on a DNS deliveryservice - bypass starts when serving more than the globalMaxMbps traffic on this deliveryservice.
`dnsBypassIp6`	string	The IPv6 IP to use for bypass on a DNS deliveryservice - bypass starts when serving more than the globalMaxMbps traffic on this deliveryservice.
`dnsBypassTtl`	string	The TTL of the DNS bypass response.
`dscp`	string	The Differentiated Services Code Point (DSCP) with which to mark downstream (EDGE -> customer) traffic.
`edgeHeaderRewrite`	string	The EDGE header rewrite actions to perform.
`geoLimit`	string	• 0: None - no limitations • 1: Only route on CZF file hit • 2: Only route on CZF hit or when from USA Note that this does not prevent access to content or makes content secure; it just prevents routing to the content by Traffic Router.

Continued on next page

Table 4 – continued from previous page

Parameter	Type	Description
`globalMaxMbps`	string	The maximum global bandwidth allowed on this deliveryservice. If exceeded, the traffic routes to the dnsByPassIp* for DNS deliveryservices and to the httpBypassFqdn for HTTP deliveryservices.
`globalMaxTps`	string	The maximum global transactions per second allowed on this deliveryservice. When this is exceeded traffic will be sent to the dnsByPassIp* for DNS deliveryservices and to the httpBypassFqdn for HTTP deliveryservices
`headerRewrite`	string	The EDGE header rewrite actions to perform.
`httpBypassFqdn`	string	The HTTP destination to use for bypass on an HTTP deliveryservice - bypass starts when serving more than the globalMaxMbps traffic on this deliveryservice.
`id`	string	The deliveryservice id (database row number).
`infoUrl`	string	Use this to add a URL that points to more information about that deliveryservice.
`ipv6RoutingEnabled`	bool	false: send IPv4 address of Traffic Router to client on HTTP type del.
`longDesc`	string	Description field 1.
`longDesc1`	string	Description field 2.
`longDesc2`	string	Description field 2.
`matchList`	array	Array of matchList hashes.
`>>type`	string	The type of MatchList (one of :ref:to-api-types use_in_table='regex').
`>>setNumber`	string	The set Number of the matchList.
`>>pattern`	string	The regexp for the matchList.
`maxDnsAnswers`	string	The maximum number of IPs to put in a A/AAAA response for a DNS deliveryservice (0 means all available).
`missLat`	string	The latitude to use when the client cannot be found in the CZF or the Geo lookup.

Continued on next page

Table 4 – continued from previous page

Parameter	Type	Description
`missLong`	string	The longitude to use when the client cannot be found in the CZF or the Geo lookup.
`midHeaderRewrite`	string	The MID header rewrite actions to perform.
`orgServerFqdn`	string	The origin server base URL (FQDN when used in this instance, includes the protocol (http:// or https://) for use in retrieving content from the origin server.
`profileDescription`	string	The description of the Traffic Router Profile with which this deliveryservice is associated.
`profileName`	string	The name of the Traffic Router Profile with which this deliveryservice is associated.
`qstringIgnore`	string	• 0: no special query string handling; it is for use in the cache-key and pass up to origin. • 1: ignore query string in cache-key, but pass it up to parent and or origin. • 2: drop query string at edge, and do not use it in the cache-key.
`regexRemap`	string	Regex Remap rule to apply to this delivery service at the Edge tier.
`remapText`	string	Additional raw remap line text.
`signed`	bool	• false: token based auth (see :ref:token-based-auth) is not enabled for this deliveryservice. • true: token based auth is enabled for this deliveryservice.

Continued on next page

Table 4 – continued from previous page

Parameter	Type	Description
`rangeRequestHandling`	string	How to treat range requests: • 0 Do not cache (ranges requested from files taht are already cached due to a non range request will be a HIT) • 1 Use the background_fetch plugin. • 2 Use the cache_range_requests plugin.
`type`	string	The type of this deliveryservice (one of :ref:to-api-types use_in_table='deliveryservice').
`xmlId`	string	Unique string that describes this deliveryservice.

Response Example

```
{
  "response": [
    {
      "active": true,
      "cacheurl": null,
      "protocol": "0",
      "ccrDnsTtl": "3600",
      "checkPath": "/crossdomain.xml",
      "dnsBypassIp": "",
      "dnsBypassIp6": null,
      "dnsBypassTtl": null,
      "dscp": "40",
      "geoLimit": "0",
      "globalMaxMbps": "0",
      "globalMaxTps": "0",
      "headerRewrite": "add-header X-Powered-By: KABLETOWN [L]",
      "edgeHeaderRewrite": "add-header X-Powered-By: KABLETOWN [L]",
      "midHeaderRewrite": null,
      "httpBypassFqdn": "",
      "rangeRequestHandling": "0",
      "id": "12",
      "infoUrl": "",
      "ipv6RoutingEnabled": false,
      "longDesc": "long_desc",
      "longDesc1": "long_desc_1",
      "longDesc2": "long_desc_2",
      "matchList": [
        {
          "type": "HOST_REGEXP",
          "setNumber": "0",
          "pattern": ".*\\.images\\..*"
        }
      ],
      "maxDnsAnswers": "0",
```

(continues on next page)

(continued from previous page)

```
        "missLat": "41.881944",
        "missLong": "-87.627778",
        "orgServerFqdn": "http://cdl.origin.kabletown.net",
        "profileDescription": "Comcast Content Router for cdn2.comcast.net",
        "profileName": "EDGE_CDN2",
        "qstringIgnore": "0",
        "remapText": null,
        "regexRemap": null,
        "signed": true,
        "type": "HTTP",
        "xmlId": "cdl-c2"
      }
    ],
    "version": "1.1"
}
```

Health

GET /api/1.1/deliveryservices/:id/capacity.json

Retrieves the capacity percentages of a delivery service.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
id	yes	delivery service id.

Response Properties

Parameter	Type	Description
`availablePercent`	number	The percentage of server capacity assigned to the delivery service that is available.
`unavailablePercent`	number	The percentage of server capacity assigned to the delivery service that is unavailable.
`utilizedPercent`	number	The percentage of server capacity assigned to the delivery service being used.
`maintenancePercent`	number	The percentage of server capacity assigned to the delivery service that is down for maintenance.

Response Example

```
{
 "response": {
    "availablePercent": 89.0939840205533,
    "unavailablePercent": 0,
    "utilizedPercent": 10.9060020300395,
    "maintenancePercent": 0.0000139494071146245
 },
 "version": "1.1"
}
```

GET /api/1.1/deliveryservices/:id/routing.json

Retrieves the routing method percentages of a delivery service.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
id	yes	delivery service id.

Response Properties

Parameter	Type	Description
`staticRoute`	number	The percentage of Traffic Router responses for this deliveryservice satisfied with pre-configured DNS entries.
`miss`	number	The percentage of Traffic Router responses for this deliveryservice that were a miss (no location available for client IP).
`geo`	number	The percentage of Traffic Router responses for this deliveryservice satisfied using 3rd party geo-IP mapping.
`err`	number	The percentage of Traffic Router requests for this deliveryservice resulting in an error.
`cz`	number	The percentage of Traffic Router requests for this deliveryservice satisfied by a CZF hit.
`dsr`	number	The percentage of Traffic Router requests for this deliveryservice satisfied by sending the client to the overflow CDN.

Response Example

```
{
 "response": {
    "staticRoute": 0,
    "miss": 0,
    "geo": 37.8855391018869,
    "err": 0,
    "cz": 62.1144608981131,
    "dsr": 0
 },
 "version": "1.1"
}
```

Metrics

GET /api/1.1/deliveryservices/:id/edge/metric_types/:metric/start_date/:start/end_date/:end/\ interval/:interval/window_start/:window_start/window_end/:window_end.json

Retrieves edge summary metrics of all cache groups for a delivery service.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`id`	yes	The delivery service id.
`metric`	yes	One of the following: "kbps", "tps_total", "tps_2xx", "tps_3xx", "tps_4xx", "tps_5xx".
`start`	yes	UNIX time, yesterday, now.
`end`	yes	UNIX time, yesterday, now.
`interval`	yes	> 10
`window_start`	yes	UNIX time, yesterday, now.
`window_end`	yes	UNIX time, yesterday, now.

Request Query Parameters

Name	Required	Description
`summary`	no	Flag used to return summary metrics only.

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`ninetyFifth`	number	
`average`	int	
`min`	number	
`max`	number	
`total`	number	

Response Example

```
{
 "response": {
    "ninetyFifth": 183982091.479,
    "average": 97444798,
    "min": 31193860.46233,
    "max": 205772883.28367,
    "total": 3643217414091.13
 },
 "version": "1.1"
}
```

GET /api/1.1/usage/deliveryservices/:ds/cachegroups/:name/metric_types/:metric/start_date/:start_date/end_date/:end_date/interval/:interval.json

Retrieves edge metrics of one or all locations (cache groups) for a delivery service.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`id`	yes	The delivery service id.
`cache_group_name`	yes	name, all.
`usage_type`	yes	One of the following: "kbps", "tps_total", "tps_2xx", "tps_3xx", "tps_4xx", "tps_5xx".
`start`	yes	UNIX time, yesterday, now.
`end`	yes	UNIX time, yesterday, now.
`interval`	yes	> 10

Response Properties

Parameter	Type	Description
`deliveryServiceName`	string	
`statName`	string	
`deliveryServiceId`	string	
`interval`	int	
`series`	array	
`>>timeBase`	int	
`>>samples`	array	
`end`	string	
`elapsed`	number	
`cdnName`	string	
`hostName`	string	
`summary`	hash	
>‘‘ninetyFifth‘‘	number	
>‘‘average‘‘	int	
>‘‘min‘‘	number	
>‘‘max‘‘	number	
>‘‘total‘‘	number	
`cacheGroupName`	string	
`start`	string	

Response Example

```
TBD
```

GET /api/1.1/cdns/peakusage/:peak_usage_type/deliveryservice/:ds/cachegroup/:name/start_date/:start/\ end_date/:end/interval/:interval.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`TotalGBytesServedSinceStart`	number	
`>>item`	number	
`>>item`	number	
`>>item`	number	
`>>item`	number	
`>>item`	number	
`>>item`	number	

Response Example

```
TBD
```

GET /api/1.1/deliveryservices/:id/:server_type/metrics/:metric_type/:start/:end.json

Retrieves detailed and summary metrics for MIDs or EDGEs for a delivery service.

Authentication Required: No

Request Route Parameters

Name	Required	Description
`id`	yes	The delivery service id.
`server_type`	yes	EDGE or MID.
`metric_type`	yes	One of the following: "kbps", "tps_total", "tps_2xx", "tps_3xx", "tps_4xx", "tps_5xx".
`start`	yes	UNIX time, yesterday, now.
`end`	yes	UNIX time, yesterday, now.

Response Properties

Parameter	Type	Description
`stats`	hash	
`>>count`	int	
`>>98thPercentile`	number	
`>>min`	number	
`>>max`	number	
`>>5thPercentile`	number	
`>>95thPercentile`	number	
`>>median`	number	
`>>mean`	number	
`>>stddev`	number	
`>>sum`	number	
`data`	array	
`>>item`	array	
`>>time`	number	
`>>value`	number	
`label`	string	

Response Example

```
{
 "response": [
    {
       "stats": {
          "count": 988,
          "98thPercentile": 16589105.55958,
          "min": 3185442.975,
          "max": 17124754.257,
          "5thPercentile": 3901253.95445,
          "95thPercentile": 16013210.034,
          "median": 8816895.576,
          "mean": 8995846.31741194,
          "stddev": 3941169.83683573,
          "sum": 333296106.060112
       },
       "data": [
          [
             1414303200000,
             12923518.466
          ],
          [
             1414303500000,
             12625139.65
          ]
       ],
       "label": "MID Kbps"
    }
 ],
 "version": "1.1"
}
```

Server

GET /api/1.1/deliveryserviceserver.json

Authentication Required: Yes

Request Query Parameters

Name	Required	Description
`page`	no	The page number for use in pagination.
`limit`	no	For use in limiting the result set.

Response Properties

Parameter	Type	Description
`lastUpdated`	array	
`server`	string	
`deliveryService`	string	

Response Example

```
{
 "page": 2,
 "orderby": "deliveryservice",
 "response": [
    {
       "lastUpdated": "2014-09-26 17:53:43",
       "server": "20",
       "deliveryService": "1"
    },
    {
       "lastUpdated": "2014-09-26 17:53:44",
       "server": "21",
       "deliveryService": "1"
    },
 ],
 "version": "1.1",
 "limit": 2
}
```

SSL Keys

GET /api/1.1/deliveryservices/xmlId/:xmlid/sslkeys.json

Authentication Required: Yes

Role Required: Admin

Request Route Parameters

Name	Required	Description
`xmlId`	yes	xml_id of the desired delivery service

Request Query Parameters

Name	Required	Description
`version`	no	The version number to retrieve

Response Properties

Parameter	Type	Description
`crt`	string	base64 encoded crt file for delivery service
`csr`	string	base64 encoded csr file for delivery service
`key`	string	base64 encoded private key file for delivery service
`businessUnit`	string	The business unit entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`city`	string	The city entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`organization`	string	The organization entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`hostname`	string	The hostname entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`country`	string	The country entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`state`	string	The state entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`version`	string	The version of the certificate record in Riak

Response Example

```
{
  "version": "1.1",
  "response": {
    "certificate": {
      "crt": "crt",
      "key": "key",
      "csr": "csr"
    },
    "businessUnit": "CDN_Eng",
    "city": "Denver",
    "organization": "KableTown",
    "hostname": "foober.com",
    "country": "US",
    "state": "Colorado",
    "version": "1"
  }
}
```

GET /api/1.1/deliveryservices/hostname/:hostname/sslkeys.json

Authentication Required: Yes

Role Required: Admin

Request Route Parameters

Name	Required	Description
`hostname`	yes	pristine hostname of the desired delivery service

Request Query Parameters

Name	Required	Description
`version`	no	The version number to retrieve

Response Properties

Parameter	Type	Description
`crt`	string	base64 encoded crt file for delivery service
`csr`	string	base64 encoded csr file for delivery service
`key`	string	base64 encoded private key file for delivery service
`businessUnit`	string	The business unit entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`city`	string	The city entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`organization`	string	The organization entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`hostname`	string	The hostname entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`country`	string	The country entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`state`	string	The state entered by the user when generating certs. Field is optional and if not provided by the user will not be in response
`version`	string	The version of the certificate record in Riak

Response Example

```
{
  "version": "1.1",
  "response": {
    "certificate": {
      "crt": "crt",
      "key": "key",
      "csr": "csr"
    },
    "businessUnit": "CDN_Eng",
    "city": "Denver",
    "organization": "KableTown",
    "hostname": "foober.com",
    "country": "US",
    "state": "Colorado",
    "version": "1"
  }
}
```

GET /api/1.1/deliveryservices/xmlId/:xmlid/sslkeys/delete.json

Authentication Required: Yes

Role Required: Admin

Request Route Parameters

Name	Required	Description
`xmlId`	yes	xml_id of the desired delivery service

Request Query Parameters

Name	Required	Description
`version`	no	The version number to retrieve

Response Properties

Parameter	Type	Description
`response`	string	success response

Response Example

```
{
  "version": "1.1",
  "response": "Successfully deleted ssl keys for <xml_id>"
}
```

POST /api/1.1/deliveryservices/sslkeys/generate

Generates SSL crt, csr, and private key for a delivery service

Authentication Required: Yes Role Required: Admin

Response Content Type: application/json

Request Properties

Parameter	Type	Description
`key`	string	xml_id of the delivery service
`version`	string	version of the keys being generated
`hostname`	string	the *pristine hostname* of the delivery service
`country`	string	
`state`	string	
`city`	string	
`org`	string	
`unit`	boolean	

Request Example

```
{
  "key": "ds-01",
  "businessUnit": "CDN Engineering",
  "version": "3",
  "hostname": "tr.ds-01.ott.kabletown.com",
```

(continues on next page)

(continued from previous page)

```
  "certificate": {
    "key": "some_key",
    "csr": "some_csr",
    "crt": "some_crt"
  },
  "country": "US",
  "organization": "Kabletown",
  "city": "Denver",
  "state": "Colorado"
}
```

Response Properties

Parameter	Type	Description
`response`	string	response string
`version`	string	API version

Response Example

```
{
  "version": "1.1",
  "response": "Successfully created ssl keys for ds-01"
}
```

POST /api/1.1/deliveryservices/sslkeys/add

Allows user to add SSL crt, csr, and private key for a delivery service

Authentication Required: Yes Role Required: Admin

Request Properties

Parameter	Type	Description
`key`	string	xml_id of the delivery service
`version`	string	version of the keys being generated
`csr`	string	
`crt`	string	
`key`	string	

Request Example

```
{
  "key": "ds-01",
  "version": "1",
  "certificate": {
    "key": "some_key",
    "csr": "some_csr",
    "crt": "some_crt"
  }
}
```

Response Properties

Parameter	Type	Description
`response`	string	response string
`version`	string	API version

Response Example

```
{
  "version": "1.1",
  "response": "Successfully added ssl keys for ds-01"
}
```

hwinfo

GET /api/1.1/hwinfo.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`serverId`	string	Local unique identifier for this specific server's hardware info
`serverHostName`	string	Hostname for this specific server's hardware info
`lastUpdated`	string	The Time and Date for the last update for this server.
`val`	string	Freeform value used to track anything about a server's hardware info
`description`	string	Freeform description for this specific server's hardware info

Response Example

```
{
 "response": [
    {
       "serverId": "odol-atsmid-cen-09",
       "lastUpdated": "2014-05-27 09:06:02",
       "val": "D1S4",
       "description": "Physical Disk 0:1:0"
    },
    {
       "serverId": "odol-atsmid-cen-09",
       "lastUpdated": "2014-05-27 09:06:02",
       "val": "D1S4",
       "description": "Physical Disk 0:1:1"
    }
 ],
 "version": "1.1"
}
```

Parameter

GET /api/1.1/parameters.json

Authentication Required: Yes

Return Values

Parameter	Type	Description
`last_updated`	string	The Time / Date this server entry was last updated
`value`	string	The parameter value
`name`	string	The parameter name
`config_file`	string	The parameter config_file

Response Example

```
{
 "response": [
    {
       "last_updated": "2012-09-17 21:41:22",
       "value": "foo.bar.net",
       "name": "domain_name",
       "config_file": "FooConfig.xml"
    },
    {
       "last_updated": "2012-09-17 21:41:22",
       "value": "0,1,2,3,4,5,6",
       "name": "Drive_Letters",
       "config_file": "storage.config"
    },
    {
       "last_updated": "2012-09-17 21:41:22",
       "value": "STRING __HOSTNAME__",
       "name": "CONFIG proxy.config.proxy_name",
       "config_file": "records.config"
    }
 ],
 "version": "1.1"
}
```

GET /api/1.1/parameters/profile/:profile_name.json

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`profile_name`	yes	

Return Values

Parameter	Type	Description
`last_updated`	string	The Time / Date this server entry was last updated
`value`	string	The parameter value
`name`	string	The parameter name
`config_file`	string	The parameter config_file

Response Example

```
{
 "response": [
    {
       "last_updated": "2012-09-17 21:41:22",
       "value": "foo.bar.net",
       "name": "domain_name",
       "config_file": "FooConfig.xml"
    },
    {
       "last_updated": "2012-09-17 21:41:22",
       "value": "0,1,2,3,4,5,6",
       "name": "Drive_Letters",
       "config_file": "storage.config"
    },
    {
       "last_updated": "2012-09-17 21:41:22",
       "value": "STRING __HOSTNAME__",
       "name": "CONFIG proxy.config.proxy_name",
       "config_file": "records.config"
    }
 ],
 "version": "1.1"
}
```

Physical Location

GET /api/1.1/phys_locations.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
region	string	
poc	string	
name	string	
comments	string	
phone	string	
state	string	
email	string	
city	string	
zip	string	
id	string	
address	string	
shortName	string	

Response Example

```
{
 "response": [
    {
       "region": "Mile High",
```

(continues on next page)

(continued from previous page)

```
        "poc": "Jane Doe",
        "name": "Albuquerque",
        "comments": "Albuquerque",
        "phone": "(123) 555-1111",
        "state": "NM",
        "email": "jane.doe@email.com",
        "city": "Albuquerque",
        "zip": "87107",
        "id": "2",
        "address": "123 East 3rd St",
        "shortName": "Albuquerque"
     },
     {
        "region": "Chicago",
        "poc": "John Doe",
        "name": "Chicago",
        "comments": "",
        "phone": "(321) 555-1111",
        "state": "IL",
        "email": "john.doe@email.com",
        "city": "Chicago",
        "zip": "60636",
        "id": "3",
        "address": "123 East 4th Street",
        "shortName": "chicago"
     }
 ],
 "version": "1.1"
}
```

GET /api/1.1/phys_locations/trimmed.json

Authentication Required: Yes

Response Messages

Response Properties

Parameter	Type	Description
name	array	

Response Example

```
{
 "response": [
     {
        "name": "Albuquerque"
     },
     {
        "name": "Ashburn"
     }
 ],
```

(continues on next page)

(continued from previous page)

```
 "version": "1.1"
}
```

Profiles

GET /api/1.1/profiles

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`lastUpdated`	array	The Time / Date this server entry was last updated
`name`	string	The name for the profile
`id`	string	Primary key
`description`	string	The description for the profile

Response Example

```
TBD
```

GET /api/1.1/profiles/trimmed.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`alerts`	array	
`>level`	string	
`>text`	string	
`version`	string	

Response Example

```
TBD
```

Redis

Note: The redis documentation needs a thorough review!

GET /api/1.1/traffic_monitor/stats.json

Authentication Required: Yes

Response Content Type: application/json

Response Messages

```
HTTP Status Code: 200
Reason: Success
```

Response Properties

Parameter	Type	Description
`aaData`	array	

Response Example

```
{
 "aaData": [
    [
       "0",
       "ALL",
       "ALL",
       "ALL",
       "true",
       "ALL",
       "142035",
       "172365661.85"
    ],
    [
       1,
       "EDGE1_TOP_421_PSPP",
       "odol-atsec-atl-03",
       "us-ga-atlanta",
       "1",
       "REPORTED",
       "596",
       "923510.04",
       "69.241.82.126"
    ]
 ],
 "version": "1.1"
}
```

GET /api/1.1/redis/stats.json

Authentication Required: Yes

Response Content Type: application/json

Response Messages

```
HTTP Status Code: 200
Reason: Success
```

Response Properties

Parameter	Type	Description
number	array	
what	string	
which	string	
interval	string	
elapsed	string	
end	string	
start	string	

Response Example

```
{
 "number": -1,
 "what": null,
 "which": null,
 "interval": " 10 seconds ",
 "elapsed": "0.11271 (0.112065) ",
 "end": "Thu Jan  1 00:00:00 1970",
 "start": "Thu Jan  1 00:00:00 1970"
}
```

GET /api/1.1/redis/info/:host_name.json

Authentication Required: Yes

Request Route Parameters

Parameter	Type	Description
host_name	string	

Request Example

Response Content Type: application/json

Response Messages

```
HTTP Status Code: 200
Reason: Success
```

Response Properties

Parameter	Type	Description
Server	hash	
>redis_build_id	string	
>config_file	string	
>uptime_in_seconds	string	
>hz	string	
>os	string	
>redis_git_sha1	string	

Continued on next page

Table 5 – continued from previous page

Parameter	Type	Description
>redis_version	string	
>tcp_port	string	
>redis_git_dirty	string	
>redis_mode	string	
>run_id	string	
>uptime_in_days	string	
>gcc_version	string	
>arch_bits	string	
>lru_clock	string	
>multiplexing_api	string	
Keyspace	string	
>db0	string	
slowlog	array	
Persistence	hash	
>rdb_bgsave_in_progress	string	
>loading	string	
>rdb_current_bgsave_time_sec	string	
>aof_enabled	string	
>rdb_last_bgsave_time_sec	string	
>aof_last_rewrite_time_sec	string	
>aof_last_write_status	string	
>rdb_last_bgsave_status	string	
>aof_last_bgrewrite_status	string	
>aof_current_rewrite_time_sec	string	
>aof_rewrite_scheduled	string	
>aof_rewrite_in_progress	string	
>rdb_last_save_time	string	
>rdb_changes_since_last_save	string	
slowlen	int	
CPU	hash	
>used_cpu_user	string	
>used_cpu_sys	string	
>used_cpu_user_children	string	
>used_cpu_sys_children	string	
Memory	string	
>used_memory_lua	string	
>mem_allocator	string	
>used_memory_human	string	
>used_memory_peak_human	string	
>used_memory_peak	string	
>used_memory_rss	string	
>mem_fragmentation_ratio	string	
>used_memory	string	
Replication	hash	
>repl_backlog_first_byte_offset	string	
>repl_backlog_active	string	
>repl_backlog_histlen	string	
>repl_backlog_size	string	
>role	string	

Continued on next page

Table 5 – continued from previous page

Parameter	Type	Description
`>master_repl_offset`	string	
`>connected_slaves`	string	
`Clients`	hash	
`>client_biggest_input_buf`	string	
`>client_longest_output_list`	string	
`>blocked_clients`	string	
`>connected_clients`	string	
`Stats`	hash	
`>latest_fork_usec`	string	
`>rejected_connections`	string	
`>sync_partial_ok`	string	
`>pubsub_channels`	string	
`>instantaneous_ops_per_sec`	string	
`>total_connections_received`	string	
`>pubsub_patterns`	string	
`>sync_full`	string	
`>keyspace_hits`	string	
`>keyspace_misses`	string	
`>total_commands_processed`	string	
`>expired_keys`	string	
`>sync_partial_err`	string	

Response Example

```
{
 "Server": {
    "redis_build_id": "606641459177bc09",
    "config_file": "\/etc\/redis\/redis.conf",
    "uptime_in_seconds": "1113787",
    "hz": "10",
    "os": "Linux 2.6.32-220.el6.x86_64 x86_64",
    "redis_git_sha1": "00000000",
    "redis_version": "2.8.15",
    "process_id": "14607",
    "tcp_port": "6379",
    "redis_git_dirty": "0",
    "redis_mode": "standalone",
    "run_id": "43c5d003453b96e38ad3eae54026d8e1b078a7fd",
    "uptime_in_days": "12",
    "gcc_version": "4.4.6",
    "arch_bits": "64",
    "lru_clock": "16050046",
    "multiplexing_api": "epoll"
 },
 "Keyspace": {
    "db0": "keys=26319,expires=0,avg_ttl=0"
 },
 "slowlog": [
    [
       "32656",
       "1425336191",
       "18539",
       [
```

(continues on next page)

(continued from previous page)

```
         "keys",
         "*"
      ]
   ]
],
"Persistence": {
   "rdb_bgsave_in_progress": "0",
   "loading": "0",
   "rdb_current_bgsave_time_sec": "-1",
   "aof_enabled": "0",
   "rdb_last_bgsave_time_sec": "-1",
   "aof_last_rewrite_time_sec": "-1",
   "aof_last_write_status": "ok",
   "rdb_last_bgsave_status": "ok",
   "aof_last_bgrewrite_status": "ok",
   "aof_current_rewrite_time_sec": "-1",
   "aof_rewrite_scheduled": "0",
   "aof_rewrite_in_progress": "0",
   "rdb_last_save_time": "1424222403",
   "rdb_changes_since_last_save": "2595831724"
},
"slowlen": 128,
"CPU": {
   "used_cpu_user": "45252.98",
   "used_cpu_sys": "154718.84",
   "used_cpu_user_children": "0.00",
   "used_cpu_sys_children": "0.00"
},
"Memory": {
   "used_memory_lua": "33792",
   "mem_allocator": "jemalloc-3.6.0",
   "used_memory_human": "5.25G",
   "used_memory_peak_human": "8.08G",
   "used_memory_peak": "8675798632",
   "used_memory_rss": "8870088704",
   "mem_fragmentation_ratio": "1.57",
   "used_memory": "5633381640"
},
"Replication": {
   "repl_backlog_first_byte_offset": "0",
   "repl_backlog_active": "0",
   "repl_backlog_histlen": "0",
   "repl_backlog_size": "1048576",
   "role": "master",
   "master_repl_offset": "0",
   "connected_slaves": "0"
},
"Clients": {
   "client_biggest_input_buf": "0",
   "client_longest_output_list": "0",
   "blocked_clients": "0",
   "connected_clients": "16"
},
"Stats": {
   "latest_fork_usec": "0",
   "rejected_connections": "0",
   "sync_partial_ok": "0",
```

(continues on next page)

(continued from previous page)

```
    "pubsub_channels": "0",
    "instantaneous_ops_per_sec": "2238",
    "total_connections_received": "2502657",
    "evicted_keys": "0",
    "pubsub_patterns": "0",
    "sync_full": "0",
    "keyspace_hits": "49388626",
    "keyspace_misses": "780",
    "total_commands_processed": "2645272238",
    "expired_keys": "0",
    "sync_partial_err": "0"
  }
}
```

GET /api/1.1/redis/match/#match/start_date/:start_date/end_date/:end_date/interval/:interval.json

Authentication Required:

Request Route Parameters

Parameter	Type	Description
start_date	string	
end_date	string	
interval	string	

Request Example

Response Content Type: application/json

Response Messages

```
HTTP Status Code: 200
Reason: Success
```

Response Properties

Parameter	Type	Description
alerts	array	
>level	string	
>text	string	
version	string	

Response Example

Regions

GET /api/1.1/regions.json

Authentication Required:

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`name`	string	
`id`	string	

Response Example

```
{
 "response": [
    {
       "name": "Atlanta",
       "id": "6"
    },
    {
       "name": "Beltway",
       "id": "1"
    }
 ],
 "version": "1.1"
}
```

Roles

GET /api/1.1/roles.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`name`	string	
`id`	string	
`privLevel`	string	
`description`	string	

Response Example

```
{
 "response": [
    {
       "name": "read-only",
       "id": "2",
       "privLevel": "10",
       "description": "read-only user"
    }
 ],
 "version": "1.1"
}
```

Server

GET /api/1.1/servers.json

Retrieves properties of CDN servers.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`cachegroup`	string	The cache group name (see *Cache Group*).
`domainName`	string	The domain name part of the FQDN of the cache.
`hostName`	string	The host name part of the cache.
`id`	string	The server id (database row number).
`iloIpAddress`	string	The IPv4 address of the lights-out-management port.
`iloIpGateway`	string	The IPv4 gateway address of the lights-out-management port.
`iloIpNetmask`	string	The IPv4 netmask of the lights-out-management port.
`iloPassword`	string	The password of the of the lights-out-management user (displays as ** unless you are an 'admin' use
`iloUsername`	string	The user name for lights-out-management.
`interfaceMtu`	string	The Maximum Transmission Unit (MTU) to configure for `interfaceName`.
`interfaceName`	string	The network interface name used for serving traffic.
`ip6Address`	string	The IPv6 address/netmask for `interfaceName`.
`ip6Gateway`	string	The IPv6 gateway for `interfaceName`.
`ipAddress`	string	The IPv4 address for `interfaceName`.
`ipGateway`	string	The IPv4 gateway for `interfaceName`.
`ipNetmask`	string	The IPv4 netmask for `interfaceName`.
`lastUpdated`	string	The Time and Date for the last update for this server.
`mgmtIpAddress`	string	The IPv4 address of the management port (optional).
`mgmtIpGateway`	string	The IPv4 gateway of the management port (optional).
`mgmtIpNetmask`	string	The IPv4 netmask of the management port (optional).
`physLocation`	string	The physical location name (see *Physical Location*).
`profile`	string	The assigned profile name (see *Profiles*).
`rack`	string	A string indicating rack location.
`routerHostName`	string	The human readable name of the router.
`routerPortName`	string	The human readable name of the router port.
`status`	string	The Status string (See *Status*).
`tcpPort`	string	The default TCP port on which the main application listens (80 for a cache in most cases).
`type`	string	The name of the type of this server (see *Types*).
`xmppId`	string	Deprecated.
`xmppPasswd`	string	Deprecated.

Response Example

```
{
   "response": [
       {
           "cachegroup": "us-il-chicago",
           "domainName": "chi.kabletown.net",
           "hostName": "atsec-chi-00",
           "id": "19",
           "iloIpAddress": "172.16.2.6",
           "iloIpGateway": "172.16.2.1",
           "iloIpNetmask": "255.255.255.0",
```

(continues on next page)

(continued from previous page)

```
            "iloPassword": "********",
            "iloUsername": "",
            "interfaceMtu": "9000",
            "interfaceName": "bond0",
            "ip6Address": "2033:D0D0:3300::2:2/64",
            "ip6Gateway": "2033:D0D0:3300::2:1",
            "ipAddress": "10.10.2.2",
            "ipGateway": "10.10.2.1",
            "ipNetmask": "255.255.255.0",
            "lastUpdated": "2015-03-08 15:57:32",
            "mgmtIpAddress": "",
            "mgmtIpGateway": "",
            "mgmtIpNetmask": "",
            "physLocation": "plocation-chi-1",
            "profile": "EDGE1_CDN1_421_SSL",
            "rack": "RR 119.02",
            "routerHostName": "rtr-chi.kabletown.net",
            "routerPortName": "2",
            "status": "ONLINE",
            "tcpPort": "80",
            "type": "EDGE",
            "xmppId": "atsec-chi-00-dummyxmpp",
            "xmppPasswd": "**********"
        },
        {
        ... more server data
        }
      ]
    "version": "1.1"
}
```

GET /api/1.1/servers/summary.json

Retrieves a count of CDN servers by type.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`count`	int	The number of servers of this type in this instance of Traffic Ops.
`type`	string	The name of the type of the server count (see *Types*).

Response Example

```
{
  "response": [
    {
      "count": 4,
      "type": "CCR"
    },
    {
```

(continues on next page)

(continued from previous page)

```
      "count": 55,
      "type": "EDGE"
    },
    {
      "type": "MID",
      "count": 18
    },
    {
      "count": 0,
      "type": "REDIS"
    },
    {
      "count": 4,
      "type": "RASCAL"
    }
  "version": "1.1",
}
```

GET /api/1.1/servers/hostname/:name/details.json

Retrieves the details of a server.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`name`	yes	The host name part of the cache.

Response Properties

Parameter	Type	Description
`cachegroup`	string	The cache group name (see *Cache Group*).
`deliveryservices`	array	Array of strings with the delivery service ids assigned (see *Delivery Service*).
`domainName`	string	The domain name part of the FQDN of the cache.
`hardwareInfo`	hash	Hwinfo struct (see *hwinfo*).
`hostName`	string	The host name part of the cache.
`id`	string	The server id (database row number).
`iloIpAddress`	string	The IPv4 address of the lights-out-management port.
`iloIpGateway`	string	The IPv4 gateway address of the lights-out-management port.
`iloIpNetmask`	string	The IPv4 netmask of the lights-out-management port.
`iloPassword`	string	The password of the of the lights-out-management user (displays as ** unless you are an ‘admin’
`iloUsername`	string	The user name for lights-out-management.
`interfaceMtu`	string	The Maximum Transmission Unit (MTU) to configure for `interfaceName`.
`interfaceName`	string	The network interface name used for serving traffic.
`ip6Address`	string	The IPv6 address/netmask for `interfaceName`.
`ip6Gateway`	string	The IPv6 gateway for `interfaceName`.
`ipAddress`	string	The IPv4 address for `interfaceName`.
`ipGateway`	string	The IPv4 gateway for `interfaceName`.

Continued on next

Table 7 – continued from previous page

Parameter	Type	Description
ipNetmask	string	The IPv4 netmask for interfaceName.
lastUpdated	string	The Time/Date of the last update for this server.
mgmtIpAddress	string	The IPv4 address of the management port (optional).
mgmtIpGateway	string	The IPv4 gateway of the management port (optional).
mgmtIpNetmask	string	The IPv4 netmask of the management port (optional).
physLocation	string	The physical location name (see *Physical Location*).
profile	string	The assigned profile name (see *Profiles*).
rack	string	A string indicating rack location.
routerHostName	string	The human readable name of the router.
routerPortName	string	The human readable name of the router port.
status	string	The Status string (See *Status*).
tcpPort	string	The default TCP port on which the main application listens (80 for a cache in most cases).
type	string	The name of the type of this server (see *Types*).
xmppId	string	Deprecated.
xmppPasswd	string	Deprecated.

Response Example

```
{
  "response": {
    "cachegroup": "us-il-chicago",
    "deliveryservices": [
      "1",
      "2",
      "3",
      "4"
    ],
    "domainName": "chi.kabletown.net",
    "hardwareInfo": {
      "Physical Disk 0:1:3": "D1S2",
      "Physical Disk 0:1:2": "D1S2",
      "Physical Disk 0:1:15": "D1S2",
      "Power Supply.Slot.2": "04.07.15",
      "Physical Disk 0:1:24": "YS08",
      "Physical Disk 0:1:1": "D1S2",
      "Model": "PowerEdge R720xd",
      "Physical Disk 0:1:22": "D1S2",
      "Physical Disk 0:1:18": "D1S2",
      "Enterprise UEFI Diagnostics": "4217A5",
      "Lifecycle Controller": "1.0.8.42",
      "Physical Disk 0:1:8": "D1S2",
      "Manufacturer": "Dell Inc.",
      "Physical Disk 0:1:6": "D1S2",
      "SysMemTotalSize": "196608",
      "PopulatedDIMMSlots": "24",
      "Physical Disk 0:1:20": "D1S2",
      "Intel(R) Ethernet 10G 2P X520 Adapter": "13.5.7",
      "Physical Disk 0:1:14": "D1S2",
      "BACKPLANE FIRMWARE": "1.00",
      "Dell OS Drivers Pack, 7.0.0.29, A00": "7.0.0.29",
      "Integrated Dell Remote Access Controller": "1.57.57",
      "Physical Disk 0:1:5": "D1S2",
      "ServiceTag": "D6XPDV1",
      "PowerState": "2",
```

(continues on next page)

(continued from previous page)

```
        "Physical Disk 0:1:23": "D1S2",
        "Physical Disk 0:1:25": "D903",
        "BIOS": "1.3.6",
        "Physical Disk 0:1:12": "D1S2",
        "System CPLD": "1.0.3",
        "Physical Disk 0:1:4": "D1S2",
        "Physical Disk 0:1:0": "D1S2",
        "Power Supply.Slot.1": "04.07.15",
        "PERC H710P Mini": "21.0.2-0001",
        "PowerCap": "689",
        "Physical Disk 0:1:16": "D1S2",
        "Physical Disk 0:1:10": "D1S2",
        "Physical Disk 0:1:11": "D1S2",
        "Lifecycle Controller 2": "1.0.8.42",
        "BP12G+EXP 0:1": "1.07",
        "Physical Disk 0:1:9": "D1S2",
        "Physical Disk 0:1:17": "D1S2",
        "Broadcom Gigabit Ethernet BCM5720": "7.2.20",
        "Physical Disk 0:1:21": "D1S2",
        "Physical Disk 0:1:13": "D1S2",
        "Physical Disk 0:1:7": "D1S2",
        "Physical Disk 0:1:19": "D1S2"
      },
      "hostName": "atsec-chi-00",
      "id": "19",
      "iloIpAddress": "172.16.2.6",
      "iloIpGateway": "172.16.2.1",
      "iloIpNetmask": "255.255.255.0",
      "iloPassword": "********",
      "iloUsername": "",
      "interfaceMtu": "9000",
      "interfaceName": "bond0",
      "ip6Address": "2033:D0D0:3300::2:2/64",
      "ip6Gateway": "2033:D0D0:3300::2:1",
      "ipAddress": "10.10.2.2",
      "ipGateway": "10.10.2.1",
      "ipNetmask": "255.255.255.0",
      "mgmtIpAddress": "",
      "mgmtIpGateway": "",
      "mgmtIpNetmask": "",
      "physLocation": "plocation-chi-1",
      "profile": "EDGE1_CDN1_421_SSL",
      "rack": "RR 119.02",
      "routerHostName": "rtr-chi.kabletown.net",
      "routerPortName": "2",
      "status": "ONLINE",
      "tcpPort": "80",
      "type": "EDGE",
      "xmppId": "atsec-chi-00-dummyxmpp",
      "xmppPasswd": "X"

    }
    "version": "1.1",

  }
```

POST /api/1.1/servercheck

Post a server check result to the serverchecks table.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`id`	yes	
`host_name`	yes	
`servercheck_short_name`	yes	
`value`	yes	

Request Example

```
{
 "id": "",
 "host_name": "",
 "servercheck_short_name": "",
 "value": ""
}
```

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.
`version`	string	

Response Example

```
Response Example:

{
  "alerts":
    [
      {
        "level": "success",
        "text": "Server Check was successfully updated."
      }
    ],
  "version": "1.1"
}
```

Static DNS Entries

GET /api/1.1/staticdnsentries.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
TBD	array	

Response Example

```
TBD
```

Status

GET /api/1.1/statuses.json

Retrieves a list of the server status codes available. May be useful when the status is retrieved from other APIs as a number and not a string.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`lastUpdated`	string	The Time / Date this server entry was last updated
`name`	string	The string equivalent of the status
`id`	string	The id with which Traffic Ops stores this status, and references it internally
`description`	string	A short description of the status

Response Example

```
{
 "response": [
   {
     "description": "Temporary down. Edge: XMPP client will send status␣
↪OFFLINE to CCR, otherwise similar to REPORTED. Mid: Server will not be␣
↪included in parent.config files for its edge caches",
     "id": "4",
     "name": "ADMIN_DOWN",
     "lastUpdated": "2013-02-13 16:34:29"
   },
   {
     "lastUpdated": "2013-02-13 16:34:29",
     "name": "CCR_IGNORE",
     "id": "5",
     "description": "Edge: 12M will not include caches in this state in CCR␣
↪config files. Mid: N\/A for now"
   },
   {
     "description": "Edge: Puts server in CCR config file in this state,␣
↪but CCR will never route traffic to it. Mid: Server will not be included␣
↪in parent.config files for its edge caches",
     "id": "1",
     "lastUpdated": "2013-02-13 16:34:29",
     "name": "OFFLINE"
   },
   {
     "id": "2",
```

(continues on next page)

(continued from previous page)

```
      "description": "Edge: Puts server in CCR config file in this state, and CCR will always route traffic to it. Mid: Server will be included in parent.config files for its edges",
      "lastUpdated": "2013-02-13 16:34:29",
      "name": "ONLINE"
    },
    {
      "id": "3",
      "description": "Edge: Puts server in CCR config file in this state, and CCR will adhere to the health protocol. Mid: N\/A for now",
      "name": "REPORTED",
      "lastUpdated": "2013-02-13 16:34:29"
    }
  ],
  "version": "1.1"
}
```

System

GET /api/1.1/system/info.json

Authentication Required: Yes

Response Properties

Key	Type	Description
`parameters`	hash	This is a hash with the parameter names that describe the Traffic Ops installation as keys. These are all the parameters in the `GLOBAL` profile.
`>tm.toolname`	string	The name of the Traffic Ops tool. Usually "Traffic Ops". Used in the About screen and in the comments headers of the files generated (`# DO NOT EDIT - Generated for atsec-lax-04 by Traffic Ops (https://traffops.kabletown.net/) on Fri Mar 6 05:15:15 UTC 2015`).
`>tm.instance_name`	string	The name of the Traffic Ops instance. Can be used when multiple instances are active. Visible in the About page.
`>traffic_rtr_fwd_proxy`	string	When collecting stats from Traffic Router, Traffic Ops uses this forward proxy to pull the stats through. This can be any of the MID tier caches, or a forward cache specifically deployed for this purpose. Setting this variable can significantly lighten the load on the Traffic Router stats system and it is recommended to set this parameter on a production system.
`>tm.url`	string	The URL for this Traffic Ops instance. Used in the About screen and in the comments headers of the files generated (`# DO NOT EDIT - Generated for atsec-lax-04 by Traffic Ops (https://traffops.kabletown.net/) on Fri Mar 6 05:15:15 UTC 2015`).
`>traffic_mon_fwd_proxy`	string	When collecting stats from Traffic Monitor, Traffic Ops uses this forward proxy to pull the stats through. This can be any of the MID tier caches, or a forward cache specifically deployed for this purpose. Setting this variable can significantly lighten the load on the Traffic Monitor system and it is recommended to set this parameter on a production system.
`>tm.logourl`	string	This is the URL of the logo for Traffic Ops and can be relative if the logo is under traffic_ops/app/public.
`>tm.infourl`	string	This is the "for more information go here" URL, which is visible in the About page.

Response Example

```
{
  "response": {
    "parameters": {
      "tm.toolname": "Traffic Ops",
      "tm.infourl": "http:\/\/staging-03.cdnlab.kabletown.net\/tm\/info",
      "traffic_mon_fwd_proxy": "http:\/\/proxy.kabletown.net:81",
      "traffic_rtr_fwd_proxy": "http:\/\/proxy.kabletown.net:81",
      "tm.logourl": "\/images\/tc_logo.png",
      "tm.url": "https:\/\/tm.kabletown.net\/",
      "tm.instance_name": "Kabletown CDN"
    }
  },
  "version": "1.1"
}
```

TO Extensions

GET /api/1.1/to_extensions.json

Retrieves the list of extensions.

Authentication Required: Yes

Response Content Type: application/json

Return Values

Parameter	Type	Description
`script_file`	string	
`version`	string	
`name`	string	
`description`	string	
`info_url`	string	
`additional_config_json`	string	
`isactive`	string	
`id`	string	
`type`	string	
`servercheck_short_name`	string	

Response Example

```
{
        “response”: [
                {
                        script_file: "ping",
                        version: "1.0.0",
                        name: "ILO_PING",
                        description: null,
                        info_url: "http://foo.com/bar.html",
                        additional_config_json: "{ "path": "/api/1.1/servers.json",
↪"match": { "type": "EDGE"}, "select": "ilo_ip_address", "cron": "9 * * * *" }",
                        isactive: "1",
                        id: "1",
```

(continues on next page)

(continued from previous page)

```
                        type: "CHECK_EXTENSION_BOOL",
                        servercheck_short_name: "ILO"
                },
                {
                        script_file: "ping",
                        version: "1.0.0",
                        name: "10G_PING",
                        description: null,
                        info_url: "http://foo.com/bar.html",
                        additional_config_json: "{ "path": "/api/1.1/servers.json",
↪"match": { "type": "EDGE"}, "select": "ip_address", "cron": "18 * * * *" }",
                        isactive: "1",
                        id: "2",
                        type: "CHECK_EXTENSION_BOOL",
                        servercheck_short_name: "10G"
                }
        ],
        "version": "1.1"
}
```

POST /api/1.1/to_extensions

Creates a Traffic Ops extension.

Authentication Required: Yes

Request Parameters

Parameter	Type	Description
`name`	string	
`version`	string	
`info_url`	string	
`script_file`	string	
`isactive`	string	
`additional_config_json`	string	
`description`	string	
`servercheck_short_name`	string	
`type`	string	

Request Example

```
{
      "name": "ILO_PING",
      "version": "1.0.0",
      "info_url": "http://foo.com/bar.html",
      "script_file": "ping",
      "isactive": "1",
      "additional_config_json": "{ "path": "/api/1.1/servers.json", "match":
↪{ "type": "EDGE"}",
      "description": null,
      "servercheck_short_name": "ILO"
```

(continues on next page)

(continued from previous page)

```
      "type": "CHECK_EXTENSION_BOOL",
}
```

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.

Response Example

```
{
 "alerts": [
    {
       "level": "success",
       "text": "Check Extension loaded."
    }
 ],
 "version": "1.1"
}
```

POST /api/1.1/to_extensions/:id/delete

Deletes a Traffic Ops extension.

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
`id`	yes	TO extension id

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.

Response Example

```
 {
"alerts": [
   {
      "level": "success",
```

(continues on next page)

(continued from previous page)

```
      "text": "Extension deleted."
    }
],
"version": "1.1"
 }
```

Types

GET /api/1.1/types.json

Authentication Required: Yes

Response Properties

Parameter	Type	Description
lastUpdated	string	
useInTable	string	
name	string	
id	string	
description	string	

Response Example

```
{
 "response": [
    {
       "lastUpdated": "2013-10-23 15:28:31",
       "useInTable": "staticdnsentry",
       "name": "AAAA_RECORD",
       "id": "22",
       "description": "Static DNS AAAA entry"
    }
 ],
 "version": "1.1"
}
```

GET /api/1.1/types/trimmed.json

Authentication Required: Yes

Response Content Type: application/json

Response Properties

Parameter	Type	Description
name	string	

Response Example

```
{
 "response": [
    {
       "name": "AAAA_RECORD"
    },
    {
       "name": "ACTIVE_DIRECTORY"
    },
    {
       "name": "A_RECORD"
    },
    {
       "name": "CCR"
    }
 ],
 "version": "1.1"
}
```

Users

GET /api/1.1/users.json

Retrieves all users.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
email	string	
city	string	
id	hash	
phoneNumber	string	
company	string	
country	string	
fullName	string	
localUser	string	
uid	string	
username	string	
rolename	string	
newUser	string	
addressLine2	string	
role	string	
addressLine1	string	
postalCode	string	
gid	string	

Response Example

```
[
 {
    "email": "email@email.com",
    "city": "",
    "id": "54",
```

(continues on next page)

(continued from previous page)

```
    "phoneNumber": "",
    "company": "",
    "country": "",
    "fullName": "Bob Simpson",
    "localUser": false,
    "uid": "0",
    "stateOrProvince": "",
    "username": "bsimpson",
    "rolename": "portal",
    "newUser": true,
    "addressLine2": "",
    "role": "6",
    "addressLine1": "",
    "postalCode": "",
    "gid": "0"
 }
]
```

GET /api/1.1/user/current.json

Retrieves the profile for the authenticated user.

Authentication Required: Yes

Request Properties

Parameter	Type	Description
`email`	string	
`city`	string	
`id`	string	
`phoneNumber`	string	
`company`	string	
`country`	string	
`fullName`	string	
`localUser`	boolean	
`uid`	string	
`stateOrProvince`	string	
`username`	string	
`newUser`	boolean	
`addressLine2`	string	
`role`	string	
`addressLine1`	string	
`gid`	string	
`postalCode`	string	

Response Example

```
{
       “response”: {
                        “email”: "email@email.com",
```

(continues on next page)

(continued from previous page)

```
                        “city”: "",
                        “id”: "50",
                        “phoneNumber”: "",
                        “company”: "",
                        “country”: "",
                        “fullName”: "Tom Callahan",
                        “localUser”: true,
                        “uid”: "0",
                        “stateOrProvince”: "",
                        “username”: "tommyboy",
                        “newUser”: false,
                        “addressLine2”: "",
                        “role”: "6",
                        “addressLine1”: "",
                        “gid”: "0",
                        “postalCode”: ""
        },
        “version”: "1.1"
}
```

POST /api/1.1/user/current/update

Updates the date for the authenticated user.

Authentication Required: Yes

Request Properties

Parameter	Type	Description
`email`	string	
`city`	string	
`id`	string	
`phoneNumber`	string	
`company`	string	
`country`	string	
`fullName`	string	
`localUser`	boolean	
`uid`	string	
`stateOrProvince`	string	
`username`	string	
`newUser`	boolean	
`addressLine2`	string	
`role`	string	
`addressLine1`	string	
`gid`	string	
`postalCode`	string	

Request Example

```
{
 "user": {
    "email": "",
    "city": "",
    "id": "",
    "phoneNumber": "",
    "company": "",
    "country": "",
    "fullName": "",
    "localUser": true,
    "uid": "0",
    "stateOrProvince": "",
    "username": "tommyboy",
    "newUser": false,
    "addressLine2": "",
    "role": "6",
    "addressLine1": "",
    "gid": "0",
    "postalCode": ""
 }
}
```

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.
`version`	string	

Response Example

```
{
      "alerts": [
                {
                        "level": "success",
                        "text": "UserProfile was successfully updated."
                }
        ],
        "version": "1.1"
}
```

GET /api/1.1/user/current/jobs.json

Retrieves user purge jobs.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`keyword`	string	
`objectName`	string	
`assetUrl`	string	
`assetType`	string	
`status`	string	
`dsId`	string	
`dsXmlId`	string	
`username`	boolean	
`parameters`	string	
`enteredTime`	string	
`objectType`	string	
`agent`	string	
`id`	string	
`startTime`	string	
`version`	string	

Response Example

```
{
 "response": [
    {
        "id": "1",
        "keyword": "PURGE",
        "objectName": null,
        "assetUrl": "",
        "assetType": "file",
        "status": "PENDING",
        "dsId": "73",
        "dsXmlId": "cim-jitp",
        "username": "peewee",
        "parameters": "TTL:56h",
        "enteredTime": "2015-01-21 18:00:16",
        "objectType": null,
        "agent": "",
        "startTime": "2015-01-21 10:45:38"
    }
 ],
 "version": "1.1"
}
```

POST/api/1.1/user/current/jobs

Creates a purge job.

Authentication Required: Yes

Request Properties

Parameter	Type	Description
`dsId`	string	
`dsXmlId`	string	
`regex`	string	
`startTime`	string	
`ttl`	int	

Request Example

```
{
        "dsId": "73",
        "dsXmlId": "cim-jitp",
        "regex": "/path/to/content.jpg",
        "startTime": "2015-01-27 11:08:37",
        "ttl": 54
}
```

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.
`version`	string	

Response Example

```
{
      “alerts”:
              [
                  {
                        “level”: "success",
                        “text”: "Successfully created purge job for: ."
                  }
              ],
      “version”: "1.1"
}
```

POST /api/1.1/user/login { u: ‘’, p: ‘’ }

Authentication of a user using username and password. Traffic Ops will send back a session cookie.

Authentication Required: No

Request Properties

Parameter	Type	Description
`u`	string	username
`p`	string	password

Request Example

```
 {
  "u": "username",
  "p": "password"
}
```

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
`>level`	string	Success, info, warning or error.
`>text`	string	Alert message.
`version`	string	

Response Example

```
{
  "alerts": [
      {
         "level": "success",
         "text": "Successfully logged in."
      }
  ],
  "version": "1.1"
 }
```

GET /api/1.1/user/:id/deliveryservices/available.json

Authentication Required: Yes

Request Route Parameters

Name	Required	Description
id	yes	

Response Properties

Parameter	Type	Description
`xmlId`	string	
`id`	string	

Response Example

```
{
 "response": [
    {
       "xmlId": "ns-img",
```

(continues on next page)

(continued from previous page)

```
            "id": "90"
        },
        {
            "xmlId": "ns-img-secure",
            "id": "280"
        }
    ],
    "version": "1.1"
}
```

POST /api/1.1/user/login/token

Authentication of a user using a token.

Authentication Required: No

Request Route Properties

Parameter	Type	Description
`t`	string	token-value

Request Example

```
{
    "t": "token-value"
}
```

Response Content Type: application/json

Response Properties

Parameter	Type	Description
`alerts`	array	
`>level`	string	
`>text`	string	
`version`	string	

Response Example

```
{
    "alerts": [
        {
            "level": "error",
            "text": "Unauthorized, please log in."
        }
    ],
    "version": "1.1"
}
```

POST /api/1.1/user/logout

User logout. Invalidates the session cookie.

Authentication Required: Yes

Response Properties

Parameter	Type	Description
`alerts`	array	
• `level`	string	
• `text`	string	
`version`	string	

Response Example

```
{
 "alerts": [
    {
       "level": "success",
       "text": "You are logged out."
    }
 ],
 "version": "1.1"
}
```

POST /api/1.1/user/reset_password

Reset user password.

Authentication Required: No

Request Properties

Parameter	Type	Description
`email`	string	The email address of the user to initiate password reset.

Request Example

```
{
 "email": "email@email.com"
}
```

Response Properties

Parameter	Type	Description
`alerts`	array	A collection of alert messages.
• `level`	string	Success, info, warning or error.
• `text`	string	Alert message.
`version`	string	

Response Example

```
{
 "alerts": [
    {
       "level": "success",
       "text": "Successfully logged in."
    }
 ],
 "version": "1.1"
}
```

4.1.2 Traffic Router

Introduction

Traffic Router is a Java Tomcat application that routes clients to the closest available cache on the CDN using both HTTP and DNS. Cache availability is determined by Traffic Monitor; consequently Traffic Router polls Traffic Monitor for its configuration and cache health state information, and uses this data to make routing decisions. HTTP routing is performed by localizing the client based on the request's source IP address (IPv4 or IPv6), and issues an HTTP 302 redirect to the nearest cache. HTTP routing utilizes consistent hashing on request URLs to optimize cache performance and request distribution. DNS routing is performed by localizing clients, resolvers in most cases, requesting `A` and `AAAA` records for a configurable name such as `edge.deliveryservice.somecdn.net`. Traffic Router is comprised of four separate Maven modules:

- api - Provides a simple JSON interface into certain aspects of core and is deployed as a WAR to a Service (read: connector/listen port) within Tomcat which is separate from core
- connector - A JAR that overrides Tomcat's standard Http11Protocol Connector class and allows Traffic Router to delay opening listen sockets until it is in a state suitable for routing traffic
- core - Services DNS and HTTP requests, performs localization on routing requests, and is deployed as a WAR to a Service (read: connector/listen port) within Tomcat which is separate from api
- rpm - A simple Maven project which gathers the artifacts from the prior three modules and builds an RPM

Software Requirements

To work on Traffic Router you need a *nix (MacOS and Linux are most commonly used) environment that has the following installed:

- Eclipse >= Kepler SR2 (or another Java IDE)
- Maven >= 3.3.1
- JDK >= 6.0

Traffic Router Project Tree Overview

- `traffic_control/traffic_traffic_router/` - base directory for Traffic Router
  - `api/` - Source code for Traffic Router API, which is built as its own deployable WAR file and communicates with Traffic Router Core using JMX
    - `src/main` - Main source directory for Traffic Router API
      - `java/` - Java source code for Traffic Router API
      - `resources/` - Spring resources pulled in during an RPM build
      - `webapp/` - Java webapp resources
    - `src/test` - Test source directory for Traffic Router API
      - `java/` - JUnit based unit tests for Traffic Router API
      - `resources/` - Resources pulled in by unit tests
  - `connector/` - Source code for Traffic Router Connector;
    - `src/main/java` - Java source directory for Traffic Router Connector
  - `core/` - Source code for Traffic Router Core, which is built as its own deployable WAR file and communicates with Traffic Router API using JMX
    - `src/main` - Main source directory for Traffic Router Core
      - `etc/init.d` - Init script for Tomcat
      - `conf/` - Configuration files
      - `java/` - Java source code for Traffic Router Core
      - `opt/tomcat/conf` - Contains Tomcat configuration file(s) pulled in during an RPM build
      - `resources/` - Resources pulled in during an RPM build
      - `scripts/` - Scripts used by the RPM build process
      - `webapp/` - Java webapp resources
    - `src/test` - Test source directory for Traffic Router Core
      - `db` - Files downloaded by unit tests
      - `java/` - JUnit based unit tests for Traffic Router Core
      - `resources/` - Configuration files used by unit tests
      - `var/auto-zones` - BIND formatted zone files generated by Traffic Router Core during unit testing

Java Formatting Conventions

None at this time. The codebase will eventually be formatted per Java standards.

Installing The Developer Environment

To install the Traffic Router Developer environment:

1. Clone the traffic_control repository using Git.

2. Change directories into `traffic_control/traffic_router`.
3. If you are not running Traffic Monitor locally (http://localhost:8080) from within Eclipse, edit the following parameter in core/src/test/resources/traffic_monitor.properties and point it to an instance, or instances of Traffic Monitor for your chosen CDN:

Parameter	Value
`traffic_monitor.bootstrap.hosts`	FQDN and port of the Traffic Monitor instance(s), separated by semicolons as necessary (do not include http://).

4. Import the existing git repo into Eclipse:
   (a) File -> Import -> Git -> Projects from Git; Next
   (b) Existing local repository; Next
   (c) Add -> browse to find `traffic_control`; Open
   (d) Select `traffic_control`; Next
   (e) Ensure "Import existing projects" is selected, expand `traffic_control`, select `traffic_router`; Next
   (f) Ensure `traffic_router_api`, `traffic_router_connector`, and `traffic_router_core` are checked; Finish (this step can take several minutes to complete)
   (g) Ensure `traffic_router_api`, `traffic_router_connector`, and `traffic_router_core` have been opened by Eclipse after importing
5. From the terminal, run `mvn clean verify` from the `traffic_router` directory
6. Start the embedded Jetty instance for Core from within Eclipse
   (a) In the package explorer, expand `traffic_router_core`
   (b) Expand `src/test/java`
   (c) Expand the package `com.comcast.cdn.traffic_control.traffic_router.core`
   (d) Open and run `TrafficRouterStart.java`

   Note: If an error is displayed in the Console, run `mvn clean verify` from the `traffic_router` directory

7. Traffic Router Core should now be running; the HTTP routing interface is available on http://localhost:8081, while the DNS server and routing interface is available on localhost:1053 via TCP and UDP.

Test Cases

Unit tests can be executed using Maven by running `mvn test` at the root of the `traffic_router` project.

API

Traffic Router API

Traffic Router API

/crs/stats

General stats.

/crs/stats/ip/:ipaddress

Geolocation information for an IPv4 or IPv6 address.

/crs/locations

A list of configured cache groups.

/crs/locations/caches

A mapping of caches to cache groups and their current health state.

/crs/locations/:location/caches

A list of caches for this cache group only.

4.1.3 Traffic Monitor

Introduction

Traffic Monitor is a Java Tomcat application that monitors caches, provides health state information to Traffic Router, and collects statistics for use in tools such as Traffic Ops and Traffic Stats. The health state provided by Traffic Monitor is used by Traffic Router to control which caches are available on the CDN.

Software Requirements

To work on Traffic Monitor you need a *nix (MacOS and Linux are most commonly used) environment that has the following installed:

- Eclipse >= Kepler SR2 (or another Java IDE)
- Maven >= 3.3.1
- JDK >= 6.0

Traffic Monitor Project Tree Overview

- `traffic_control/traffic_monitor/` - base directory for Traffic Monitor
  - `etc/` - Miscellaneous simulator utilities
  - `src/main` - Main source directory for the Traffic Monitor
    - `bin/` - Configuration tools
    - `conf/` - Configuration files
    - `java/` - Java source code for Traffic Monitor
    - `opt/tomcat/conf` - Contains Tomcat configuration file(s) pulled in during an RPM build
    - `resources/` - Resources pulled in during an RPM build
    - `scripts/` - Scripts used by the RPM build process
    - `webapp/` - Java webapp resources
  - `src/test` - Test source directory for Traffic Monitor
    - `java/` - JUnit based unit tests for Traffic Monitor
    - `resources/conf` - Configuration files used by unit tests
    - `resources/db` - Files downloaded by unit tests
    - `resources/var` - Files generated by unit tests

Java Formatting Conventions

None at this time. The codebase will eventually be formatted per Java standards.

Installing The Developer Environment

To install the Traffic Monitor Developer environment:

1. Clone the traffic_control repository using Git.
2. Change directories into `traffic_control/traffic_monitor`.
3. Edit the following parameters in src/test/resources/conf/traffic_monitor_config.js:

Parameter	Value
`tm.hostname`	FQDN of the Traffic Ops instance (do not include http://).
`tm.username`	Admin username for Traffic Ops
`tm.password`	Password for admin user
`cdnName`	Name of the CDN this Traffic Monitor will monitor

4. Import the existing git repo into Eclipse:
   (a) File -> Import -> Git -> Projects from Git; Next
   (b) Existing local repository; Next
   (c) Add -> browse to find `traffic_control`; Add
   (d) Select `traffic_control`; Next

(e) Ensure "Import existing projects" is selected, expand `traffic_control`, select `traffic_monitor`; Next

(f) Ensure `traffic_monitor` is checked; Finish

(g) Ensure `traffic_monitor` has been opened by Eclipse after importing

5. Run `mvn clean verify` from the `traffic_monitor` directory
6. Start the embedded Jetty instance from within Eclipse

(a) In the package explorer, expand `traffic_monitor`

(b) Expand `src/test/java`

(c) Expand the package `com.comcast.cdn.traffic_control.traffic_monitor`

(d) Open and run `Start.java`

> **Note:** If an error is displayed in the Console, run `mvn clean verify` from the `traffic_monitor` directory

(e) With a web browser, navigate to http://localhost:8080

Test Cases

Unit tests can be executed using Maven by running `mvn test` at the root of the `traffic_monitor` project.

API

Traffic Monitor APIs

Traffic Monitor APIs

The Traffic Monitor URLs below allow certain query parameters for use in controlling the data returned. The optional query parameters are the *tabbed* in values under each URL, if they exist.

/publish/EventLog

Log of recent events.

/publish/CacheStats

Statistics gathered for each cache.

Query Parameters

Parameter	Type	Description
`hc`	int	The history count, number of items to display.
`stats`	string	A comma separated list of stats to display.
`wildcard`	boolean	Controls whether specified stats should be treated as partial strings.

/publish/CacheStats/:cache

Statistics gathered for only this cache.

Query Parameters

Parameter	Type	Description
`hc`	int	The history count, number of items to display.
`stats`	string	A comma separated list of stats to display.
`wildcard`	boolean	Controls whether specified stats should be treated as partial strings.

/publish/DsStats

Statistics gathered for delivery services.

Query Parameters

Parameter	Type	Description
`hc`	int	The history count, number of items to display.
`stats`	string	A comma separated list of stats to display.
`wildcard`	boolean	Controls whether specified stats should be treated as partial strings.

/publish/DsStats/:deliveryService

Statistics gathered for this delivery service only.

Query Parameters

Parameter	Type	Description
`hc`	int	The history count, number of items to display.
`stats`	string	A comma separated list of stats to display.
`wildcard`	boolean	Controls whether specified stats should be treated as partial strings.

/publish/CrStates

The current state of this CDN per the health protocol.

raw

The current state of this CDN per this Traffic Monitor only.

/publish/CrConfig

The CrConfig served to and consumed by Traffic Router.

/publish/PeerStates

The health state information from all peer Traffic Monitors.

Query Parameters

Parameter	Type	Description
`hc`	int	The history count, number of items to display.
`stats`	string	A comma separated list of stats to display.
`wildcard`	boolean	Controls whether specified stats should be treated as partial strings.

/publish/Stats

The general statistics about Traffic Monitor.

/publish/StatSummary

The summary of cache statistics.

Query Parameters

Parameter	Type	Description
`startTime`	number	Window start. The number of milliseconds since the epoch.
`endTime`	number	Window end. The number of milliseconds since the epoch.
`hc`	int	The history count, number of items to display.
`stats`	string	A comma separated list of stats to display.
`wildcard`	boolean	Controls whether specified stats should be treated as partial strings.
`cache`	string	Summary statistics for just this cache.

/publish/ConfigDoc

The overview of configuration options.

4.1.4 Traffic Stats

Introduction

Traffic Stats is a . . .

Software Requirements

To work on Traffic Stats you need a *nix (MacOS and Linux are most commonly used) environment that has the following installed:

- ?

Traffic Stats Project Tree Overview

Go Formatting Conventions

Installing The Developer Environment

To install the Traffic Ops Developer environment:

1. Clone the traffic_control repository using Git.

Test Cases

The test harness . . .

4.1.5 Traffic Server

See the Apache Traffic Server documentation.

CHAPTER 5

FAQ

5.1 FAQ

Table of Contents:

5.1.1 General

Who is using Traffic Control?

Comcast Cable Comcast is the original developer or Traffic Control and is using it for all it's video delivery to so-called 'second screen applications' (PCs, tablets, phones), but also for delivering images and software to it's X1 platform, and other applications. The Traffic Control CDN at Comcast serves more than a peta byte of content a day.

Cox Communications .

What is Rascal?

Rascal was the original name for Traffic Monitor. You will sometimes still see this name in the source, or in older documents.

What is the CCR?

Comcast Content Router was the original name for Traffic Router. You will sometimes still see this name in the source, or in older documents.

What is Twelve Monkeys?

Twelve Monkeys was the the original name for Traffic Ops. You will sometimes still see this name in the source, or in older documents. It's also a good movie.

5.1.2 Development

How can I become involved?

5.1.3 Running a Traffic Control CDN

Why is my CRConfig.json rejected?

Especially in version 1.1.0, there's a number of manual steps that need to be done after the initial install. Make sure that after the initial install, you perform these steps in order:

Note: Even though Traffic Ops allows you to enter the servers with no IPv6 address information, the CRConfig will not be accepted by Traffic Router without IPv6 address information for at least Traffic Router and Traffic Monitor. Traffic Control assumes in a lot of places that all servers have at least an IPv4 and an IPv6 address. If you are not using IPv6, it is best to enter dummy addresses for all server types, and turn IPv6 off in all delivery services. (https://github.com/Comcast/traffic_control/issues/44).

- **Add users** Not necessarily needed for getting your CRConfig accepted, but always a good idea.
- **Add Divisions** You will need at least one.
- **Add Regions** You will need at least one.
- **Add Physical Locations** You will need at least one.
- **Add Mid tier Cache Groups** You will need at least one.
- **Add Edge tier Cache Groups**

  You will need at least one. After creating the edge cache group, go to **Parameters > All Profiles**, type CDN_Name in the search box, and click "Edit" for any row of this parameter. Then in the Parameter detail view, click the **Add Cachegroup** button, select your newly created cachegroup, and click **Save**.

  add CDN_Name parameter
- **Add Traffic Monitors** You will need to enter at least one Traffic Monitor - make sure to change the server status to *ONLINE*.
- **Add Traffic Routers** You will need to enter at least one Traffic Router - make sure to change the server status to *ONLINE*.
- **Add Edges** You will need at least one edge cache to make Traffic Router accept the CRConfig.
- **Add Mid** Technically you don't need a mid tier, but if you have one, best to enter the info before continuing.
- **Change the `polling.url` parameters to reflect your CDN** Set where to get the coverage zone map, and the geo IP database.
- **Create at least one delivery service, and assign at least one edge cache in REPORTED state to it.** Even if it is a dummy DS, without a single DS, the CRConfig will not be accepted by Traffic Router.
- **Snapshot CRConfig** **Tools > Snapshot CRConfig** diff, and write.

Now you are ready to install the sw on Traffic Monitor and then Traffic Router.

CHAPTER 6

Indices and Tables

6.1 Glossary

302 content routing *HTTP Content Routing*.

astats (stats_over_http) An ATS plugin that allows you to monitor vitals of the ATS server. See *Cache Monitoring*.

cache A caching proxy server. See *Caching Proxies*.

cachegroup A group of caches that together create a combined larger cache using consistent hashing. See *Cache Group*.

consistent hashing See the Wikipedia article; Traffic Control uses consistent hashing when using *HTTP Content Routing* for the edge tier and when selecting parents in the mid tier.

content routing Directing clients (or client systems) to a particular location or device in a location for optimal delivery of content See also *HTTP Content Routing* and *DNS Content Routing*.

coverage zone map The coverage zone map (czm) or coverage zone file (zcf) is a file that maps network prefixes to cachegroups. See *Localization*.

delivery service A grouping of content in the CDN, usually a determined by the URL hostname. See *Delivery Service*.

edge (tier or cache) Closest to the client or end-user. The edge tier is the tier that serves the client, edge caches are caches in the edge tier. In a Traffic Control CDN the basic function of the edge cache is that of a *Reverse Proxy*. See also *Cache Group*.

(traffic ops) extension Using *extensions*, Traffic Ops be extended to use proprietary checks or monitoring sources. See *Traffic Ops Extension*.

forward proxy A proxy that works that acts like it is the client to the origin. See *Forward Proxy*.

geo localization or geo routing Localizing clients to the nearest caches using a geo database like the one from Maxmind.

health protocol The protocol to monitor the health of all the caches. See *Health Protocol*.

localization Finding location on the network, or on planet earth. See *Localization*.

mid (tier or cache) The tier above the edge tier. The mid tier does not directly serves the end-user and is used as an additional layer between the edge and the origin. In a Traffic Control CDN the basic function of the mid cache is that of a *Forward Proxy*. See also *Cache Group*.

origin The source of content for the CDN. Usually a redundant HTTP/1.1 webserver.

parent (cache or cachegroup) The (group of) cache(s) in the higher tier. See *Cache Group*.

profile A group of settings (parameters) that will be applied to a server. See *Profile*.

reverse proxy A proxy that acts like it is the origin to the client. See *Reverse Proxy*.

Index