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). CDN (Content Delivery Network)s are the technology that make that one-to-many distribution efficient. A CDN is a distributed system of servers for delivering content over HTTP(S). 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:

**Cache Servers** The cache server is a server that both proxies the requests and caches the results for reuse. Traffic Control uses Apache Traffic Server to provide cache servers.

**Content Router** A content router ensures that the end user is connected to the optimal cache server for the location of the end user and content availability. Traffic Control uses Traffic Router as a content router.

**Health Protocol** The Health Protocol monitors the usage of the cache servers and tenants in the CDN.
**Configuration Management System**  In many cases a CDN encompasses hundreds or even thousands of servers across a large geographic area. In such cases, manual configuration of servers becomes impractical, and so a central authority on configuration is used to automate the tasks as much as possible. *Traffic Ops* is the Traffic Control configuration management system, which is interacted with via *Traffic Portal*.

**Log File Analysis System**  Statistics and analysis are extremely important to the management and administration of a CDN. Transaction logs and usage statistics for a Traffic Control CDN are gathered into *Traffic Stats*.

### 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 *cache servers* function.

**See also:**

For complete details on HTTP/1.1 see [RFC 2616](https://www.rfc-editor.org/rfc/rfc2616).

What follows is a sequence of events that take place when a client requests content from an HTTP/1.1-compliant server.

1. The client sends a request to the LDNS (Local DNS) server to resolve the name `www.origin.com` to an IP address, then sends an HTTP request to that IP.

   **Note:** A DNS response is accompanied by a TTL (Time To Live) which indicates for how long a name resolution should be considered valid. 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.

   **# 1: A Client Request for */foo/bar/fun.html* from `www.origin.com`

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

2. The server at `www.origin.com` looks up the content of the path `*/foo/bar/fun.html` and sends it in a response to the client.

   **# 2: Server Response**

   ```
   HTTP/1.1 200 OK
   Content-Type: text/html; charset=UTF-8
   Content-Length: 45
   
   <!DOCTYPE html><html><body>This is a fun file</body></html>
   ```
1.1.3 Cache Control Headers and Revalidation

The HTTP/1.1 specification in RFC 2616#section-14.9 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.

**Note:** The terms “content revalidation” and “content invalidation” are often convoluted when referring to the same behavior. Within the context of Traffic Control, the two should be considered synonymous.

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

The `max-age` directive in the `Cache-Control` header tells downstream caching systems that the maximum time for which they are allowed to cache this response is the specified number of seconds. The origin can also add an `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 particular, this is the way ATS (Apache Traffic Server) handles content revalidation. Instead 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 have sent an `ETag` header that uniquely identifies the response. The client can then later use that in a revalidation request to check if the `ETag` of the requested content has changed.

```
# 4: The Cache Server Sends a Request with the Old ETag Value in the If-None-Match Header
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) the server MUST respond with the up-to-date content, usually in the body of a 200 OK response.

# 5: The Origin Responds with the Modified Content and a New ETag

```
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

<!DOCTYPE html><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) the server SHOULD respond with a 304 Not Modified. In most cases, the server will also send back an ETag header, since the client is allowed to send multiple ETag values in its If-None-Match header to check against multiple cached versions of the content, and the ETag will tell it which specifically is the current version. This is a very rare use case, and ATS will not make use of this feature without a plugin to modify its behavior.

# 6: The Content has not been Modified so the Server Indicates the Cached Version is Up-To-Date

```
HTTP/1.1 304 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
Connection: close
```
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 CDN control plane. It is made up of a suite of applications which are used to configure, manage, and direct client traffic to a tiered system of HTTP caching proxy servers (herein referred to as cache servers). In principle, a CDN may be implemented with any HTTP caching proxy. The caching software chosen for Traffic Control is Apache Traffic Server. Although the current release supports only ATS as a cache server implementation, 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 moved into the Apache Incubator in August of 2016.

Traffic Control implements the elements illustrated in green in the diagram below.
Traffic Ops  Traffic Ops stores the configuration of cache servers and CDN Delivery Services. It also serves the Traffic Ops API which can be used by tools, scripts, and programs to access and manipulate CDN data. Traffic Ops used to provide a UI for interacting with the CDN(s) that it manages.

Deprecated since version 3.0: Traffic Control 2.2 introduced the Traffic Portal UI which is the recommended, official method of manipulating CDN data. As of Traffic Control 3.0 the Traffic Ops UI is deprecated and disabled by default, and will be removed by the release of 4.0.

Traffic Router  Traffic Router is used to route client requests to the closest healthy cache server by analyzing the health, capacity, and state of the cache servers according to the Health Protocol and relative geographic distance between each Cache Group and the client.

Traffic Monitor  Traffic Monitor does health polling of the cache servers on a very short interval to keep track of which servers should be kept in rotation.

See also:

Health Protocol

Traffic Stats  Traffic Stats collects and stores real-time traffic statistics aggregated from each of the cache servers. This data is used by the Traffic Router to assess the available capacity of each cache server which it uses to balance traffic load and prevent overload.

Traffic Portal  Traffic Portal is a web interface which uses the Traffic Ops API to present CDN data and the controls to manipulate it in a user-friendly interface.

New in version 2.2: As of Traffic Control 2.2, this is the recommended, official UI for the Traffic Control platform. In Traffic Control 3.x, the Traffic Ops UI has been deprecated and disabled by default, and it will be removed with the release of Traffic Control 4.0.
Traffic Vault

Traffic Vault is used as a secure key/value store for SSL private keys used by other Traffic Control components.

2.1.2 Traffic Ops

Traffic Ops is the tool for administration (configuration and monitoring) of all components in a Traffic Control CDN. Traffic Portal uses the Traffic Ops API to manage servers, Cache Groups, Delivery Services, etc. In many cases, a configuration change requires propagation to several, or even all, cache servers 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 uses a PostgreSQL database to store the configuration information, and a combination of the Mojolicious framework and Go to provide the Traffic Ops API. Not all configuration data is in this database however; for sensitive data like private SSL keys or token-based authentication shared secrets, Traffic Stats is used as a separate, key/value store, allowing administrators to harden the Traffic Stats server better from a security perspective (i.e only allow Traffic Ops to access it, verifying authenticity with a certificate). 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 cache servers and other servers. The cache servers and other servers check in with Traffic Ops at a regular interval to see if updated configuration files require application. On cache servers this is done by the ORT script.

Traffic Ops also runs a collection of periodic checks to determine the operating state of the cache servers. These periodic checks are customizable by the Traffic Ops administrative user using Traffic Ops Extensions.

Traffic Ops is in the process of migrating from Perl to Go, and currently runs as two separate applications. The Go application serves all endpoints which have been rewritten in the Go language, and transparently proxies all other requests to the old Perl application. For this reason, users and administrators should direct all requests solely at the Go-based implementation. Both applications are installed by the RPM, and both run as a single service. When the project has fully migrated to Go, the Perl application will be removed, and the RPM and service will consist solely of the Go application.

Traffic Ops Extension

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

Check Extensions  Allow you to add custom checks to the Monitor → Cache Checks view in Traffic Portal.

Data Source Extensions  Allow you to add data sources for the graph views and usage APIs.
2.1.3 Traffic Portal

Traffic Portal is an AngularJS 1.x client served from a Node.js web server designed to consume the Traffic Ops API. It is the official replacement for the legacy Traffic Ops UI.

Features

- CDN Monitoring
- CDN Administration
- Delivery Service Configuration
- Cache Server Maintenance

See also:

See Traffic Portal - Using for an overview of the Traffic Portal UI.

2.1.4 Traffic Router

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

- Distance between the cache server and the client (not necessarily measured in physical distance, but quite often in layer 3 network hops). Less network distance between the client and cache server yields better performance and lower network load. Traffic Router helps clients connect to the best-performing cache server for their location at the lowest network cost.

- Availability of cache servers and the system processing/network load on the cache servers. 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 purposely disabled cache servers.

- Availability of content on a particular cache server. Reusing of content through “cache hits” is the most important performance gain a CDN can offer. Traffic Router sends clients to the cache server that is most likely to already have the desired content.

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

DNS Content Routing

For a DNS Delivery Service the client might receive a URL such as http://video.demol.mycdn.ciab.test/. When the LDNS is resolving this video.demol.mycdn.ciab.test hostname to an IP address, it ends at Traffic Router because it is the authoritative DNS server for mycdn.ciab.test and the domains below it, and subsequently responds with a list of IP addresses from the eligible cache servers based on the location of the LDNS. 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 server IP address (or list
of cache server IP addresses) to return is solely based on the location of the LDNS and the health of the cache servers. The client then connects to port 80 (HTTP) or port 443 (HTTPS) on the cache server, and sends the Host: video.demo1.mycdn.ciab.test header. The configuration of the cache server includes the “remap rule” http://video.demo1.mycdn.ciab.test http://origin.infra.ciab.test to map the routed name to an origin hostname.

HTTP Content Routing

For an HTTP Delivery Service the client might receive a URL such as http://video.demo1.mycdn.ciab.test/. The LDNS resolves this video.demo1.mycdn.ciab.test to an IP address, but in this case Traffic Router returns its own IP address. The client opens a connection to port 80 (HTTP) or port 443 (HTTPS) on the Traffic Router’s IP address, and sends its request.

# 7: Example Client Request to Traffic Router

```
GET / HTTP/1.1
Host: video.demo1.mycdn.ciab.test
Accept: */*
```

Traffic Router uses an HTTP 302 Found response to redirect the client to the best cache server.

# 8: Traffic Router Redirect to Edge-tier Cache Server

```
HTTP/1.1 302 Found
Location: http://edge.demo1.mycdn.ciab.test/
Content-Length: 0
Date: Tue, 13 Jan 2015 20:01:41 GMT
```

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

- The client’s IP address.
- The URL path the client is requesting, excluding query string.
- All HTTP/1.1 headers.

The client follows the redirect and performs a DNS request for the IP address for edge.demo1.mycdn.ciab.test, and normal HTTP steps follow, except the sending of the Host: header when connected to the cache is Host: edge.demo1.mycdn.ciab.test, and the configuration of the cache server includes the “remap rule” (e.g. “http://edge.demo1.mycdn.ciab.test http://origin.infra.ciab.test”). Traffic Router sends all requests for the same path in a Delivery Service to the same cache server in a Cache Group using consistent hashing, in this case all cache servers 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. Traffic Control Overview
2.1.5 Traffic Monitor

Traffic Monitor is an HTTP service that monitors the cache servers in a CDN for a variety of metrics. These metrics are for use in determining the overall “health” of a given cache server 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 server health to upstream applications such as Traffic Router. Health Protocol governs the cache server 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 cache servers configured with a status of REPORTED or ADMIN_DOWN at an interval specified as a configuration parameter in Traffic Ops. If the cache server is set to ADMIN_DOWN it is marked as unavailable but still polled for availability and statistics. If the cache server 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-tier cache server and consumes the JSON output. The special URL is served by a plugin running on the ATS cache servers 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 level. Traffic Monitor uses the system-level performance to determine the overall health of the cache server by evaluating network throughput and load against values configured in Traffic Ops. Traffic Monitor also uses throughput and transaction statistics at the Delivery Service 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 server 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 server 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 Cache Monitoring, the behavior of a single Traffic Monitor instance is described. 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 server 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 cache servers, 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 server 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 cache servers and Traffic Monitor peers help to reduce customer impact of outages. It is not uncommon for a cache server to be marked unavailable by Traffic Monitor - in fact, it is business as usual for many CDNs. Should a widely requested video asset cause a single cache server to get close to its interface capacity, the Health Protocol will “kick in”, and Traffic Monitor marks the cache server as unavailable. New clients want to see the same asset, and now Traffic Router will send these customers to another cache server in the same Cache Group. The load is now shared between the two cache servers. As clients finish watching the asset on the overloaded cache server, it will drop below the threshold and gets marked available again, and new clients will begin to be directed to it once more. 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 being impacted.

2.1.6 Traffic Stats

Traffic Stats is a program written in Go that is used to acquire and store statistics about CDNs controlled by Traffic Control. Traffic Stats mines metrics from the Traffic Monitor APIs and stores the data in InfluxDB. Data is typically stored in InfluxDB on a short-term basis (30 days
or less). The data from InfluxDB is then used to drive graphs created by Grafana - which are linked to from Traffic Portal - as well as provide data exposed through the Traffic Ops API. Traffic Stats performs two functions:

- Gathers statistics for Edge-tier cache servers and Delivery Services at a configurable interval (10 second default) from the Traffic Monitor APIs and stores the data in InfluxDB
- Summarizes all of the statistics once a day (around midnight UTC) and creates a daily report containing the Max GBPS (Gigabits per second) Served and the Total Bytes Served.

Statistics are stored in three different databases:

- cache_stats: Stores data gathered from edge-tier cache servers. The measurements stored by cache_stats are: bandwidth, maxKbps, and client_connections (ats.proxy.process.http.current_client_connections). Cache Data is stored with tags for hostname, Cache Group, and CDN. Data can be queried using tags.
- deliveryservice_stats: Stores data for Delivery Services. The measurements stored by Delivery Service are:
  - kbps
  - status_4xx
  - status_5xx
  - tps_2xx
  - tps_3xx
  - tps_4xx
  - tps_5xx
  - tps_total

  Delivery Service statistics are stored with tags for Cache Group, CDN, and Delivery Service xml_id.
- daily_stats: Stores summary data for daily activities. The statistics that are currently summarized are Max Bandwidth and Bytes Served and they are stored by CDN.

Traffic Stats does not influence overall CDN operation, but is required in order to display charts in Traffic Portal.

2.1.7 Traffic Vault

Traffic Vault is a key/value store used for storing the following types of information:

- SSL Certificates
  - Private Key
  - Certificate
  - CSR (Certificate Signing Request)
- DNSSEC (DNS Security Extensions) 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 administrative privileges. Keys can be created via the Traffic Portal 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.

See also:

Information on the Riak API can be found in their documentation.

2.1.8 Delivery Services

“Delivery Services” are a very important construct in ATC (Apache Traffic Control). At their most basic, they are a source of content and a set of cache servers and configuration options used to distribute that content.

Delivery Services are modeled several times over, in the Traffic Ops database, in Traffic Portal forms and tables, in the legacy Perl Traffic Ops codebase, and several times for various Traffic Ops API versions in the new Go Traffic Ops codebase. Go-specific data structures can be found in the project’s GoDoc documentation. Rather than application-specific definitions, what follows is an attempt at consolidating all of the different properties and names of properties of Delivery Service objects throughout the ATC suite. The names of these fields are typically chosen as the most human-readable and/or most commonly-used names for the fields, and when reading please note that in many cases these names will appear camelCased or snake_cased to be machine-readable. Any aliases of these fields that are not merely case transformations of the indicated, canonical names will be noted in a table of aliases.

See also:

The API reference for Delivery Service-related endpoints such as deliverieservices contains definitions of the Delivery Service object(s) returned and/or accepted by those endpoints.

Active

Whether or not this Delivery Service is active on the CDN and can be served. When a Delivery Service is not “active”, Traffic Router will not be made aware of its existence - i.e. it will not appear in CDN Snapshots. Setting a Delivery Service to be “active” (or “inactive”) will require that a new Snapshot be taken.
Anonymous Blocking

Enables/Disables blocking of anonymized IP address - proxies, TOR (The Onion Ring) exit nodes, etc - for this Delivery Service. Set to true to enable blocking of anonymous IPs for this Delivery Service.

Table 1: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous-BlockingEnabled</td>
<td>Traffic Ops client and server Go code, Traffic Ops API requests and responses</td>
<td>usually unchanged (boolean), but sometimes as a string containing a boolean e.g. in the response of a GET request to cdns/{{name}}/snapshot</td>
</tr>
</tbody>
</table>

Note: Anonymous Blocking requires an anonymous IP address database from the Delivery Service’s Geolocation Provider. E.g. MaxMind’s Anonymous IP Database when MaxMind is used as the Geolocation Provider.

See also:
The Configure Anonymous Blocking “Quick-How-To” guide.

Cache URL Expression

Deprecated since version 3.0: This feature is no longer supported by ATS and consequently it will be removed from Traffic Control in the future.

Manipulates the cache key of the incoming requests. Normally, the cache key is the origin domain. This can be changed so that multiple services can share a cache key, can also be used to preserve cached content if service origin is changed.

Warning: This field provides access to a feature that was only present in ATS 6.X and earlier. As cache servers must now use ATS 7.1.X, this field must be blank unless all cache servers can be guaranteed to use that older ATS version (NOT recommended).

CDN

A CDN to which this Delivery Service belongs. Only cache servers within this CDN are available to route content for this Delivery Service. Additionally, only Traffic Routers assigned to this CDN will perform said routing. Most often cdn/CDN refers to the name of the CDN to which the Delivery Service belongs, but occasionally (most notably in the payloads and/or query parameters of certain Traffic Ops API endpoints) it actually refers to the integral, unique identifier of said CDN.
Check Path

A request path on the origin server which is used to by certain Traffic Ops Extensions to indicate the “health” of the origin.

Deep Caching

Controls the Deep Caching feature of Traffic Router when serving content for this Delivery Service. This should always be represented by one of two values:

**ALWAYS** This Delivery Service will always use Deep Caching

**NEVER** This Delivery Service will never use Deep Caching

Implementation Detail

Traffic Ops and Traffic Ops client Go code use an empty string as the name of the enumeration member that represents “NEVER”.

Display Name

The “name” of the Delivery Service. Since nearly any use of a string-based identification method for Delivery Services (e.g. in Traffic Portal tables) uses xml_id, this is of limited use. For that reason and for consistency’s sake it is suggested that this be the same as the xml_id. However, unlike the xml_id, this can contain any UTF-8 characters without restriction.

DNS Bypass CNAME

When the limits placed on this Delivery Service by the Global Max Mbps and/or Global Max Tps are exceeded, a DNS-Routed Delivery Service will direct excess traffic to the host referred to by this CNAME (Canonical Name) record.

**Note:** IPv6 traffic will be redirected if and only if IPv6 Routing Enabled is “true” for this Delivery Service.

DNS Bypass IP

When the limits placed on this Delivery Service by the Global Max Mbps and/or Global Max Tps are exceeded, a DNS-Routed Delivery Service will direct excess IPv4 traffic to this IPv4 address.
DNS Bypass IPv6

When the limits placed on this Delivery Service by the *Global Max Mbps* and/or *Global Max Tps* are exceeded, a DNS-Routed Delivery Service will direct excess IPv6 traffic to this IPv6 address.

**Note:** This requires an accompanying configuration of *IPv6 Routing Enabled* such that IPv6 traffic is allowed at all.

DNS Bypass TTL

When the limits placed on this Delivery Service by the *Global Max Mbps* and/or *Global Max Tps* are exceeded, a DNS-Routed Delivery Service will direct excess traffic to their DNS Bypass IP, DNS Bypass IPv6, or DNS Bypass CNAME.

DNS TTL

The TTL on the DNS record for the Traffic Router A and AAAA records. DNS-Routed Delivery Services will send this TTL along with their record responses to clients requesting access to this Delivery Service. Setting too high or too low will result in poor caching performance.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCR DNS TTL</td>
<td>In Delivery Service objects returned by the Traffic Ops API</td>
<td>unchanged (int, integer etc.)</td>
</tr>
<tr>
<td>CCR TTL</td>
<td>Legacy Traffic Ops UI, documentation for older Traffic Control versions</td>
<td>unchanged (int, integer etc.)</td>
</tr>
<tr>
<td>ttl</td>
<td>In CDN Snapshot structures, where it is displayed on a per-record-type-basis</td>
<td>map of record type names to integral values</td>
</tr>
</tbody>
</table>

DSCP

The DSCP (Differentiated Services Code Point) which will be used to mark IP packets as they are sent out of the CDN to the client.

**See also:**

The Differentiated Services Wikipedia article.

**Warning:** The DSCP setting in Traffic Portal is *only* for setting traffic towards the client, and gets applied *after* the initial TCP handshake is complete and the HTTP request has been received. Before that the cache can’t determine what Delivery Service is being requested,
and consequently can’t know what DSCP to apply. Therefore, the DSCP feature can not be used for security settings; the IP packets that form the TCP handshake are not going to be DSCP-marked.

Implementation Detail

DSCP settings only apply on cache servers that run Apache Traffic Server. The implementation uses the ATS Header Rewrite Plugin to create a rule that will mark traffic bound outward from the CDN to the client.

Edge Header Rewrite Rules

This field in general contains the contents of the a configuration file used by the ATS Header Rewrite Plugin when serving content for this Delivery Service - on Edge-tier cache servers.

Tip: Because this ultimately is the contents of an ATS configuration file, it can make use of the Strings with Special Meaning to ORT.

Fair-Queuing Pacing Rate Bps

The maximum bytes per second a cache server will deliver on any single TCP connection. This uses the Linux kernel's Fair-Queuing setsockopt(2) (SO_MAX_PACING_RATE) to limit the rate of delivery. Traffic exceeding this speed will only be rate-limited and not diverted. This option requires extra configuration on all cache servers assigned to this Delivery Service - specifically, the line net.core.default_qdisc = fq must exist in /etc/sysctl.conf.

See also:
tc-fq_codel(8)

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FQPacingRate</td>
<td>Traffic Ops source code, Delivery Service objects returned by the Traffic Ops API</td>
<td>unchanged (int, integer etc.)</td>
</tr>
</tbody>
</table>

Geo Limit

Limits access to a Delivery Service by geographic location. The only practical difference between this and Regional Geoblocking is the configuration method; as opposed to Regional Geoblocking, GeoLimit configuration is handled by country-wide codes and the Coverage Zone

2.1. Traffic Control Overview
When a client is denied access to a requested resource on an HTTP-*Routed* Delivery Service, they will receive a 503 Service Unavailable instead of the usual 302 Found response - unless *Geo Limit Redirect URL* is defined, in which case a 302 Found response pointing to that URL will be returned by Traffic Router. If the Delivery Service is a DNS-*Routed* Delivery Service, the IP address of the *resolver* for the client DNS request is what is checked. If the IP address of this resolver is found to be in a restricted location, the Traffic Router will respond with an NXDOMAIN response, causing the name resolution to fail. This is nearly always an integral, unique identifier for a behavior set to be followed by Traffic Router. The defined values are:

0 Geographic access limiting is not enabled, and content served by this Delivery Service will be accessible regardless of the clients geographic location. (Aliased as “0 - None” in Traffic Portal forms)

1 A client will be allowed to request content if and only if their IP address is found by Traffic Router within the *Coverage Zone File*. Otherwise, access will be denied. (Aliased as “1 - CZF Only” in Traffic Portal forms)

2 A client will be allowed to request content if their IP address is found by Traffic Router within the *Coverage Zone File*, or if looking up the client’s IP address in the Geographic IP mapping database provided by *Geolocation Provider* indicates the client resides in a country that is found in the *Geo Limit Countries* array. (Aliased as “2 - CZF + Country Code(s)” in Traffic Portal forms - formerly was known as “CZF + US” when only the US country code was supported)

**Warning:** The definitions of each integral, unique identifier are hidden in implementations in each ATC component. Different components will handle invalid values differently, and there’s no actual enforcement that the stored integral, unique identifier actually be within the representable range.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>coverageZoneOnly</td>
<td>In CDN <em>Snapshot</em> structures, especially in <em>Traffic Ops API</em> responses</td>
<td>A boolean which, if <em>true</em>, tells Traffic Router to only service requests when the client IP address is found in the <em>Coverage Zone File</em></td>
</tr>
</tbody>
</table>

**Danger:** Geographic access limiting is **not** sufficient to guarantee access is properly restricted. The limiting is implemented by Traffic Router, which means that direct requests to *Edge-tier cache servers* will bypass it entirely.

**Geo Limit Countries**

When *Geo Limit* is being used with this Delivery Service (and is set to exactly 2), this is optionally a list of country codes to which access to content provided by the Delivery Service
will be restricted. Normally, this is a comma-delimited string of said country codes. When creating a Delivery Service with this field or modifying the Geo Limit Countries field on an existing Delivery Service, any amount of whitespace between country codes is permissible, as it will be removed on submission, but responses from the Traffic Ops API should never include such whitespace.

### Table 5: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>geoEnabled</td>
<td>In CDN Snapshot structures, especially in Traffic Ops API responses</td>
<td>An array of objects each having the key “countryCode” that is a string containing an allowed country code - one should exist for each allowed country code</td>
</tr>
</tbody>
</table>

**Geo Limit Redirect URL**

If Geo Limit is being used with this Delivery Service, this is optionally a URL to which clients will be redirected when Traffic Router determines that they are not in a geographic zone that permits their access to the Delivery Service content. This changes the response from Traffic Router from 503 Service Unavailable to 302 Found with a provided location that will be this URL. There is no restriction on the provided URL; it may even be the path to a resource served by this Delivery Service. In fact, this field need not even be a full URL, it can be a relative path. Both of these cases are handled specially by Traffic Router.

- If the provided URL is a resource served by the Delivery Service (e.g. if the client requests http://cdn.dsXMLID.somedomain.example.com/index.html but are denied access by Geo Limit and the Geo Limit Redirect URL is something like http://cdn.dsXMLID.somedomain.example.com/help.php), Traffic Router will find an appropriate Edge-tier cache server and redirect the client, ignoring Geo Limit restrictions for this request only.

- If the provided “URL” is actually a relative path, it will be considered relative to the requested Delivery Service :abbr:`FQDN (Fully Qualified Domain Name)`. This means that e.g. if the client requests http://cdn.dsXMLID.somedomain.example.com/index.html but are denied access by Geo Limit and the Geo Limit Redirect URL is something like /help.php, Traffic Router will find an appropriate Edge-tier cache server and redirect the client to it as though they had requested http://cdn.dsXMLID.somedomain.example.com/help.php, ignoring Geo Limit restrictions for this request only.
Table 6: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGB (National GeoBlock)</td>
<td>Older documentation, in Traffic Router comments and error logs</td>
<td>unchanged (string, String etc.)</td>
</tr>
<tr>
<td>geoRedirectURLType</td>
<td>Internally in Traffic Router</td>
<td>A String that describes whether or not the actual Geo Limit Redirect URL is relative to the Delivery Service base FQDN (Fully Qualified Domain Name). Should be one of: INVALID_URL The Geo Limit Redirect URL has not yet been parsed, or an error occurred during parsing DS_URL The Geo Limit Redirect URL is served by this Delivery Service NOT_DS_URL The Geo Limit Redirect URL is external to this Delivery Service</td>
</tr>
</tbody>
</table>

**Note:** The use of a redirect URL relies on the ability of Traffic Router to redirect the client using HTTP 302 Found responses. As such, this field has no effect on DNS-**Routed** Delivery Services.

**Geolocation Provider**

This is nearly always the integral, unique identifier of a provider for a database that maps IP addresses to geographic locations. Less frequently, this may be accompanied by the actual name of the provider. Only two values are possible at the time of this writing:

0: **MaxMind** IP address to geographic location mapping will be provided by a MaxMind GeoIP2 database.

1: **Neustar** IP address to geographic location mapping will be provided by a Neustar GeoPoint IP address database.
Warning: It’s not clear whether Neustar databases are actually supported; this is an old option and compatibility may have been broken over time.

Table 7: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo-</td>
<td>Traffic Ops and Traffic Ops client code, <em>Traffic Ops API</em> requests and responses</td>
<td>unchanged (integral, unique identifier)</td>
</tr>
<tr>
<td>Provider</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Geo Miss Default Latitude

Default Latitude for this Delivery Service. When the geographic location of the client cannot be determined, they will be routed as if they were at this latitude.

Table 8: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>missLat</td>
<td>In <em>Traffic Ops API</em> responses and Traffic Ops source code</td>
<td>unchanged (numeric)</td>
</tr>
</tbody>
</table>

Geo Miss Default Longitude

Default Longitude for this Delivery Service. When the geographic location of the client cannot be determined, they will be routed as if they were at this longitude.

Table 9: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>miss-</td>
<td>In <em>Traffic Ops API</em> responses and Traffic Ops source code</td>
<td>unchanged (numeric)</td>
</tr>
<tr>
<td>Long</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Global Max Mbps

The maximum MBPS (Megabits per second) this Delivery Service can serve across all Edge-tier cache servers before traffic will be diverted to the bypass destination. For a DNS-Routed Delivery Service, the *DNS Bypass IP* or *DNS Bypass IPv6* will be used (depending on whether this was a A or AAAA request), and for HTTP-Routed Delivery Services the *HTTP Bypass FQDN* will be used.

Table 10: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalKbps-threshold</td>
<td><em>Traffic Ops API</em> responses - most notably cdns/{{name}}/configs/monitoring</td>
<td>unchanged (numeric), but converted from MBPS to KBPS (kilobits per second)</td>
</tr>
</tbody>
</table>
Global Max TPS

The maximum TPS (Transactions per Second) this Delivery Service can serve across all *Edge-tier cache servers* before traffic will be diverted to the bypass destination. For a *DNS-Routed* Delivery Service, the *DNS Bypass IP* or *DNS Bypass IPv6* will be used (depending on whether this was a A or AAAA request), and for *HTTP-Routed* Delivery Services the *HTTP Bypass FQDN* will be used.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalTp-sThreshold</td>
<td>In <em>Traffic Ops API</em> responses - most notably cdns/{{name}}/configs/monitoring</td>
<td>unchanged (numeric)</td>
</tr>
</tbody>
</table>

HTTP Bypass FQDN

When the limits placed on this Delivery Service by the *Global Max Mbps* and/or *Global Max Tps* are exceeded, an *HTTP-Routed* Delivery Service will direct excess traffic to this **FULLY QUALIFIED DOMAIN NAME**.

IPv6 Routing Enabled

A boolean value that controls whether or not clients using IPv6 can be routed to this Delivery Service by Traffic Router. When creating a Delivery Service in Traffic Portal, this will default to “true”.

Info URL

This should be a URL (though neither the *Traffic Ops API* nor the Traffic Ops Database in any way enforce the validity of said URL) to which administrators or others may refer for further information regarding a Delivery Service - e.g. a related JIRA ticket.

Initial Dispersion

The number of *Edge-tier cache servers* across which a particular asset will be distributed within each *Cache Group*. For most use-cases, this should be 1, meaning that all clients requesting a particular asset will be directed to 1 *cache server* per *Cache Group*. Depending on the popularity and size of assets, consider increasing this number in order to spread the request load across more than 1 *cache server*. The larger this number, the more copies of a particular asset are stored in a *Cache Group*, which can “pollute” caches (if load distribution is unnecessary) and decreases caching efficiency (due to cache misses if the asset is not requested enough to stay “fresh” in all the caches).
Logs Enabled

A boolean switch that can be toggled to enable/disable logging for a Delivery Service.

**Note:** This doesn’t actually do anything. It was part of the functionality for a planned Traffic Control component named “Traffic Logs” - which was never created.

Long Description

Free text field that has no strictly defined purpose, but it is suggested that it contain a short description of the Delivery Service and its purpose.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>longDesc</td>
<td>Traffic Control source code and <em>Traffic Ops API</em> responses</td>
<td>unchanged (string, String etc.)</td>
</tr>
</tbody>
</table>

Long Description 2

Free text field that has no strictly defined purpose.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>longDesc2</td>
<td>Traffic Control source code and <em>Traffic Ops API</em> responses</td>
<td>unchanged (string, String etc.)</td>
</tr>
</tbody>
</table>

Long Description 3

Free text field that has no strictly defined purpose.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>longDesc3</td>
<td>Traffic Control source code and <em>Traffic Ops API</em> responses</td>
<td>unchanged (string, String etc.)</td>
</tr>
</tbody>
</table>

Match List

A Match List is a set of regular expressions used by Traffic Router to determine whether a given request from a client should be served by this Delivery Service. Under normal circumstances this field should only ever be read-only as its contents should be generated by Traffic Ops based on the Delivery Service’s configuration. These regular expressions can each be one of the following types:

2. In source code and *Traffic Ops API* responses, the “Long Description” fields of a Delivery Service are “0-indexed” - hence the names differing slightly from the ones displayed in user-friendly UIs.
HEADER_REGEXP  This Delivery Service will be used if an HTTP Header/Value pair can be found in the clients request matching this regular expression.\(^4\)

HOST_REGEXP  This Delivery Service will be used if the requested host matches this regular expression. The host can be found using the Host HTTP Header, or as the requested name in a DNS request, depending on the Type of the Delivery Service.

PATH_REGEXP  This Delivery Service will be used if the request path matches this regular expression.\(^4\)

STEERING_REGEXP  This Delivery Service will be used if this regular expression matches the xml\_id of one of this Delivery Service’s “targets”

---

**Note:** This regular expression type can only exist in the Match List of STEERING-Type Delivery Services - and **not** CLIENT\_STEERING.

---

### Table 12: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>deliveryservice_regex</td>
<td>Traffic Ops database</td>
<td>unique, integral identifier for a regular expression</td>
</tr>
</tbody>
</table>

---

**Max DNS Answers**

The maximum number of Edge-tier cache server IP addresses that the Traffic Router will include in responses to DNS requests for DNS-Routed Delivery Services. The Traffic Ops API restricts this value to the range \([1, 15]\), but no matching restraints are placed on the actual data as stored in the Traffic Ops Database. When provided, the cache server IP addresses included are rotated in each response to spread traffic evenly. This number should scale according to the amount of traffic the Delivery Service is expected to serve.

---

**Mid Header Rewrite Rules**

This field in general contains the contents of the a configuration file used by the ATS Header Rewrite Plugin when serving content for this Delivery Service - on Mid-tier cache servers.

---

**Tip:** Because this ultimately is the contents of an ATS configuration file, it can make use of the Strings with Special Meaning to ORT.

---

**Origin Server Base URL**

The Origin Server’s base URL which includes the protocol (http or https). Example: [http://movies.origin.com](http://movies.origin.com). Must not include paths, query parameters, document fragment

---

\(^4\) These regular expression types can only appear in the Match List of HTTP-Routed Delivery Services.
identifiers, or username/password URL fields.

Table 13: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgServer-Fqdn</td>
<td>Traffic Ops API responses and in Traffic Control source code</td>
<td>unchanged (usually str, string etc.)</td>
</tr>
</tbody>
</table>

**Origin Shield**

An experimental feature that allows administrators to list additional forward proxies that sit between the *Mid-tier* and the *origin*. In most scenarios, this is represented (and required to be input) as a pipe (|)-delimited string.

**Profile**

Either the name of a *Profile* used by this Delivery Service, or an integral, unique identifier for said *Profile*.

Table 14: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>profileId</td>
<td>In Traffic Control source code and some Traffic Ops API responses dealing with Delivery Services</td>
<td>Unlike the more general “Profile”, this is always an integral, unique identifier</td>
</tr>
<tr>
<td>profileName</td>
<td>In Traffic Control source code and some Traffic Ops API responses dealing with Delivery Services</td>
<td>Unlike the more general “Profile”, this is always a name (str, string, etc.)</td>
</tr>
</tbody>
</table>

**Protocol**

The protocol with which to serve content from this Delivery Service. This defines the way the Delivery Service will handle client requests that are either HTTP or HTTPS, which is distinct from what protocols are used to direct traffic. For example, this can be used to direct clients to only request content using HTTP, or to allow clients to use either HTTP or HTTPS, etc. Normally, this will be the name of the protocol handling, but occasionally this will appear as the integral, unique identifier of the protocol handling instead. The integral, unique identifiers and their associated names and meanings are:

0: **HTTP** This Delivery Service will only accept unsecured HTTP requests. Requests made with HTTPS will fail.

1: **HTTPS** This Delivery Service will only accept secured HTTPS requests. Requests made with HTTP will fail.

2: **HTTP AND HTTPS** This Delivery Service will accept both unsecured HTTP requests and secured HTTPS requests.
3: HTTP TO HTTPS  When this Delivery Service is using HTTP Content Routing unsecured HTTP requests will be met with a response that indicates to the client that further requests must use HTTPS.

**Note:** If any other type of Content Routing is used, this functionality cannot be used. In those cases, a protocol setting of 3/"HTTP TO HTTPS" will result in the same behavior as 1/"HTTPS". This behavior is tracked by GitHub Issue #3221

---

**Warning:** The definitions of each integral, unique identifier are hidden in implementations in each ATC component. Different components will handle invalid values differently, and there's no actual enforcement that the stored integral, unique identifier actually be within the representable range.

---

### Table 15: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pro-</td>
<td>CDN</td>
<td>An object containing the key &quot;acceptHttps&quot; that is a string containing a boolean that expresses whether Traffic Router should accept HTTPS requests for this Delivery Service, and the key &quot;redirectToHttps&quot; that is also a string containing a boolean which expresses whether or not Traffic Router should redirect HTTP requests to HTTPS URLs. Optionally, the key &quot;acceptHttp&quot; may also appear, once again a string containing a boolean that expresses whether or not Traffic Router should accept unsecured HTTP requests - this is implicitly treated as &quot;true&quot; by Traffic Router when it is not present.</td>
</tr>
<tr>
<td>to-</td>
<td>Snap-</td>
<td>shot</td>
</tr>
<tr>
<td>col</td>
<td>shots</td>
<td></td>
</tr>
</tbody>
</table>

### Query String Handling

Describes how query strings should be handled by the **Edge-tier cache servers** when serving content for this Delivery Service. This is nearly always expressed as an integral, unique identifier for each behavior, though in Traffic Portal a more descriptive value is typically used, or at least provided in addition to the integral, unique identifier. The allowed values and their meanings are:

- **0** For the purposes of caching, *Edge-tier cache servers* will consider URLs unique if and only if they are unique up to and including any and all query parameters. They will also pass the query parameters in their own requests to *Mid-tier cache servers* (which in turn will exhibit the same caching behavior and pass the query parameters in requests to the *origin*). (Aliased as “USE” in Traffic Portal tables, and “0 - use qstring in cache key, and pass up” in Traffic Portal forms)

- **1** For the purposes of caching, neither *Edge-tier* nor *Mid-tier cache servers* will consider the query parameter string when determining if a URL is stored in cache. However, the query string will still be passed in upstream requests to *Mid-tier cache servers* and in turn the *origin*. (Aliased as “IGNORE” in Traffic Portal tables and “1 - ignore in cache key, and
The query parameter string will be stripped from URLs immediately when the request is received by an **Edge-tier cache server**. This means it is never considered for the purposes of caching unique URLs and will not be passed in upstream requests. (Aliased as “DROP” in Traffic Portal tables and “2 - drop at edge” in Traffic Portal forms)

**Warning:** The implementation of dropping query parameter strings at the **Edge-tier** uses a **Regex Remap Expression** and thus Delivery Services with this type of query string handling cannot make use of **Regex Remap Expressions**.

### Table 16: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qstring Handling</td>
<td>Traffic Portal tables</td>
<td>One of the Traffic Portal value aliases “USE” (0), “IGNORE” (1), “DROP” (2)</td>
</tr>
<tr>
<td>qstringIg-nore</td>
<td>Traffic Ops Go/Perl code, <strong>Traffic Ops API</strong> requests/responses</td>
<td>unchanged (integral, unique identifier)</td>
</tr>
</tbody>
</table>

The Delivery Service’s Query String Handling can be set directly as a field on the Delivery Service object itself, or it can be overridden by a **Parameter** on a **Profile** used by this Delivery Service. The special **Parameter** named `psel.qstring_handling` and configuration file `parent.config` will have it’s contents directly inserted into the `parent.config` file on all **cache servers** assigned to this Delivery Service.

**Danger:** Using the `psel.qstring_handling` **Parameter** is **strongly** discouraged for several reasons. Firstly, at a Delivery Service level it will **NOT** change the configuration of that Delivery Service’s own Query String Handling - which will cause it to appear in Traffic Portal and in **Traffic Ops API** responses as though it were configured one way while actually behaving a different way altogether. Also, no validation is performed on the value given to it. Because it’s inserted verbatim into the `qstring` field of a line in ATS `parent.config` configuration file, a typo or an ignorant user can easily cause ATS instances on all **cache servers** assigned to that Delivery Service to fail to reload their configuration, possibly grinding entire CDNs to a halt.

**See also:**

When implemented as a **Parameter** (`psel.qstring_handling`), its value must be a valid value for the `qstring` field of a line in the **ATS parent.config** configuration file. For a description of valid values, see the **documentation for parent.config**

### Range Request Handling

Describes how HTTP “Range Requests” should be handled by the Delivery Service at the **Edge-tier**. This is nearly always an integral, unique identifier for the behavior set required of the **Edge-tier cache servers**. The valid values and their respective meanings are:
0 Do not cache Range Requests at all. (Aliased as “0 - Don’t cache” in Traffic Portal forms)

Note: This is not retroactive - when modifying an existing Delivery Services to have this value for “Range Request Handling”, ranges requested from files that are already cached due to a non-range request will be served out of cache for as long as the Cache-Control headers allow.

1 Use the background_fetch plugin to service the range request while caching the whole object. (Aliased as “1 - Use background_fetch plugin” in Traffic Portal forms)

2 Use the cache_range_requests plugin to cache ranges as unique objects. (Aliased as “2 - Use cache_range_requests plugin” in Traffic Portal forms)

Note: Range Request Handling can only be implemented on cache servers using ATS because of its dependence on ATS plugins. The value may be set on any Delivery Service, but will have no effect when the cache servers that ultimately end up serving the content are e.g. Grove, Nginx, etc.

Warning: The definitions of each integral, unique identifier are hidden in implementations in each ATC component. Different components will handle invalid values differently, and there’s no actual enforcement that the stored integral, unique identifier actually be within the representable range.

Raw Remap Text

For HTTP and DNS-Routed Delivery Services, this will be added to the end of a line in the remap.config ATS configuration file line on the cache verbatim. For ANY_MAP-Type Delivery Services this must be defined.

Tip: Because this ultimately is a raw line of content in a configuration file, it can make use of the Strings with Special Meaning to ORT. Of particular note is the Remap Override template string.

Note: This field must be defined on ANY_MAP-Type Delivery Services, but is otherwise optional.

Table 17: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>remap-Text</td>
<td>In Traffic Ops source code and Traffic Ops API requests/responses</td>
<td>unchanged, (text, string etc.)</td>
</tr>
</tbody>
</table>
Regex Remap Expression

Allows remapping of incoming requests URL using regular expressions to search and replace text. In a more literal sense, this is the raw contents of a configuration file used by the ATS regex_remap plugin. At its most basic, the contents of this field should consist of map followed by a regular expression and then a “template URL” - all space-separated. The regular expression matches a client’s request path (i.e. not a full URL - /path/to/content and not https://origin.example.com/path/to/content) and when such a match occurs, the request is transformed into a request for the template URL. The most basic usage of the template URL is to use $1-$9 to insert the corresponding regular expression capture group. For example, a regular expression of ^/a/(.*) and a template URL of https://origin.example.com/b/$1 maps requests for origin content under path /a/ to the same sub-paths under path b. Note that since it’s a full URL, this mapping can be made to another server entirely.

See also:
The documentation for the regex_remap plugin for ATS

Caution: This field is not validated by Traffic Ops to be correct syntactically, and can cause Traffic Server to not start if invalid. Please use with caution.

Warning: Regex remap expressions are incompatible with Query String Handling being set to 2. The behavior of a cache server under that configuration is undefined.

Tip: It is, of course, entirely possible to write a Regex Remap Expression that reproduces the desired Query String Handling as well as any other desired behavior.

Table 18: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>regexRemap</td>
<td>Traffic Ops source code and database, and Traffic Ops API</td>
<td>unchanged (string etc.)</td>
</tr>
</tbody>
</table>

Regional Geoblocking

A boolean value that defines whether or not Regional Geoblocking is active on this Delivery Service. The actual configuration of Regional Geoblocking is done in the Profile used by the Traffic Router serving the Delivery Service. Rules for this Delivery Service may exist, but they will not actually be used unless this field is true.

Tip: Regional Geoblocking is configured primarily with respect to Canadian postal codes, so unless specifically Canadian regions should be allowed/disallowed to access content, Geo Limit
is probably a better setting for controlling access to content according to geographic location.

Routing Name

A DNS label in the Delivery Service’s domain that forms the FQDN that is used by clients to request content. All together, the constructed FQDN looks like: Delivery Service Routing Name.Delivery Service xml_id.CDN Subdomain.CDN Domain.Top-Level Domain.

Servers

Servers can be assigned to Delivery Services using the Servers and Delivery Service Traffic Portal sections, or by directly using the deliveryserviceserver endpoint. Only Edge-tier cache servers can be assigned to a Delivery Service, and once they are so assigned they will begin to serve content for the Delivery Service (after updates are queued and then applied). Any servers assigned to a Delivery Service must also belong to the same CDN as the Delivery Service itself. At least one server must be assigned to a Delivery Service in order for it to serve any content.

Signing Algorithm

URLs/URIs may be signed using one of two algorithms before a request for the content to which they refer is sent to the origin (which in practice can be any upstream network). At the time of this writing, this field is restricted within the Traffic Ops Database to one of two values (or NULL/“None”, to indicate no signing should be done).

See also:
The url_sig README.

url_sig URL signing will be implemented in this Delivery Service using the url_sig Apache Traffic Server plugin. (Aliased as “URL Signature Keys” in Traffic Portal forms)

uri_signing URL signing will be implemented in this Delivery Service using an algorithm based on a work-in-progress RFC specification draft. (Aliased as “URI Signing Keys” in Traffic Portal forms)

Table 19: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed</td>
<td>In all components prior to Traffic Control v2.2. Some endpoints in early versions of the Traffic Ops API will still return this field instead of “signingAlgorithm”.</td>
<td>A boolean value where true was the same as “url_sig” in current versions, and false indicated URL signing would not be done for the Delivery Service.</td>
</tr>
</tbody>
</table>

1 Some things to consider when choosing an xml_id and routing name: the name should be descriptive and unique, but as brief as possible to avoid creating a monstrous FQDN. Also, because these are combined to form an FQDN, they should not contain any characters that are illegal for a DNS subdomain, e.g. . (period/dot). Finally, the restrictions on what characters are allowable (especially in xml_id) are, in general, NOT enforced by the Traffic Ops API, so take care that the name is appropriate. See RFC 1035 for exact guidelines.
Keys for either algorithm can be generated within *Traffic Portal*.

### Static DNS Entries

Static DNS Entries can be added *under* a Delivery Service’s domain. These DNS records can be configured in the *Delivery Service* section of Traffic Portal, and can be any valid CNAME, A or AAAA DNS record - provided the associated hostname falls within the DNS domain for the Delivery Service. For example, a Delivery Service with `xml_id` “demo1” and belonging to a CDN with domain “mycdn.ciab.test” could have Static DNS Entries for hostnames “foo.demo1.mycdn.ciab.test” or “foo.bar.demo1.mycdn.ciab.test” but not “foo.bar.mycdn.ciab.test” or “foo.bar.test”.

**Note:** The *Routing Name* of a Delivery Service is not part of the SOA (Start of Authority) record for the Delivery Service’s domain, and so there is no need to place Static DNS Entries below a domain containing it.

### Tenant

The *Tenant* who owns this Delivery Service. They (and their parents, if any) are the only ones allowed to make changes to this Delivery Service. Typically, `tenant/Tenant` refers to the *name* of the owning *Tenant*, but occasionally (most notably in the payloads and/or query parameters of certain *Traffic Ops API* requests) it actually refers to the *integral, unique identifier* of said *Tenant*.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TenantID</td>
<td>Go code and <em>Traffic Ops API</em> requests/responses</td>
<td>Integral, unique identifier (bigint, int etc.)</td>
</tr>
</tbody>
</table>

### Traffic Router Additional Response Headers

List of HTTP header `{{name}}:{{value}}` pairs separated by `__RETURN__` or simply on separate lines. Listed pairs will be included in all HTTP responses from Traffic Router for HTTP-*Routed* Delivery Services.

Deprecated since version 4.0: The use of `__RETURN__` as a substitute for a real newline is unnecessary and the ability to do so will be removed in the future.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>trResponseHeaders</td>
<td>Traffic Control source code and Delivery Service objects returned by the <em>Traffic Ops API</em></td>
<td>unchanged (string etc.)</td>
</tr>
</tbody>
</table>
**Traffic Router Log Request Headers**

List of HTTP header names separated by \__RETURN\__ or simply on separate lines. Listed pairs will be logged for all HTTP requests to Traffic Router for HTTP-Routed Delivery Services.

Deprecated since version 4.0: The use of \__RETURN\__ as a substitute for a real newline is unnecessary and the ability to do so will be removed in the future.

### Table 22: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>trRequest-Headers</td>
<td>Traffic Control source code and Delivery Service objects returned by the Traffic Ops API</td>
<td>unchanged (string etc.)</td>
</tr>
</tbody>
</table>

**Type**

Defines the content routing method used by the Delivery Service. In most cases this is an integral, unique identifier that corresponds to an enumeration of the Delivery Service Types. In other cases, this the actual name of said type.

The “Type” of a Delivery Service can mean several things. First, it can be used to refer to the “routing type” of Delivery Service. This is one of:

**Tip:** The only way to get the integral, unique identifier of a Type of Delivery Service is to look at the database after it has been generated; these are non-deterministic and cannot be guaranteed to have any particular value, or even consistent values. This can be done directly or, preferably, using the types endpoint. Unfortunately, knowing the name of the Type is rarely enough for many applications. The useInColumn values of these Types will be deliveryservice.

**DNS** Delivery Services of this routing type are routed by Traffic Router by providing DNS records that provide the IP addresses of cache servers when clients look up the full Delivery Service FQDN.

**HTTP** The Traffic Router(s) responsible for routing this Delivery Service will still answer DNS requests for the Delivery Service FQDN, but will provide its own IP address. The client then directs its HTTP request to the Traffic Router, which will use an HTTP redirection response to direct the client to a cache server.

More generally, though, Delivery Services have a Type that defines not only how traffic is routed, but also how content is cached and semantically defines what “content” means in the context of a given Delivery Service.

**ANY_MAP** This is a special kind of Delivery Service that should only be used when control over the clients is guaranteed, and very fine control over the ATS remap.config line for this Delivery Service is required. ANY_MAP is not known to Traffic Router. It is not routed in any way. For Delivery Services of this type, the “Raw Remap Text” field must be defined, as it is the only configuration generated by Traffic Control. The only way for
a client to utilize delivery through an ANY_MAP service is by knowing in advance the IP address of one or more *Edge-tier cache servers* and make the appropriate request(s).

**DNS**  Uses DNS content routing. Delivers content normally. This is the recommended Type for delivering smaller objects like web page images.

**DNS_LIVE**\(^3\) Uses DNS Content routing, but optimizes caching for live video streaming. Specifically, the configuration generated for *cache servers* responsible for serving content for this Delivery Service will not cache that content on storage disks. Instead, they will make use of RAM block devices dedicated to ATS - as specified by the special `RAM_Drive_Prefix` and `RAM_Drive_Letters` Parameters. Also, any *Mid-tier* of caching is bypassed.

**DNS_LIVE_NATNL** Works exactly the same as DNS_LIVE, but is optimized for delivery of live video content across a wide physical area. What this means is that the *Mid-tier* of caching is *not* bypassed, unlike DNS_LIVE. The *Mid-tier* will also use block RAM devices.

**HTTP**  Uses HTTP content routing, delivers content normally. This is the recommended Type for delivering larger objects like video streams.

**HTTP_LIVE**\(^3\) Uses HTTP Content routing, but optimizes caching for live video streaming. Specifically, the configuration generated for *cache servers* responsible for serving content for this Delivery Service will not cache that content on storage disks. Instead, they will make use of RAM block devices dedicated to ATS - as specified by the special `RAM_Drive_Prefix` and `RAM_Drive_Letters` Parameters. Also, any *Mid-tier* of caching is bypassed.

**HTTP_LIVE_NATNL** Works exactly the same as HTTP_LIVE, but is optimized for delivery of live video content across a wide physical area. What this means is that the *Mid-tier* of caching is *not* bypassed, unlike HTTP_LIVE. The *Mid-tier* will also use block RAM devices.

**HTTP_NO_CACHE**\(^3\) Uses HTTP Content Routing, but *cache servers* will not actually cache the delivered content - they act as just proxies. This will bypass any existing *Mid-tier* entirely (as it’s totally useless when content is not being cached).

**STEERING**  This is a sort of “meta” Delivery Service. It is used for directing clients to one of a set of Delivery Services, rather than delivering content directly itself. The Delivery Services to which a STEERING Delivery Service routes clients are referred to as “targets”. Targets in general have an associated “value” and can be of several *Types* that define the meaning of the value - these being:

**STEERING_ORDER** The value of a STEERING_ORDER target sets a strict order of preference. In cases where a response to a client contains multiple Delivery Services, those targets with a lower “value” appear earlier than those with a higher “value”. In cases where two or more targets share the same value, they each have an equal chance of being presented to the client - effectively spreading traffic evenly across them.

---

\(^3\) These Delivery Services Types are vulnerable to what this writer likes to call the “Duplicate Origin Problem”. This problem is tracked by issue #3537.
STEERING_WEIGHT  The values of STEERING_WEIGHT targets are interpreted as “weights”, which define how likely it is that any given client will be routed to a specific Delivery Service - effectively this determines the spread of traffic across each target.

The targets of a Delivery Service may be set using the appropriate section of Traffic Portal or via the steering/{{ID}}/targets and steering/{{ID}}/targets/{{targetID}} Traffic Ops API endpoints.

See also:

For more information on setting up a STEERING (or CLIENT STEERING) Delivery Service, see Configure Delivery Service Steering.

See also:

For implementation details about how Traffic Router routes STEERING (and CLIENT_STEERING) Delivery Services, see Steering Feature.

CLIENT_STEERING  A CLIENT_STEERING Delivery Service is exactly like STEERING except that it provides clients with methods of bypassing the weights, orders, and localizations of targets in order to choose any arbitrary target at will. When utilizing these methods, the client will either directly choose a target immediately or request a list of all available targets from Traffic Router and then choose one to which to send a subsequent request for actual content. CLIENT_STEERING also supports two additional target types:

STEERING_GEO_ORDER  These targets behave exactly like STEERING_ORDER targets, but Delivery Services are grouped according to the “locations” of their origins. Before choosing a Delivery Service to which to direct the client, Traffic Router will first create subsets of choices according to these groupings, and order them by physical distance from the client (closest to farthest). Within these subsets, the values of the targets establish a strict precedence ordering, just like STEERING_ORDER targets.

STEERING_GEO_WEIGHT  These targets behave exactly like STEERING_WEIGHT targets, but Delivery Services are grouped according to the “locations” of their origins. Before choosing a Delivery Service to which to direct the client, Traffic Router will first create subsets of choices according to these groupings, and order them by physical distance from the client (closest to farthest). Within these subsets, the values of the targets establish the likelihood that any given target within the subset will be chosen for the client - effectively determining the spread of traffic across targets within that subset.

Important: To make use of the STEERING_GEO_ORDER and/or STEERING_GEO_WEIGHT target types, it is first necessary to ensure that at least the “primary” origin of the Delivery Service has an associated geographic coordinate pair. This can be done either from the Origins page in Traffic Portal, or using the origins Traffic Ops API endpoint.
Note: “Steering” is also commonly used to collectively refer to either of the kinds of Delivery Services that can participate in steering behavior (STEERING and CLIENT_STEERING).

### Table 23: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Routing Type</td>
<td>Traffic Portal forms</td>
<td>The name of any of the Delivery Service Types (string)</td>
</tr>
<tr>
<td>TypeID</td>
<td>In Go code and Traffic Ops API requests/responses</td>
<td>Integral, unique identifier (bigint, int etc.)</td>
</tr>
</tbody>
</table>

### Use Multi-Site Origin Feature

A boolean value that indicates whether or not this Delivery Service serves content for an origin that provides content from two or more redundant servers. There are very few good reasons for this to not be false. When true, Traffic Ops will configure Mid-tier cache servers to perform load-balancing and other optimizations for redundant origin servers.

Naturally, this assumes that each redundant server is exactly identical, from request paths to actual content. If Multi-Site Origin is configured for servers that are not identical, the client’s experience is undefined. Furthermore, the origin servers may have differing IP addresses, but must serve content for a single FQDN - as defined by the Delivery Service’s Origin Server Base URL. These redundant servers must be configured as servers (server Type ORG) in Traffic Ops - either using the appropriate section of Traffic Portal or the servers endpoint.

**Important:** In order for a given Mid-tier cache server to support Multi-Site Origins, the value of a Parameter named http.parent_proxy_routing_enable in configuration file records.config must be set to 1 on that server’s Profile. If using an optional secondary grouping of Multi-Site Origins, the Parameter named url_remap.remap_required in configuration file records.config must also be set to 1 on that Profile. These settings must be applied to all Mid-tier cache servers’ that are the parents of any Edge-tier cache server assigned to this Delivery Service.

**See also:**

These parameters are described in the ATS documentation sections for Parent Proxy Configuration and URL Remap Rules, respectively.

### Table 24: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiSiteOrigin</td>
<td>In Go code and Traffic Ops API requests/responses</td>
<td>unchanged (bool, boolean etc.)</td>
</tr>
<tr>
<td>MSO (Multi-Site Origin)</td>
<td>In documentation and used heavily in discussion in Slack, mailing list etc.</td>
<td>unchanged (usually only used where implicitly true)</td>
</tr>
</tbody>
</table>
A Delivery Service Profile can have Parameters that affect Multi-Site Origin configuration. These are detailed in the Parameters of a Delivery Service Profile that Affect MSO Configuration table. All of these Parameters should have their Configuration File set to parent.config. Each Parameter directly corresponds to a field in a line of the ATS parent.config file <https://docs.trafficserver.apache.org/en/7.1.x/admin-guide/files/parent.config.en.html> (usually by almost the same name), and documentation for these fields is provided in the form of links to their entries in the ATS documentation.

Table 25: Parameters of a Delivery Service Profile that Affect MSO Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>ATS parent.config field</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>mso.algorithm</td>
<td>round_robin</td>
<td>Sets the algorithm used to determine from which origin server content will be requested.</td>
</tr>
<tr>
<td>mso.max_simple_retries</td>
<td>max_simple_retries</td>
<td>Sets a strict limit on the number of “simple retries” allowed before giving up</td>
</tr>
<tr>
<td>mso.max_unavailable_server_retries</td>
<td>max_unavailable_server_retries</td>
<td>Sets a strict limit on the number of times the cache server will attempt to request content from an origin server that has previously been considered “unavailable”.</td>
</tr>
<tr>
<td>mso.parent_retry</td>
<td>parent_retry</td>
<td>Sets whether the cache servers will use “simple retries”, “unavailable server retries”, or both.</td>
</tr>
<tr>
<td>mso.simple_retry_response_codes</td>
<td>UNKNOWN</td>
<td>UNKOWN - supposedly defines HTTP response codes from an origin server that necessitate a “simple retry”.</td>
</tr>
<tr>
<td>mso.unavailable_server_retry_response_codes</td>
<td>UNKOWN</td>
<td>UNKOWN - supposedly defines HTTP response codes from an origin server that indicate it is currently “unavailable”.</td>
</tr>
</tbody>
</table>

**Warning:** The mso.simple_retry_response_codes Parameter has no apparent, possible use according to the ATS parent.config documentation. Whether or not it has any effect - let alone the intended effect - is not known, and its use is therefore strongly discouraged.

See also:

A quick guide on setting up Multi-Site Origins is given in Configure Multi-Site Origin.

See also:

See the Apache Traffic Server documentation for more information on its implementation of Multi-Site Origins.

xml_id

A text-based unique identifier for a Delivery Service. Many Traffic Ops API endpoints and internal ATC functions use this to uniquely identify a Delivery Service as opposed to the historically favored “ID”. This string will become a part of the CDN service domain, which
all together looks like: *Delivery Service Routing Name*.Delivery Service xml_id.CDN Subdomain.CDN Domain.Top-Level Domain. Must be all lowercase, no spaces or special characters, but may contain dashes/hyphens.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Traffic Portal tables and forms</td>
<td>unchanged (string)</td>
</tr>
</tbody>
</table>
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.

When installing a complete CDN from scratch, a sample recommended order is:

1. Traffic Ops
2. Traffic Vault (Riak)
3. Traffic Portal
4. Traffic Monitor
5. Apache Traffic Server Mid-Tier Caches
6. Apache Traffic Server Edge-Tier Caches
7. Traffic Router
8. Traffic Stats

Once everything is installed, you will need to configure the servers to talk to each other. You will also need Origin server(s), from which the Mid-Tier Cache(s) will obtain content. An Origin server is simply an HTTP(S) server which serves the content you wish to cache on the CDN.
3.1.1 Traffic Ops - Configuring

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

Installing the SSL Certificate

By default, Traffic Ops runs as an SSL web server (that is, over HTTPS), and a certificate needs to be installed.

Self-signed Certificate (Development)

# 9: Example Procedure

```
$ openssl genrsa -des3 -passout pass:x -out localhost.pass.key 2048
Generating RSA private key, 2048 bit long modulus 
...
$ openssl rsa -passin pass:x -in localhost.pass.key -out localhost.key
writing RSA key 
$ rm localhost.pass.key

$ openssl req -new -key localhost.key -out localhost.csr
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<enter>
State or Province Name (full name) []:CO<enter>
Locality Name (eg, city) [Default City]:Denver<enter>
Organization Name (eg, company) [Default Company Ltd]: <enter>
Organizational Unit Name (eg, section) []: <enter>
Common Name (eg, your name or your server's hostname) []: <enter>
Email Address []: <enter>

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []: pass<enter>
An optional company name []: <enter>
$ openssl x509 -req -sha256 -days 365 -in localhost.csr -signkey localhost.key -out localhost.crt
Signature ok
subject=/C=US/ST=CO/L=Denver/O=Default Company Ltd
Getting Private key
```

(continues on next page)
Certificate from Certificate Authority (Production)

Note: You will need to know the appropriate answers when generating the certificate request file `trafficopss.csr` below.

Example Procedure

# 10: Example Procedure

```bash
$ openssl genrsa -des3 -passout pass:x -out trafficops.pass.key 2048
Generating RSA private key, 2048 bit long modulus...
$ openssl rsa -passin pass:x -in trafficops.pass.key -out trafficops.key
writing RSA key
$ rm localhost.pass.key
```

Generate the CSR file needed for CA (Certificate Authority) request

# 11: Example Certificate Signing Request File Generation

```bash
$ openssl req -new -key trafficops.key -out trafficops.csr
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]: <enter country code>
State or Province Name (full name) []: <enter state or province>
Locality Name (eg, city) [Default City]: <enter locality name>
Organization Name (eg, company) [Default Company Ltd]: <enter organization name>
Organizational Unit Name (eg, section) []: <enter organizational unit name>
Common Name (eg, your name or your server's hostname) []: <enter server's hostname name>
```
Email Address []: <enter e-mail address>

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

$ sudo cp trafficops.key /etc/pki/tls/private
$ sudo chown trafops:trafops /etc/pki/tls/private/trafficops.key

You must then take the output file trafficops.csr and submit a request to your CA. Once you get approved and receive your trafficops.crt file

# 12: Certificate Installation

sudo cp trafficops.crt /etc/pki/tls/certs
sudo chown trafops:trafops /etc/pki/tls/certs/trafficops.crt

If necessary, install the CA (CERTIFICATE AUTHORITY) CERTIFICATE’s .pem‘ and .crt files in /etc/pki/tls/certs.

You will need to update the file /opt/traffic_ops/app/conf/cdn.conf with the any necessary changes.

# 13: Sample ‘listen’ Line When Path to trafficops.crt and trafficops.key are Known

'hypnotoad' => ...
'listen' => 'https://[::]:443?cert=/etc/pki/tls/certs/trafficops.crt&key=/etc/pki/tls/private/trafficops.key&ca=/etc/pki/tls/certs/localhost.ca&verify=0x00&ciphers=AES128-GCM-SHA256:HIGH:!RC4:!MD5:!aNULL:!EDH:!ED'
...

Content Delivery Networks

Profile Parameters

Many of the settings for the different servers in a Traffic Control CDN are controlled by parameters in the Configure → Parameters view of Traffic Portal. Parameters are grouped in profiles and profiles are assigned to a server or a Delivery Service. For a typical cache there are hundreds of configuration settings to apply. The Traffic Portal Parameters view contains the defined settings. To make life easier, Traffic Portal 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 explained in the Global Profile Parameters table.
Table 27: Global Profile Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Config-File</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tm.url</td>
<td>global</td>
<td>The URL at which this Traffic Ops instance services requests</td>
</tr>
<tr>
<td>tm.rev_proxy.url</td>
<td>global</td>
<td>Not required. The URL where a caching proxy for configuration files generated by Traffic Ops may be found. Requires a minimum ORT version of 2.1. When configured, ORT will request configuration files via this FQDN, which should be set up as a reverse proxy to the Traffic Ops server(s). The suggested cache lifetime for these files is 3 minutes or less. This setting allows for greater scalability of a CDN maintained by Traffic Ops by caching configuration files of profile and CDN scope, as generating these is a very computationally expensive process.</td>
</tr>
<tr>
<td>tm.toolname</td>
<td>global</td>
<td>The name of the Traffic Ops tool. Usually “Traffic Ops” - this will appear in the comment headers of the generated files</td>
</tr>
<tr>
<td>tm.infourl</td>
<td>global</td>
<td>This is the “for more information go here” URL, which used to be visible in the “About” page of the now-deprecated Traffic Ops UI</td>
</tr>
<tr>
<td>tm.logo</td>
<td>global</td>
<td>This is the URL of the logo for Traffic Ops and can be relative if the logo is under traffic_ops/app/public</td>
</tr>
<tr>
<td>tm.instance_name</td>
<td>global</td>
<td>The name of the Traffic Ops instance - typically to distinguish instances when multiple are active</td>
</tr>
<tr>
<td>tm.traffic_mon_fwd_proxy</td>
<td>global</td>
<td>When collecting stats from Traffic Monitor, Traffic Ops will use this forward proxy instead of the actual Traffic Monitor host. 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.</td>
</tr>
<tr>
<td>geolocation.polling.url</td>
<td>Config.json</td>
<td>The location of a geographic IP mapping database for Traffic Router instances to use</td>
</tr>
<tr>
<td>geolocation6.polling.url</td>
<td>Config.json</td>
<td>The location of a geographic IPv6 mapping database for Traffic Router instances to use</td>
</tr>
<tr>
<td>maxmind.default.override</td>
<td>Config.json</td>
<td>The destination geographic coordinates to use for client location when the geographic IP mapping database returns a default location that matches the country code. This parameter can be specified multiple times with different values to support default overrides for multiple countries. The reason for the name “maxmind” is because MaxMind’s GeoIP2 database is the default geographic IP mapping database implementation used by Comcast production servers (and the only officially supported implementation at the time of this writing). The format of this Parameter’s value is: &lt;Country Code&gt;;&lt;Latitude&gt;,&lt;Longitude&gt;, e.g. US;37.751,-97.822</td>
</tr>
</tbody>
</table>

These parameters should be set to reflect the local environment.

After running the **postinstall** script, Traffic Ops has the Default Profiles pre-loaded.
Table 28: Default Profiles

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGE1</td>
<td>The profile to be applied to the latest supported version of ATS, when running as an Edge-tier cache</td>
</tr>
<tr>
<td>TR1</td>
<td>The profile to be applied to the latest version of Traffic Router</td>
</tr>
<tr>
<td>TM1</td>
<td>The profile to be applied to the latest version of Traffic Monitor</td>
</tr>
<tr>
<td>MID1</td>
<td>The profile to be applied to the latest supported version of ATS, when running as a Mid-tier cache</td>
</tr>
<tr>
<td>RIAK_ALL</td>
<td>&quot;Ria&quot; profile for all CDNs to be applied to the Traffic Vault servers (&quot;Ria&quot; being the name of the underlying database used by Traffic Vault)</td>
</tr>
</tbody>
</table>

**Note:** The “EDGE1” and “MID1” 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. The Cache Server Hardware Parameters table tabulates the cache parameters that are likely to need changes from the default profiles shipped with Traffic Ops.
Table 29: Cache Server Hardware Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Config-File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow_ip</td>
<td>astats.config</td>
<td>This is a comma-separated list of IPv4 CIDR (Classless Inter-Domain Routing) blocks that will have access to the ‘astats’ statistics on the cache servers. The Traffic Monitor IP addresses have to be included in this if they are using IPv4 to monitor the cache servers</td>
</tr>
<tr>
<td>allow_ip6</td>
<td>astats.config</td>
<td>This is a comma-separated list of IPv6 CIDR blocks that will have access to the ‘astats’ statistics on the cache servers. The Traffic Monitor IP addresses have to be included in this if they are using IPv6 to monitor the cache servers</td>
</tr>
<tr>
<td>Drive_Prefix</td>
<td>storage.config</td>
<td>The device path start of the disks. For example, if storage devices /dev/sda through /dev/sdf are to be used for caching, this should be set to /dev/sd</td>
</tr>
<tr>
<td>Drive_Letters</td>
<td>storage.config</td>
<td>A comma-separated list of the letter part of the storage devices to be used for caching. For example, if storage devices /dev/sda through /dev/sdf are to be used for caching, this should be set to a, b, c, d, e, f</td>
</tr>
<tr>
<td>purge_allow_ip</td>
<td>ip_allow.config</td>
<td>The IP address range that is allowed to execute the PURGE method on the caches (not related to Invalidate Content)</td>
</tr>
<tr>
<td>coalesce_masklen_v4</td>
<td>ip_allow.config</td>
<td>The mask length to use when coalescing IPv4 networks into one line using the NetAddr::IP Perl library</td>
</tr>
<tr>
<td>coalesce_number_v4</td>
<td>ip_allow.config</td>
<td>The number to use when coalescing IPv4 networks into one line using the NetAddr::IP Perl library</td>
</tr>
<tr>
<td>coalesce_masklen_v6</td>
<td>ip_allow.config</td>
<td>The mask length to use when coalescing IPv6 networks into one line using the NetAddr::IP Perl library</td>
</tr>
<tr>
<td>health.threshold.loadavg</td>
<td>ras-cadal.properties</td>
<td>The Unix ‘load average’ (as given by uptime(1)) at which Traffic Router will stop sending traffic to this cache</td>
</tr>
<tr>
<td>health.threshold.availableBandwidthInKbps</td>
<td>ras-cadal.properties</td>
<td>The bandwidth (in kilobits per second) that Traffic Router will try to keep available on the cache. For example “&gt;1500000” means “stop sending new traffic to this cache server when traffic is at 8.5Gbps on a 10Gbps interface”</td>
</tr>
</tbody>
</table>

The Plugin Parameters table contains all Traffic Server plug-ins that must be configured as global parameters.
### Table 30: Plugin Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Config-File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>astats_over_http</td>
<td>package</td>
<td>The package version for the ATS <code>astats_over_http</code> plugin</td>
</tr>
<tr>
<td>traffic-server</td>
<td>package</td>
<td>The package version of ATS</td>
</tr>
<tr>
<td>regex_revalidate</td>
<td>plugin.config</td>
<td>The configuration to be used for the ATS <code>regex_revalidate</code> plugin</td>
</tr>
<tr>
<td>remap_stats</td>
<td>plugin.config</td>
<td>The configuration to be used for the ATS <code>remap_stats</code> plugin - value should be left blank</td>
</tr>
</tbody>
</table>

Cache server parameters for special configurations, which are unlikely to need changes but may be useful in particular circumstances, may be found in the *Special Parameters* table.

### Table 31: Special Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>ConfigFile</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>not_a_parent</td>
<td>parent.config</td>
<td>This is a boolean flag and is considered <strong>true</strong> if it exists and has a value. It prevents cache servers that have the affected cache server(s)’s <em>Cache Group</em> as a parent of their own. <em>Cache Group</em>. This is primarily useful for when Edge-tier cache servers are configured to have a <em>Cache Group</em> of other Edge-tier cache servers as parents (a highly unusual configuration), and it is necessary to exclude some - but not all - Edge-tier cache servers in the parent <em>Cache Group</em> from the parent <em>config</em> (for example because they lack necessary capabilities), but still have all Edge-tier cache servers in the same <em>Cache Group</em> in order to take traffic from ordinary <em>Delivery Services</em> at that <em>Cache Group</em>’s geographic location. Once again, this is a highly unusual scenario, and under ordinary circumstances this parameter should not exist.</td>
</tr>
</tbody>
</table>

### Regions, Locations and Cache Groups

All servers have to have a *Physical Location*, which defines their geographic latitude and longitude. Each *Physical Location* is part of a *Region*, and each *Region* is part of a *Division*. For example, Denver could be the name of a *Physical Location* in the Mile High *Region* and that *Region* could be part of the West *Division*. The hierarchy between these terms is illustrated graphically in the *Topography Hierarchy*.

To create these structures in Traffic Portal, first make at least one *Division* under *Topology* →
Fig. 1: Topography Hierarchy
Divisions. Next enter the desired Region(s) in Topology → Regions, referencing the earlier-entered Division(s). Finally, enter the desired Physical Location(s) in Topology → Phys Locations, referencing the earlier-entered Region(s).

All servers also have to be part of a Cache Group. A Cache Group is a logical grouping of cache servers, that don’t have to be in the same Physical Location (in fact, usually a Cache Group is spread across minimally two Physical Locations for redundancy purposes), but share geographical coordinates for content routing purposes.

Configuring Content Purge

Purging cached content using ATS is not simple; there is no file system from which to delete files and/or directories, and in large caches it can be hard to delete content matching a simple regular expression from the cache. This is why Traffic Control uses the Regex Revalidate Plugin to purge content from the cache. The cached content is not actually removed, instead a check that runs before each request on each cache server is serviced to see if this request matches a list of regular expressions. If it does, the cache server is forced to send the request upstream to its parents (possibly other caches, possibly the origin) without checking for the response in its cache. The Regex Revalidate Plugin will monitor its configuration file, and will pick up changes to it without needing to alert ATS (APACHE TRAFFIC SERVER). CHANGES TO THIS FILE NEED TO BE DISTRIBUTED TO THE HIGHEST TIER (MID-TIER) CACHE SERVERS IN THE CDN BEFORE THEY ARE DISTRIBUTED TO THE LOWER TIERS, TO PREVENT FILLING THE LOWER TIERS WITH THE CONTENT THAT SHOULD BE PURGED FROM THE HIGHER TIERS WITHOUT HITTING THE ORIGIN. THIS IS WHY THE :term:`ORT` script will - by default - push out configuration changes to Mid-tier cache servers first, confirm that they have all been updated, and then push out the changes to the lower tiers. In large CDNs, this can make the distribution and time to activation of the purge too long, and because of that there is the option to not distribute the regex_revalidate.config file using the ORT script, but to do this using other means. By default, Traffic Ops will use ORT to distribute the regex_revalidate.config file. Content Purge is controlled by the parameters in the profile of the cache server specified in the Content Purge Parameters table.
### Table 32: Content Purge Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Config-File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>regex_revalidate.config</td>
<td>Where in the file system the regex_revalidate.config file should be located on the cache server. The presence of this parameter tells ORT to distribute this file; delete this parameter from the profile if this file is distributed using other means.</td>
</tr>
<tr>
<td>maxRevalDurationDays</td>
<td>regex_revalidate.config</td>
<td>The configuration for which a purge shall be active. To prevent a buildup of many checks before each request, this is longest duration (in days) for which the system will allow content purges to remain active.</td>
</tr>
<tr>
<td>regex_plugin.config</td>
<td></td>
<td>The configuration to be used for the regex_revalidate plugin.</td>
</tr>
<tr>
<td>use_reval_pending</td>
<td></td>
<td>Configures Traffic Ops to use a separate reval_pending flag for each cache server. When this flag is in use, ORT will check for a new regex_revalidate.config every 60 seconds in “SYNCDS” mode during the dispersal timer. This will also allow ORT to be run in “REVALIDATE” mode, which will check for and clear the reval_pending flag. This can be set to run via cron(8) task. Enable with a value of 1.</td>
</tr>
</tbody>
</table>

New in version 2.1: use_reval_pending was unavailable prior to Traffic Ops version 2.1.

**Note:** The TTL entered by the administrator in the purge request should be longer than the TTL of the content to ensure the bad content will not be used. If the CDN is serving content of unknown, or unlimited TTL, the administrator should consider using proxy-config-http-cache-guaranteed-min-lifetime to limit the maximum time an object can be in the cache before it is considered stale, and set that to the same value as maxRevalDurationDays (Note that the former is in seconds and the latter is in days, so convert appropriately).

### Creating the CentOS Kickstart File

The Kickstart file is a text file, containing a list of items, each identified by a keyword. This file can be generated using the Red Hat Kickstart Configurator application, or it can be written from scratch. The Red Hat Enterprise Linux installation program also creates a sample Kickstart file based on the options selected during installation. It is written to the file /root/anaconda-ks.cfg in this case. This file is editable using most text editors.

### Generating a System Image

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.

# 14: Creating a New System Image Definition Tree from an Existing One

```bash
# Starting from the Kickstart root directory (`/var/www/files` by default)
mkdir newdir
cd newdir/

# In this example, the pre-existing system image definition tree is for CentOS 7.4 located in `centos74`
cp -r ../centos74/* .
vim ks.src
vim isolinux/isolinux.cfg
cd ..
vim osversions.cfg
```

*ks.src* is a standard, Kickstart-formatted file that will be used to create the Kickstart (ks.cfg) file for the install whenever a system image is generated from the source tree. *ks.src* is a template - it will be overwritten by any information set in the form submitted from *Tools → Generate ISO* in Traffic Portal. Ultimately, the two are combined to create the final Kickstart file (ks.cfg).

**Note:** It is highly recommended for ease of use that the system image source trees be kept under 1GB in size.

**See also:**
For in-depth instructions, please see *Kickstart Installation* in the Red Hat documentation.

**Configuring the Go Application**

Traffic Ops is in the process of migrating from Perl to Go, and currently runs as two applications. The Go application serves all endpoints which have been rewritten in the Go language, and transparently proxies all other requests to the old Perl application. Both applications are installed by the RPM, and both run as a single *systemd*(1) service. When the project has fully migrated to Go, the Perl application will be removed, and the RPM and service will consist solely of the Go application.

By default, the *postinstall* script configures the Go application to behave and transparently serve as the old Perl Traffic Ops did in previous versions. This includes reading the old *cdn.conf* and *database.conf* config files, and logging to the old *access.log* location. However, the Go Traffic Ops application may be customized by passing the command-line flag, `-oldcfg=false`. By default, it will then look for a configuration file at
The new Go Traffic Ops application as a systemd(1) service with a new configuration file, the -oldcfg=false and -cfg flags may be added to the start function in the service file, located by default at /etc/init.d/traffic_ops.

### 3.1.2 Managing Traffic Ops Extensions

Traffic Ops supports two types of extensions. **Check Extensions** are analytics scripts that collect and display information as columns in the table under Monitor → Cache Checks in Traffic Portal. **Data Source Extensions** provide ways to add data to the graph views and usage APIs.

#### Check Extensions

Check Extensions are scripts that, after registering with Traffic Ops, have a column reserved in the Monitor → Cache Checks view and usually run periodically using cron(8). Each extension is a separate executable located in $TO_HOME/bin/checks/ on the Traffic Ops server (though all of the default extensions are written in Perl, this is in no way a requirement; they can be any valid executable). The currently registered extensions can be listed by running /opt/traffic_ops/app/bin/extensions -a. Some extensions automatically registered with the Traffic Ops database (to_extension table) at install time (see traffic_ops/app/db/seeds.sql). However, cron(8) must still be configured to run these checks periodically. The extensions are called like so:

```
# 15: Example Check Extension Call

$TO_HOME/bin/checks/<name> -c "{"base_url": ",https://", 
<traffic_ops_ip>": ", "check_name": ", <check_name>"}" -l <log_level>
```

- **name** The basename of the extension executable
- **traffic_ops_ip** The IP address or FQDN of the Traffic Ops server
- **check_name** The name of the check e.g. CDU, CHR, DSCP, MTU, etc...
- **log_level** A whole number between 1 and 4 (inclusive), with 4 being the most verbose. Implementation of this field is optional

It is the responsibility of the check extension script to iterate over the servers it wants to check and post the results. An example script might proceed by logging into the Traffic Ops server using the HTTPS base_url provided on the command line. The script is hard-coded with an authentication token that is also provisioned in the Traffic Ops User database. This token allows the script to obtain a cookie used in later communications with the Traffic Ops API. The script then obtains a list of all cache servers to be polled by accessing servers. This list is then iterated, running a command to gather the stats from each server. For some extensions, an HTTP GET request might be made to the ATS astats plugin, while for others the server might be pinged, or a command might run over ssh(1). The results are
then compiled into a numeric or boolean result and the script submits a POST request containing the result back to Traffic Ops using `servercheck`. A check extension can have a column of ‘✓’s and ‘✗’s (CHECK_EXTENSION_BOOL) or a column that shows a number (CHECK_EXTENSION_NUM).

**Check Extensions Installed by Default**

**CDU (Cache Disk Usage)** This check shows how much of the available total cache disk is in use. A “warm” cache should show 100.00.

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

**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)

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

**ORT (Operational Readiness Test)** The ORT column shows how many changes the *ORT* script would apply if it was run. The number in this column should be 0 for caches that do not have updates pending.

**10G** Is the `ip_address` (the main IPv4 address) from the server table ping-able?

**ILO (Integrated Lights-Out)** Is the `ilo_ip_address` (the lights-out-management IPv4 address) from the server table ping-able?

**10G6** Is the `ip6_address` (the main IPv6 address) from the server table ping-able?

**FQDN** Is the FQDN (the concatenation of `host_name` and `domain_name` from the server table) ping-able?

**RTR (Responds to Traffic Router)** Checks the state of each cache as perceived by all Traffic Monitors (via Traffic Router). This extension asks each Traffic Router for the state of the cache. A check failure is indicated if one or more monitors report an error for a cache. A cache is only marked as good if all reports are positive. (This is a pessimistic approach, opposite of how TM marks a cache as up, “the optimistic approach”)

**Data Source Extensions**

Data Source Extensions work in much the same way as Check Extensions, but are implemented differently. Rather than being a totally external executable, a Data Source Extension must be written in Perl 5, as they are injected via manipulation of the `$PERL5LIB` environment variable. These extensions are not very well-documented (as you may be able to tell), and support for extending them may be phased out in future releases.
Example Cron File

The `cron(8)` file should be edited by running `crontab(1)` as the `traffops` user, or with `sudo(8)`. You may need to adjust the path to your `$TO_HOME` to match your system.

```bash
PERL5LIB=/opt/traffic_ops/app/local/lib/perl5:/opt/traffic_ops/app/

# IPv4 ping examples - The 'select: ["hostName","domainName"]' works but, if you want to check DNS resolution use FQDN.
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "10G\", \"select\": [\"hostName\",\"domainName\"]\"" >> /var/log/traffic_ops/extensionCheck.log 2>&1
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "10G\", \"select\": \"ipAddress\"\"" >> /var/log/traffic_ops/extensionCheck.log 2>&1
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "10G\", \"name\": \"IPv4 Ping\", \"select\": \"ipAddress\", \"syslog_facility\": \"local0\"\"" > /dev/null 2>&1

# IPv6 ping examples
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "10G6\", \"name\": \"IPv6 Ping\", \"select\": \"ip6Address\", \"syslog_facility\": \"local0\"\"" > /dev/null 2>&1
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "10G6\", \"select\": \"ip6Address\"\"" >> /var/log/traffic_ops/extensionCheck.log 2>&1

# iLO ping
18 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "ILO\", \"select\": \"iloIpAddress\"\"" >> /var/log/traffic_ops/extensionCheck.log 2>&1
18 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "ILO\", \"name\": \"ILO ping\", \"select\": \"iloIpAddress\", \"syslog_facility\": \"local0\"\"" >/dev/null 2>&1

# MTU ping
45 0 * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "\"base_url\": "https://localhost\", \"check_name\": "MTU\", \"select\": \"ipAddress\"\"" >> /var/log/traffic_ops/extensionCheck.log 2>&1
```

(continues on next page)
Apache Traffic Control Documentation, Release 3.0.0

(continued from previous page)

45 0 * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"MTU\", \n  \"select\": \"ip6Address\""> /var/log/traffic_ops/
  extensionCheck.log 2>&1
45 0 * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"MTU\", \n  \"name\": \"Max Trans Unit\", \"select\": \"ipAddress\", \n  \"syslog_facility\": \"local0\""> /dev/null 2>&1
45 0 * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"MTU\", \n  \"name\": \"Max Trans Unit\", \"select\": \"ip6Address\", \n  \"syslog_facility\": \"local0\""> /dev/null 2>&1

# FQDN
27 * * * * root /opt/traffic_ops/app/bin/checks/ToFQDNCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"FQDN\""> /var/log/traffic_ops/
  extensionCheck.log 2>&1
27 * * * * root /opt/traffic_ops/app/bin/checks/ToFQDNCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"FQDN\", \n  \"name\": \"DNS Lookup\", \n  \"syslog_facility\": \"local0\""> /dev/null 2>&1

# DSCP
36 * * * * root /opt/traffic_ops/app/bin/checks/ToDSCPCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"DSCP\", \n  \"cms_interface\": \"eth0\""> /var/log/traffic_ops/
  extensionCheck.log 2>&1
36 * * * * root /opt/traffic_ops/app/bin/checks/ToDSCPCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"DSCP\", \n  \"name\": \"Delivery Service\", \n  \"cms_interface\": \"eth0\", \n  \"syslog_facility\": \"local0\""> /dev/null 2>&1

# RTR
10 * * * * root /opt/traffic_ops/app/bin/checks/ToRTRCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"RTR\""> /var/log/traffic_ops/
  extensionCheck.log 2>&1
10 * * * * root /opt/traffic_ops/app/bin/checks/ToRTRCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"RTR\", \n  \"name\": \"Content Router Check\", \n  \"syslog_facility\": \"local0\""> /dev/null 2>&1

# CHR
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToCHRCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"CHR\""> /var/log/traffic_ops/
  extensionCheck.log 2>&1

# CDU
20 * * * * root /opt/traffic_ops/app/bin/checks/ToCDUCheck.pl -c "
  \"base_url\": "https://localhost", \"check_name\": \"CDU\""> /var/log/traffic_ops/
  extensionCheck.log 2>&1

(continues on next page)
3.1.3 Traffic Ops - Installing

System Requirements

The user must have the following for a successful minimal install:

- CentOS 7+
- Two machines - physical or virtual -, each with at least two (v)CPUs, 4GB of RAM, and 20 GB of disk space.
- Access to CentOS Base and EPEL repositories
- Access to The Comprehensive Perl Archive Network (CPAN)

Installation

1. Install PostgreSQL Database. For a production install it is best to install PostgreSQL on its own server/VM.

   See also:

   For more information on installing PostgreSQL, see their documentation.

   # 17: Example PostgreSQL Install Procedure

   ```
   yum update -y
   yum install -y https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-centos96-9.6-3.noarch.rpm
   yum install -y postgresql96-server
   su - postgres -c '/usr/pgsql-9.6/bin/initdb -A md5 -W'
   # forces the user to provide a superuser (postgres) password
   ```

2. Edit `/var/lib/pgsql/9.6/data/pg_hba.conf` to allow the Traffic Ops instance to access the PostgreSQL server. For example, if the IP address of the machine to be used as the Traffic Ops host is 99.33.99.1 add the line `host all all 99.33.99.1/32 md5` to the appropriate section of this file.
3. Edit the /var/lib/pgsql/9.6/data/postgresql.conf file to add the appropriate listen_addresses or listen_addresses = '*', set timezone = 'UTC', and start the database

```
# 18: Starting PostgreSQL with systemctl(1)
systemctl enable postgresql-9.6
systemctl start postgresql-9.6
systemctl status postgresql-9.6  # Prints the status of the PostgreSQL service, to prove it's running
```

4. Build a traffic_ops-version string.rpm file using the instructions under the Building Traffic Control page.

5. Install a PostgreSQL client on the Traffic Ops host

```
# 19: Installing PostgreSQL Client from a Hosted Source
yum install -y https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-centos96-9.6-3.noarch.rpm
```

6. Install the Traffic Ops RPM. The Traffic Ops RPM file should have been built in an earlier step.

```
# 20: Installing a Generated Traffic Ops RPM
yum install -y ./dist/traffic_ops-3.0.0-xxxx.yyyyyyy.el7.x86_64.rpm
```

**Note:** This will install the PostgreSQL client, psql as a dependency.

7. Login to the Database from the Traffic Ops machine. At this point you should be able to login from the Traffic Ops (hostname to in the example) host to the PostgreSQL (hostname pg in the example) host

```
# 21: Example Login to Traffic Ops Database from Traffic Ops Server
to-$ psql -h pg -U postgres
Password for user postgres:
psql (9.6.3)
Type "help" for help.
postgres=#
```

8. Create the user and database. By default, Traffic Ops will expect to connect as the traffic_ops user to the traffic_ops database.
# 22: Creating the Traffic Ops User and Database

```
psql -U postgres -h 99.33.99.1 -c "CREATE USER traffic_ops WITH ENCRYPTED PASSWORD 'tcr0cks';"
```

Password for user postgres:

```
CREATE ROLE
```

```
createdb traffic_ops --owner traffic_ops -U postgres -h 99.33.99.1
```

Password:

```
to-
```

9. Now, run the following command as the root user (or with sudo): `/opt/traffic_ops/install/bin/postinstall`. The `postinstall` script will first get all required Perl packages from CPAN (The Comprehensive Perl Archive Network). This may take a while, expect up to 30 minutes on the first install. 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 Complete! Modules were installed into `/opt/traffic_ops/app/local`. Some additional files will be installed, and then it will proceed with the next phase of the install, where it will ask you about the local environment for your CDN. Please make sure you remember all your answers and verify that the database answers match the information previously used to create the database.

# 23: Example Output

```
/opt/traffic_ops/install/bin/postinstall
...

==========/opt/traffic_ops/app/conf/production/
DATABASE_CONF==========
Database type [Pg]:
Database type: Pg
Database name [traffic_ops]:
Database name: traffic_ops
Database server hostname IP or FQDN [localhost]: 99.33.99.1
Database server hostname IP or FQDN: 99.33.99.1
Database port number [5432]:
Database port number: 5432
Traffic Ops database user [traffic_ops]:
Traffic Ops database user: traffic_ops
Password for Traffic Ops database user:
Re-Enter Password for Traffic Ops database user:
Writing json to /opt/traffic_ops/app/conf/production/
DATABASE_CONF
Database configuration has been saved
==========/opt/traffic_ops/app/db/dbconf.yml==========
```
Password for database server admin:
Re-Enter Password for database server admin:
Download Maxmind Database? [yes]:
Download Maxmind Database?: yes
===========/opt/traffic_ops/app/conf/cdn.conf===========
Generate a new secret? [yes]:
Generate a new secret?: yes
Number of secrets to keep? [10]:
Number of secrets to keep?: 10
Not setting up ldap
===========/opt/traffic_ops/install/data/json/users.json===========
Administration username for Traffic Ops [admin]:
Administration username for Traffic Ops: admin
Password for the admin user:
Re-Enter Password for the admin user:
Writing json to /opt/traffic_ops/install/data/json/users.json
===========/opt/traffic_ops/install/data/json/openssl_configuration.json===========
Do you want to generate a certificate? [yes]:
Country Name (2 letter code): US
State or Province Name (full name): CO
Locality Name (eg, city): Denver
Organization Name (eg, company): Super CDN, Inc
Organizational Unit Name (eg, section):
Common Name (eg, your name or your server's hostname):
RSA Passphrase:
Re-Enter RSA Passphrase:
writing json to /opt/traffic_ops/install/data/json/profiles.json
human-readable CDN Name. (No whitespace, please)
[cdn1.kabletown.net]: blue_cdn
Human-readable CDN Name. (No whitespace, please): blue_cdn
DNS sub-domain for which your CDN is authoritative:
[cdn1.kabletown.net]: blue-cdn.supercdn.net
DNS sub-domain for which your CDN is authoritative:
blue-cdn.supercdn.net
Writing json to /opt/traffic_ops/install/data/json/profiles.json
Downloading Maxmind data
Resolving geolite.maxmind.com (geolite.maxmind.com)... 2400:cb00:2048:1::6810:262f, 2400:cb00:2048:1::6810:252f, 104.16.38.47, ...
(continues on next page)
Connecting to geolite.maxmind.com (geolite.maxmind.
.com)|2400:cb00:2048:1::6810:262f|:80... connected.

... much SQL output skipped

Starting Traffic Ops
Restarting traffic_ops (via systemctl):

```
[ OK ]
```

Waiting for Traffic Ops to restart
Success! Postinstall complete.

Table 33: Explanation of the information that needs to be provided:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database type</td>
<td>This requests the type of database to be used. Answer with the default - ‘Pg’ to indicate a PostgreSQL database.</td>
</tr>
<tr>
<td>Database name</td>
<td>The name of the database Traffic Ops uses to store the configuration information.</td>
</tr>
<tr>
<td>Database server hostname IP</td>
<td>The hostname of the database server (pg in the example).</td>
</tr>
<tr>
<td>or FQDN</td>
<td></td>
</tr>
<tr>
<td>Database port number</td>
<td>The database port number. The default value, 5432, should be correct unless you changed it during the setup.</td>
</tr>
<tr>
<td>Traffic Ops database user</td>
<td>The username Traffic Ops will use to read/write from the database.</td>
</tr>
<tr>
<td>Password for Traffic Ops</td>
<td>The password for the database user that Traffic Ops uses.</td>
</tr>
<tr>
<td>Database server root (admin)</td>
<td>Privileged database user that has permission to create the database and user for Traffic Ops.</td>
</tr>
<tr>
<td>user name</td>
<td></td>
</tr>
<tr>
<td>Database server root (admin)</td>
<td>The password for the privileged database user.</td>
</tr>
<tr>
<td>user password</td>
<td></td>
</tr>
<tr>
<td>Traffic Ops URL</td>
<td>The URL to connect to this instance of Traffic Ops, usually https://&lt;Traffic Ops host&gt;/</td>
</tr>
<tr>
<td>Human-readable CDN Name</td>
<td>The name of the first CDN which Traffic Ops will be manage.</td>
</tr>
<tr>
<td>DNS sub-domain for which</td>
<td>The DNS domain that will be delegated to this Traffic Control CDN.</td>
</tr>
<tr>
<td>your CDN is authoritative</td>
<td></td>
</tr>
<tr>
<td>Administration username for</td>
<td>The Administration (highest privilege) Traffic Ops user to create. Use this user to login for the first time and create other users.</td>
</tr>
<tr>
<td>Traffic Ops</td>
<td></td>
</tr>
<tr>
<td>Password for the admin user</td>
<td>The password for the administrative Traffic Ops user.</td>
</tr>
</tbody>
</table>
Traffic Ops is now installed!

See also:

Default Profiles for initial configuration of the Traffic Ops instance.

Upgrading Traffic Ops

To upgrade from older Traffic Ops versions, stop the service, use `yum(8)` to upgrade to the latest available Traffic Ops package, and use the `admin` tool to perform the database upgrade.

See also:

`app/db/admin` for more details about `admin`.

# 24: Sample Script for Upgrading Traffic Ops

```bash
systemctl stop traffic_ops
yum upgrade traffic_ops
pushd /opt/traffic_ops/app/
./db/admin --env production upgrade
popd
```

After this completes, see Installation for instructions on running the postinstall script. Once the postinstall script has finished, run the following command as the root user (or with `sudo`): `systemctl start traffic_ops` to start the service.

3.1.4 Traffic Ops - Using

Deprecated since version 3.0: The Traffic Ops UI is deprecated, and will be removed entirely in the next major release (4.0). A much better way to interact with the CDN is to use Traffic Portal, which is the only UI that will be receiving updates for the foreseeable future.

The Traffic Ops Menu

Fig. 2: The Traffic Ops Landing Page

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:
### 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.

## 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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload Server CSV</td>
<td>Bulk add of servers from a CSV file.</td>
</tr>
</tbody>
</table>

## Parameters

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Profile</td>
<td>The table of global parameters. See <em>Profile Parameters</em>. This is where you Create/Read/Update/Delete parameters in the Global profile.</td>
</tr>
<tr>
<td>All Cache Groups</td>
<td>The table of all parameters <em>that are assigned to a cachegroup</em> - this may be slow to pull up, as there can be thousands of parameters.</td>
</tr>
<tr>
<td>All Profiles</td>
<td>The table of all parameters <em>that are assigned to a profile</em> - this may be slow to pull up, as there can be thousands of parameters.</td>
</tr>
<tr>
<td>Select Profile</td>
<td>Select the parameter list by profile first, then get a table of just the parameters for that profile.</td>
</tr>
<tr>
<td>Export Profile</td>
<td>Profiles can be exported from one Traffic Ops instance to another using ‘Select Profile’ and under the “Profile Details” dialog for the desired profile.</td>
</tr>
<tr>
<td>Import Profile</td>
<td>Profiles can be imported from one Traffic Ops instance to another using the button “Import Profile” after using the “Export Profile” feature.</td>
</tr>
<tr>
<td>Orphaned Parameters</td>
<td>A table of parameters that are not associated to any profile of <em>Cache Group</em>. These parameters either should be deleted or associated with a profile of <em>Cache Group</em>.</td>
</tr>
</tbody>
</table>

3.1. Administrator's Guide 63
Tools

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate ISO</td>
<td>Generate a bootable image for any of the servers in the Servers table (or any server for that matter). See Generate ISO</td>
</tr>
<tr>
<td>Queue Updates</td>
<td>Send Updates to the caches. See Queue Updates and Snapshot CRConfig</td>
</tr>
<tr>
<td>DB Dump</td>
<td>Backup the Database to a .sql file.</td>
</tr>
<tr>
<td>Snapshot CR-Config</td>
<td>Send updates to the Traffic Monitor / Traffic Router servers. See Queue Updates and Snapshot CRConfig</td>
</tr>
<tr>
<td>Invalidate Content</td>
<td>Invalidate or purge content from all caches in the CDN. See Invalidate Content</td>
</tr>
<tr>
<td>Manage DNSSEC keys</td>
<td>Manage DNSSEC Keys for a chosen CDN.</td>
</tr>
</tbody>
</table>

Misc

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

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

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 submenus for this tab.
Help

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>About</td>
<td>Traffic Ops information, such as version, database information, etc</td>
</tr>
<tr>
<td>Release Notes</td>
<td>Release notes for the most recent releases of Traffic Ops</td>
</tr>
<tr>
<td>Logout</td>
<td>Logout from Traffic Ops</td>
</tr>
</tbody>
</table>

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.
- **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.
- **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.
- **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 statistics 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

The server checks page is intended to give an overview of the Servers managed by Traffic Control as well as their status. This data comes from Traffic Ops extensions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILO</td>
<td>Ping the iLO interface for EDGE or MID servers</td>
</tr>
<tr>
<td>10G</td>
<td>Ping the IPv4 address of the EDGE or MID servers</td>
</tr>
<tr>
<td>10G6</td>
<td>Ping the IPv6 address of the EDGE or MID servers</td>
</tr>
<tr>
<td>MTU</td>
<td>Ping the EDGE or MID using the configured MTU from Traffic Ops</td>
</tr>
<tr>
<td>FQDN</td>
<td>DNS check that matches what the DNS servers responds with compared to what Traffic Ops has.</td>
</tr>
<tr>
<td>DSCP</td>
<td>Checks the DSCP value of packets from the edge server to the Traffic Ops server.</td>
</tr>
<tr>
<td>RTR</td>
<td>Content Router checks. Checks the health of the Content Routers. Checks the health of the caches using the Content Routers.</td>
</tr>
<tr>
<td>CHR</td>
<td>Cache Hit Ratio in percent.</td>
</tr>
<tr>
<td>CDU</td>
<td>Total Cache Disk Usage in percent.</td>
</tr>
<tr>
<td>ORT</td>
<td>Operational Readiness Test. Uses the ORT script on the edge and mid servers to determine if the configuration in Traffic Ops matches the configuration on the edge or mid. The user that this script runs as must have an ssh key on the edge servers.</td>
</tr>
</tbody>
</table>

Daily Summary

Displays daily max gbps and bytes served for all CDNs. In order for the graphs to appear, the ‘daily_bw_url’ and ‘daily_served_url’ parameters need to be created, assigned to the global profile, and have a value of a grafana graph. For more information on configuring grafana, see the Traffic Stats section.

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 IPAddr 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 icon 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:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGE</td>
<td>Edge Cache</td>
</tr>
<tr>
<td>MID</td>
<td>Mid Tier Cache</td>
</tr>
<tr>
<td>ORG</td>
<td>Origin</td>
</tr>
<tr>
<td>CCR</td>
<td>Traffic Router</td>
</tr>
<tr>
<td>RASCAL</td>
<td>Rascal health polling &amp; reporting</td>
</tr>
<tr>
<td>TOOLS_SERVER</td>
<td>Ops hosts for management</td>
</tr>
<tr>
<td>RIAK</td>
<td>Riak keystore</td>
</tr>
<tr>
<td>SPLUNK</td>
<td>SPLUNK indexer search head etc</td>
</tr>
<tr>
<td>TRAFFIC_STATS</td>
<td>traffic_stats server</td>
</tr>
<tr>
<td>INFLUXDB</td>
<td>influxDb server</td>
</tr>
</tbody>
</table>

The Coverage Zone File and ASN Table

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

```json
{
    "coverageZones": {
        "cache-group-01": {
            "coordinates": {
                "latitude": 1.1,
                "longitude": 2.2
            },
            "network6": [
                "1234:5678::/64",
                "1234:5679::/64"
            ],
            "network": [
                "192.168.8.0/24",
                "192.168.9.0/24"
            ]
        },
        "cache-group-02": {
            "coordinates": {
                "latitude": 3.3,
                "longitude": 4.4
            },
            "network6": [
                "1234:567a::/64",
                "1234:567b::/64"
            ],
            "network": [
                "192.168.4.0/24",
                "192.168.5.0/24"
            ]
        }
    }
}
```

(continues on next page)
The Deep Coverage Zone File

The Deep Coverage Zone File (DCZF) format is similar to the CZF format but adds a `caches` list under each `deepCoverageZone`:

```json
{
    "deepCoverageZones": {
        "location-01": {
            "coordinates": {
                "latitude": 5.5,
                "longitude": 6.6
            },
            "network6": [
                "1234:5678::/64",
                "1234:5679::/64"
            ],
            "network": [
                "192.168.8.0/24",
                "192.168.9.0/24"
            ],
            "caches": [
                "edge-01",
                "edge-02"
            ]
        },
        "location-02": {
            "coordinates": {
                "latitude": 7.7,
                "longitude": 8.8
            },
            "network6": [
                "1234:567a::/64",
                "1234:567b::/64"
            ],
            "network": [
                "192.168.4.0/24",
                "192.168.5.0/24"
            ],
            "caches": [
                "edge-02",
                "edge-03"
            ]
        }
    }
}
```
Each entry in the `caches` list is the hostname of an edge cache registered in Traffic Ops which will be used for “deep” caching in that Deep Coverage Zone. Unlike a regular CZF, coverage zones in the DCZF do not map to a `Cache Group` in Traffic Ops, so currently the deep coverage zone name only needs to be unique.

If the Traffic Router gets a DCZF “hit” for a requested `Delivery Service` that has Deep Caching enabled, the client will be routed to an available “deep” cache from that zone’s `caches` list.

**Note:** The "coordinates" section is optional.

### 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:**

Profile Parameters in the Configuring Traffic Ops section.

### Traffic Router Profile

<table>
<thead>
<tr>
<th>Name</th>
<th>Config_file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>dns.zone</td>
<td>Location to store the DNS zone files in</td>
</tr>
<tr>
<td>location</td>
<td>http-log4j.properties</td>
<td>Location to find the log4j.properties file</td>
</tr>
<tr>
<td>location</td>
<td>dns-log4j.properties</td>
<td>Location to find the dns-log4j.properties file</td>
</tr>
<tr>
<td>location</td>
<td>geolocation.properties</td>
<td>Location to find the geolocation.properties file</td>
</tr>
<tr>
<td>CDN_name</td>
<td>rascal-config.txt</td>
<td>The human readable name of the CDN</td>
</tr>
<tr>
<td>CoverageZoneJsonURL</td>
<td>CRConfig.xml</td>
<td>The location (URL) to retrieve the coverage zone map file in JSON format from.</td>
</tr>
<tr>
<td>ecsEnable</td>
<td>CRConfig.json</td>
<td>Boolean value to enable or disable ENDS0 client subnet extensions.</td>
</tr>
<tr>
<td>geolocation.polling.url</td>
<td>CRConfig.json</td>
<td>The location (URL) to retrieve the coverage geo database file.</td>
</tr>
<tr>
<td>geolocation.polling.interval</td>
<td>CRConfig.json</td>
<td>How often to refresh the coverage geo database file in ms.</td>
</tr>
<tr>
<td>coveragezone.polling.interval</td>
<td>CRConfig.json</td>
<td>How often to refresh the coverage zone map in ms.</td>
</tr>
<tr>
<td>coveragezone.polling.url</td>
<td>CRConfig.json</td>
<td>The location (URL) to retrieve the coverage zone map file in JSON format from.</td>
</tr>
<tr>
<td>deepcoveragezone.polling.interval</td>
<td>CRConfig.json</td>
<td>How often to refresh the deep coverage zone map in ms.</td>
</tr>
<tr>
<td>deepcoveragezone.polling.url</td>
<td>CRConfig.json</td>
<td>The location (URL) to retrieve the deep coverage zone map file in JSON format from.</td>
</tr>
<tr>
<td>Name</td>
<td>Config_file</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tld.soa.expire</td>
<td>CRConfig.json</td>
<td>The value for the expire field the Traffic Router DNS Server will respond with on Start of Authority (SOA) records.</td>
</tr>
<tr>
<td>tld.soa.minimum</td>
<td>CRConfig.json</td>
<td>The value for the minimum field the Traffic Router DNS Server will respond with on SOA records.</td>
</tr>
<tr>
<td>tld.soa.admin</td>
<td>CRConfig.json</td>
<td>The DNS Start of Authority admin. Should be a valid support email address for support if DNS is not working correctly.</td>
</tr>
<tr>
<td>tld.soa.retry</td>
<td>CRConfig.json</td>
<td>The value for the retry field the Traffic Router DNS Server will respond with on SOA records.</td>
</tr>
<tr>
<td>tld.soa.refresh</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on SOA records.</td>
</tr>
<tr>
<td>tld.ttls.SOAsOA</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on SOA records.</td>
</tr>
<tr>
<td>tld.ttls.AAAAA</td>
<td>CRConfig.json</td>
<td>The Time To Live (TTL) the Traffic Router DNS Server will respond with on AAAA records.</td>
</tr>
<tr>
<td>tld.ttls.A</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on A records.</td>
</tr>
<tr>
<td>tld.ttls.DNSKEY</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on DNSKEY records.</td>
</tr>
<tr>
<td>tld.ttls.DS</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on DS records.</td>
</tr>
<tr>
<td>api.port</td>
<td>server.xml</td>
<td>The TCP port Traffic Router listens on for API (REST) access.</td>
</tr>
<tr>
<td>api.cache-control.max-age</td>
<td>CRConfig.json</td>
<td>The value of the Cache-Control: max-age= header in the API responses of Traffic Router.</td>
</tr>
<tr>
<td>api.auth.url</td>
<td>CRConfig.json</td>
<td>The API authentication URL (https://${tmHostname}/api/1.1/user/login); ${tmHostname} is a search and replace token used by Traffic Router to construct the correct URL.</td>
</tr>
<tr>
<td>consistent.dns.routing</td>
<td>CRConfig.json</td>
<td>Control whether DNS Delivery Service use consistent hashing on the edge FQDN to select caches for answers. May improve performance if set to true; defaults to false.</td>
</tr>
<tr>
<td>dnssec.enabled</td>
<td>CRConfig.json</td>
<td>Whether DNSSEC is enabled; this parameter is updated via the DNSSEC administration user interface.</td>
</tr>
<tr>
<td>dnssec.allow.expired.keys</td>
<td>CRConfig.json</td>
<td>Allow Traffic Router to use expired DNSSEC keys to sign zones; default is true. This helps prevent DNSSEC related outages due to failed Traffic Control components or connectivity issues.</td>
</tr>
<tr>
<td>dynamic.cache.primer.enabled</td>
<td>CRConfig.json</td>
<td>Allow Traffic Router to attempt to prime the dynamic zone cache; defaults to true.</td>
</tr>
<tr>
<td>dynamic.cache.primer.limit</td>
<td>CRConfig.json</td>
<td>Limit the number of permutations to prime when dynamic zone cache priming is enabled; defaults to 500.</td>
</tr>
<tr>
<td>keystore.maintenance.interval</td>
<td>CRConfig.json</td>
<td>The interval in seconds which Traffic Router will check the keystore API for new DNSSEC keys.</td>
</tr>
<tr>
<td>keystore.api.url</td>
<td>CRConfig.json</td>
<td>The keystore API URL (https://${tmHostname}/api/1.1/cdns/name/${cdnName}/dnsseckeys.json; ${tmHostname} and ${cdnName} are search and replace tokens used by Traffic Router to construct the correct URL).</td>
</tr>
<tr>
<td>keystore.fetch.timeout</td>
<td>CRConfig.json</td>
<td>The timeout in milliseconds for requests to the keystore API.</td>
</tr>
<tr>
<td>keystore.fetch.retries</td>
<td>CRConfig.json</td>
<td>The number of times Traffic Router will attempt to load keys before giving up; defaults to 5.</td>
</tr>
<tr>
<td>keystore.fetch.wait</td>
<td>CRConfig.json</td>
<td>The number of milliseconds Traffic Router will wait before a retry.</td>
</tr>
<tr>
<td>signaturemanager.expiration.multiplier</td>
<td>CRConfig.json</td>
<td>Multiplier used in conjunction with a zone’s maximum TTL to calculate DNSSEC signature durations; defaults to 5.</td>
</tr>
<tr>
<td>zonemanager.threadpool.scale</td>
<td>CRConfig.json</td>
<td>Multiplier used to determine the number of cores to use for zone signing operations; defaults to 0.75.</td>
</tr>
<tr>
<td>zonemanager.maintenance.interval</td>
<td>CRConfig.json</td>
<td>The interval in seconds which Traffic Router will check for zones that need to be resigned or if dynamic zones need to be expired from cache.</td>
</tr>
<tr>
<td>zonemanager.dynamic.response.expiration</td>
<td>CRConfig.json</td>
<td>A string (e.g.: 300s) that defines how long a dynamic zone should remain in the cache.</td>
</tr>
<tr>
<td>DNSKEY:generation.multiplier</td>
<td>CRConfig.json</td>
<td>Used to determine when new keys need to be generated.</td>
</tr>
<tr>
<td>DNSKEY.effective.multiplier</td>
<td>CRConfig.json</td>
<td>Used when creating an effective date for new key sets.</td>
</tr>
</tbody>
</table>

**Tools**

**Generate ISO**

Generate ISO is a tool for building custom ISOs for building caches on remote hosts. Currently it only supports Centos 7, but if you’re brave and pure of heart you MIGHT be able to get it to work with other unix-like OS’s.

The interface is mostly self-explanatory as it’s got hints.
Field | Explanation
--- | ---
Choose a server from list: | This option gets all the server names currently in the Traffic Ops database and will autofill known values.
OS Version: | There needs to be an _osversions.cfg_ file in the ISO directory that maps the name of a directory to a name that shows up here.
Hostname: | This is the FQDN of the server to be installed. It is required.
Root password: | If you don’t put anything here it will default to the salted MD5 of “Fred”. Whatever put is MD5 hashed and written to disk.
DHCP: | if yes, other IP settings will be ignored
IP Address: | Required if DHCP=no
Netmask: | Required if DHCP=no
Gateway: | Required if DHCP=no
IPV6 Address: | Optional. /64 is assumed if prefix is omitted
IPV6 Gateway: | Ignored if an IPV4 gateway is specified
Network Device: | Optional. Typical values are bond0, eth4, etc. Note: if you enter bond0, a LACP bonding config will be written
MTU: | If unsure, set to 1500
Specify disk for OS install: | Optional. Typical values are “sda”.

When you click the **Download ISO** button the following occurs (all paths relative to the top level of the directory specified in _osversions.cfg_):

1. Reads /etc/resolv.conf to get a list of nameservers. This is a rather ugly hack that is in place until we get a way of configuring it in the interface.

2. Writes a file in the ks_scripts/state.out that contains directory from _osversions.cfg_ and the mkisofs string that we’ll call later.

3. Writes a file in the ks_scripts/network.cfg that is a bunch of key=value pairs that set up networking.

4. Creates an MD5 hash of the password you specify and writes it to ks_scripts/password.cfg. Note that if you do not specify a password “Fred” is used. Also note that we have experienced some issues with web browsers autofilling that field.

5. Writes out a disk configuration file to ks_scripts/disk.cfg.

6. mkisofs is called against the directory configured in _osversions.cfg_ and an ISO is generated in memory and delivered to your web browser.

You now have a customized ISO that can be used to install Red Hat and derivative Linux installations with some modifications to your ks.cfg file.

Kickstart/Anaconda will mount the ISO at /mnt/stage2 during the install process (at least with 6).

You can directly include the password file anywhere in your ks.cfg file (usually in the top) by doing `%include /mnt/stage2/ks_scripts/password.cfg`
What we currently do is have 2 scripts, one to do hard drive configuration and one to do network configuration. Both are relatively specific to the environment they were created in, and both are probably wrong for other organizations, however they are currently living in the “misc” directory as examples of how to do things.

We trigger those in a %pre section in ks.cfg and they will write config files to /tmp. We will then include those files in the appropriate places using %pre.

For example this is a section of our ks.cfg file:

```bash
%include /mnt/stage2/ks_scripts/packages.txt
%pre
    python /mnt/stage2/ks_scripts/create_network_line.py
    bash /mnt/stage2/ks_scripts/drive_config.sh
%end
```

These two scripts will then run _before_ anaconda sets up it’s internal structures, then a bit further up in the ks.cfg file (outside of the %pre %end block) we do an

```bash
%include /mnt/stage2/ks_scripts/password.cfg
...%include /tmp/network_line
%include /tmp/drive_config
...
```

This snarfs up the contents and inlines them.

If you only have one kind of hardware on your CDN it is probably best to just put the drive config right in the ks.cfg.

If you have simple networking needs (we use bonded interfaces in most, but not all locations and we have several types of hardware meaning different ethernet interface names at the OS level etc.) then something like this:

```bash
#!/bin/bash
source /mnt/stage2/ks_scripts/network.cfg
echo "network --bootproto=static --activate --ipv6=${IPV6ADDR} --ip=${IPADDR} --netmask=${NETMASK} --gateway=${GATEWAY} --ipv6gateway=${GATEWAY} --nameserver=${NAMESERVER} --mtu=${MTU} --hostname=${HOSTNAME}" >> /tmp/network.cfg
```

"Note:: that this is an example and may not work at all.

You could also put this in the %pre section. Lots of ways to solve it.

We have included the two scripts we use in the “misc” directory of the git repo:

- kickstart_create_network_line.py
- kickstart_drive_config.sh

These scripts were written to support a very narrow set of expectations and environment and are almost certainly not suitable to just drop in, but they might provide a good starting point.
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 should 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 queue updates for or select All.
3. Select the Cache Group to queue updates for or select 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 exists; Traffic Monitor polls Traffic Ops for a new CRConfig, and Traffic Router polls Traffic Monitor for the same file. This is necessary to ensure that Traffic Monitor sees configuration changes first, which helps to ensure that the health and state of caches and delivery services propagates properly to Traffic Router. See Traffic Router Profile 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.

Note: Snapshotting the CDN also deletes all HTTPS certificates for every Delivery Service which has been deleted since the last Snapshot.

Invalidate Content

Invalidating content on the CDN is sometimes necessary when the origin was mis-configured and something is cached in the CDN 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 inaccessible 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 rule will be active for. It usually makes sense to make this the same as the Cache-Control header from the origin which 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 revalidation rule will be made active. It is pre-populated with the current time, leave as is to schedule ASAP.

3. Click the **Submit** button.

**Manage DNSSEC Keys**

In order to support DNSSEC in Traffic Router, Traffic Ops provides some actions for managing DNSSEC keys for a CDN and associated Delivery Services. DNSSEC Keys consist of a Key Signing Keys (KSK) which are used to sign other DNSKEY records as well as Zone Signing Keys (ZSK) which are used to sign other records. DNSSEC Keys are stored in Traffic Vault and should only be accessible to Traffic Ops. Other applications needing access to this data, such as Traffic Router, must use the Traffic Ops DNSSEC APIs to retrieve this information.

To Manage DNSSEC Keys: 1. Click **Tools -> Manage DNSSEC Keys** 2. Choose a CDN and click **Manage DNSSEC Keys**

   - If keys have not yet been generated for a CDN, this screen will be mostly blank with just the **CDN** and **DNSSEC Active?** fields being populated.

   - If keys have been generated for the CDN, the Manage DNSSEC Keys screen will show the TTL and Top Level Domain (TLD) KSK (Key Signing Key) Expiration for the CDN as well as DS Record information which will need to be added to the parent zone of the TLD in order for DNSSEC to work.

The Manage DNSSEC Keys screen also allows a user to perform the following actions:

**Activate/Deactivate DNSSEC for a CDN**

Fairly straight forward, this button set the **dnssec.enabled** param to either **true** or **false** on the Traffic Router profile for the CDN. The Activate/Deactivate option is only available if DNSSEC keys exist for CDN. In order to active DNSSEC for a CDN a user must first generate keys and then click the **Active DNSSEC** button.
Generate Keys

Generate Keys will generate DNSSEC keys for the CDN TLD as well as for each Delivery Service in the CDN. It is important to note that this button will create a new KSK for the TLD and, therefore, a new DS Record. Any time a new DS Record is created, it will need to be added to the parent zone of the TLD in order for DNSSEC to work properly. When a user clicks the Generate Keys button, they will be presented with a screen with the following fields:

- **CDN**: This is not editable and displays the CDN for which keys will be generated.
- **ZSK Expiration (Days)**: Sets how long (in days) the Zone Signing Key will be valid for the CDN and associated Delivery Services. The default is 30 days.
- **KSK Expiration (Days)**: Sets how long (in days) the Key Signing Key will be valid for the CDN and associated Delivery Services. The default is 365 days.
- **Effective Date (GMT)**: The time from which the new keys will be active. Traffic Router will use this value to determine when to start signing with the new keys and stop signing with the old keys.

Once these fields have been correctly entered, a user can click Generate Keys. The user will be presented with a confirmation screen to help them understand the impact of generating the keys. If a user confirms, the keys will be generated and stored in Traffic Vault.

Regenerate KSK

Regenerate KSK will create a new Key Signing Key for the CDN TLD. A new DS Record will also be generated and need to be put into the parent zone in order for DNSSEC to work correctly. The Regenerate KSK button is only available if keys have already been generated for a CDN. The intent of the button is to provide a mechanism for generating a new KSK when a previous one expires or if necessary for other reasons such as a security breach. When a user goes to generate a new KSK they are presented with a screen with the following options:

- **CDN**: This is not editable and displays the CDN for which keys will be generated.
- **KSK Expiration (Days)**: Sets how long (in days) the Key Signing Key will be valid for the CDN and associated Delivery Services. The default is 365 days.
- **Effective Date (GMT)**: The time from which the new KSK and DS Record will be active. Since generating a new KSK will generate a new DS Record that needs to be added to the parent zone, it is very important to make sure that an effective date is chosen that allows for time to get the DS Record into the parent zone. Failure to get the new DS Record into the parent zone in time could result in DNSSEC errors when Traffic Router tries to sign responses.

Once these fields have been correctly entered, a user can click Generate KSK. The user will be presented with a confirmation screen to help them understand the impact of generating the KSK. If a user confirms, the KSK will be generated and stored in Traffic Vault.

Additionally, Traffic Ops also performs some systematic management of DNSSEC keys. This management is necessary to help keep keys in sync for Delivery Services in a CDN as well as to make sure keys do not expire without human intervention.
Generation of keys for new Delivery Services

If a new Delivery Service is created and added to a CDN that has DNSSEC enabled, Traffic Ops will create DNSSEC keys for the Delivery Service and store them in Traffic Vault.

Regeneration of expiring keys for a Delivery Service

Traffic Ops has a process, controlled by cron(8), to check for expired or expiring keys and re-generate them. The process runs at 5 minute intervals to check and see if keys are expired or close to expiring (within 10 minutes by default). If keys are expired for a Delivery Service, Traffic Ops will regenerate new keys and store them in Traffic Vault. This process is the same for the CDN TLD (Top-Level Domain) ZSK (Zone Signing Key), however Traffic Ops will not re-generate the CDN TLD KSK systematically. The reason is that when a KSK is regenerated for the CDN TLD then a new DS Record will also be created. The new DS Record needs to be added to the parent zone before Traffic Router attempts to sign with the new KSK in order for DNSSEC to work correctly. Therefore, management of the KSK needs to be a manual process.

3.1.5 Default Profiles

Traffic Ops has the concept of Parameters and Profiles, which are an integral component of Traffic Ops. To get started, a set of default Traffic Ops profiles are provided. These can be imported into Traffic Ops, and are required by the Traffic Control components Traffic Router, Traffic Monitor, and Apache Traffic Server (Edge-tier and Mid-tier caches). Download Default Profiles from here

Minimum Traffic Ops Profiles Needed

- EDGE_ATS_version_platform_PROFILE.traffic_ops
- MID_ATS_version_platform_PROFILE.traffic_ops
- TRAFFIC_MONITOR_PROFILE.traffic_ops
- TRAFFIC_ROUTER_PROFILE.traffic_ops
- TRAFFIC_STATS_PROFILE.traffic_ops
- EDGE_GROVE_PROFILE.traffic_ops

Note: Despite that these have the .traffic_ops extension, they use JSON to store data. If your syntax highlighting doesn’t work in some editor or viewer, try changing the extension to .json.
Steps to Import a Profile

1. Sign into Traffic Portal
2. Navigate to Configure → Profiles
3. Click on More → Import Profile
4. Drag and drop your desired profile into the upload pane
5. Click Import
6. Continue these steps for each of the Minimum Traffic Ops Profiles Needed.

3.1.6 Traffic Portal Administration

Traffic Portal is only supported on CentOS Linux distributions version 7.x. It runs on NodeJS and requires version 6.0 or higher.

Installing Traffic Portal

1. Download the Traffic Portal RPM from Apache Jenkins or build the Traffic Portal RPM from source using the instructions in Building Traffic Control.
2. Copy the Traffic Portal RPM to your server
3. Install NodeJS. This can be done by building it from source, installing with yum(8) if it happens to be in your available repositories (at version 6.0+), or using the NodeSource setup script.

# 25: Installing NodeJS using the NodeSource Setup Script

curl --silent --location https://rpm.nodesource.com/\n  → setup_6.x | sudo bash -

4. Install the Traffic Portal RPM with yum(8) or rpm(8) e.g. by running

    yum install path/to/traffic_portal.rpm as the root user or with sudo(8).

Configuring Traffic Portal

- update /etc/traffic_portal/conf/config.js (if Traffic Portal is being upgraded, reconcile config.js with config.js.rpmnew and then delete config.js.rpmnew)
• update /opt/traffic_portal/public/traffic_portal_properties.json (if Traffic Portal is being upgraded, reconcile traffic_portal_properties.json with traffic_portal_properties.json.rpmnew and then delete traffic_portal_properties.json.rpmnew)


Starting Traffic Portal

The Traffic Portal RPM comes with a `systemd(1)` unit file, so under normal circumstances Traffic Portal may be started with `systemctl(1)`.

```
# 26: Starting Traffic Portal
systemctl start traffic_portal
```

Stopping Traffic Portal

The Traffic Portal RPM comes with a `systemd(1)` unit file, so under normal circumstances Traffic Portal may be stopped with `systemctl(1)`.

```
# 27: Stopping Traffic Portal
systemctl stop traffic_portal
```

3.1.7 Traffic Portal - Using

Traffic Portal is the official Traffic Control UI. Traffic Portal typically runs on a different machine than Traffic Ops, and works by using the Traffic Ops API. The following high-level items are available in the Traffic Portal menu.

![Traffic Portal Start Page](image)

Fig. 3: Traffic Portal Start Page
Dashboard

The Dashboard is the default landing page for Traffic Portal. It provides a real-time view into the main performance indicators of the CDNs managed by Traffic Control. It also displays various statistics about the overall health of your CDN.

**Current Bandwidth**  The current bandwidth of all of your CDNs.

**Current Connections**  The current number of connections to all of your CDNs.

**Healthy Caches**  Displays the number of healthy caches across all CDNs. Click the link to view the healthy caches on the cache stats page.

**Unhealthy Caches**  Displays the number of unhealthy caches across all CDNs. Click the link to view the unhealthy caches on the cache stats page.

**Online Caches**  Displays the number of cache servers with ONLINE status. Traffic Monitor will not monitor the state of ONLINE servers.

**Reported Caches**  Displays the number of cache servers with REPORTED status.

**Offline Caches**  Displays the number of cache servers with OFFLINE status.

**Admin Down Caches**  Displays the number of caches with ADMIN_DOWN status.

Each component of this view is updated on the intervals defined in the `traffic_portal_properties.json` configuration file.

CDNs

A table of CDNs with the following columns:

- **Name**  The name of the CDN
- **Domain**  The CDN’s Top-Level Domain (TLD)
- **DNSSEC Enabled**  ‘true’ if DNSSEC is enabled on this CDN, ‘false’ otherwise.

CDN management includes the ability to (where applicable):

- create a new CDN
- update an existing CDN
- delete an existing CDN
- queue/clear updates on all servers in a CDN
- diff CDN snapshots
- create a CDN snapshot
- manage a CDN’s DNSSEC keys
- manage a CDN’s federations
- view Delivery Services of a CDN

---

1 For more information, see Health Protocol.
• view CDN profiles
• view servers within a CDN

**Monitor**

The *Monitor* section of Traffic Portal is used to display statistics regarding the various *cache servers* within all CDNs visible to the user. It retrieves this information through the Traffic Ops API from Traffic Monitor instances.

![Monitor Menu](image)

**Fig. 4: The ‘Monitor’ Menu**

**Cache Checks**

A real-time view into the status of each cache. The *Monitor → Cache Checks* page is intended to give an overview of the caches managed by Traffic Control as well as their status.

- **Hostname** Cache host name
- **Profile** The name of the profile applied to the cache
- **Status** The status of the cache (one of: ONLINE, REPORTED, ADMIN_DOWN, OFFLINE)

See also:

*Health Protocol*

- **UPD** Configuration updates pending for an Edge-tier or Mid-tier *cache server*
- **RVL** Content invalidation requests are pending for this server and/or its parent(s)
- **ILO** Ping the iLO interface for Edge-tier or Mid-tier *cache servers*
- **10G** Ping the IPv4 address of the Edge-tier or Mid-tier *cache servers*
- **FQDN** DNS check that matches what the DNS servers responds with compared to what Traffic Ops has
- **DSCP** Checks the DSCP value of packets from the Edge-tier *cache server* to the Traffic Ops server
- **10G6** Ping the IPv6 address of the Edge-tier or Mid-tier *cache servers*
MTU  Ping the Edge-tier or Mid-tier using the configured MTU from Traffic Ops
RTR  Content Router checks. Checks the health of the Traffic Router servers. Also checks the health of the cache servers using the Traffic Routers
CHR  Cache Hit Ratio percent
CDU  Total Cache Disk Usage percent
ORT  Operational Readiness Test - uses the ORT script on the Edge-tier and Mid-tier cache servers to determine if the configuration in Traffic Ops matches the configuration on the Edge-tier or Mid-tier. The user as whom this script runs must have an SSH key on the Edge-tier servers.

Cache Stats

A table showing the results of the periodic Check Extensions that are run. These can be grouped by Cache Group and/or Profile.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>Name of the Profile applied to the Edge-tier or Mid-tier cache server</td>
</tr>
<tr>
<td>Host</td>
<td>‘ALL’ for entries grouped by Cache Group, or the hostname of a particular cache server</td>
</tr>
<tr>
<td>Cache Group</td>
<td>Name of the Cache Group to which this server belongs, or the name of the Cache Group that is grouped for entries grouped by Cache Group</td>
</tr>
<tr>
<td>Healthy</td>
<td>True/False as determined by Traffic Monitor</td>
</tr>
</tbody>
</table>

See also: Health Protocol

Status  Status of the cache server or Cache Group
Connections Number of connections to this cache server or Cache Group
MbpsOut  Data flow rate outward from the CDN (toward client) in Megabits per second

Services

Services groups the functionality to modify Delivery Services - for those users with the necessary permissions - or make Requests for such changes - for users without necessary permissions.

Delivery Service

This page contains a table displaying all Delivery Services visible to the user. Each entry in this table has the following fields:

Key (XML ID)  A unique string that identifies this Delivery Service
Tenant The tenant to which the Delivery Service is assigned
**Origin** The Origin Server’s base URL. This includes the protocol (HTTP or HTTPS). Example: http://movies.origin.com

**Active** When this is set to ‘false’, Traffic Router will not serve DNS or HTTP responses for this Delivery Service

**Type** The type of content routing this Delivery Service will use

- **See also:**
  - **Type**

**Protocol** The protocol which this Delivery Service serves clients. Its value is one of:

- **HTTP** Only insecure requests will be serviced
- **HTTPS** Only secure requests will be serviced
- **HTTP and HTTPS** Both secure and insecure requests will be serviced
- **HTTP to HTTPS** Insecure requests will be redirected to secure locations and secure requests are serviced normally

**CDN** The CDN to which the Delivery Service belongs

**IPv6 Enabled** When set to ‘true’, the Traffic Router will respond to AAAA DNS requests for the routed name of this Delivery Service. Otherwise, only A records will be served

**DSCP** The DSCP value with which to mark IP packets sent to the client

**Signing Algorithm** The algorithm used to sign URLs used by the Delivery Service

**Query String Handling** Describes how the Delivery Service treats query strings. It has one of the following possible values:

- **USE** The query string will be used in the ATS ‘cache key’ and is passed in requests to the origin (each unique query string is treated as a unique URL)
- **IGNORE** The query string will not be used in the ATS ‘cache key’, but will be passed in requests to the origin
**DROP** The query string is stripped from the request URL at the Edge-tier cache, and so is not used in the ATS ‘cache key’, and is not passed in requests to the origin.

**See also:**

*Query String Handling*

**Last Updated** The time at which the *Delivery Service* was last updated

*Delivery Service* management includes the ability to (where applicable):

- create a new *Delivery Service*
- clone an existing *Delivery Service*
- update an existing *Delivery Service*
- delete an existing *Delivery Service*
- compare *Delivery Services*
- manage *Delivery Service* SSL keys
- manage *Delivery Service* URL signature keys
- manage *Delivery Service* URI signing keys
- view and assign *Delivery Service* servers
- create, update and delete *Delivery Service* regular expressions
- view and create *Delivery Service* invalidate content jobs
- manage steering targets
- view and manage static DNS records within a *Delivery Service* subdomain

**See also:**

*Configuring Static DNS Entries*

**Delivery Service Requests**

If enabled in the `traffic_portal_properties.json` configuration file, all *Delivery Service* changes (create, update and delete) are captured as a Delivery Service Request and must be reviewed before fulfillment/deployment.

*Delivery Service*: A unique string that identifies the *Delivery Service* with which the request is associated. This unique string is also known (and often referred to within documentation and source code) as a *Delivery Service* key’ or ‘XML ID’/‘xml_id’/‘xmlid’ :Type: The type of Delivery Service Request: ‘create’, ‘update’, or ‘delete’ according to what was requested :Status: The status of the Delivery Service Request. Has the following possible values:

- **draft** The Delivery Service Request is not ready for review and fulfillment
- **submitted** The Delivery Service Request is ready for review and fulfillment
- **rejected** The Delivery Service Request has been rejected and cannot be modified
**pending** The Delivery Service Request has been fulfilled but the changes have yet to be deployed

**complete** The Delivery Service Request has been fulfilled and the changes have been deployed

**Author** The user responsible for creating the Delivery Service Request

**Assignee** The user responsible for fulfilling the Delivery Service Request. Currently, the operations role or above is required to assign Delivery Service Requests

**Last Edited By** The last user to edit the Delivery Service Request

**Created** Relative time indicating when the Delivery Service Request was created

**Actions** Actions that can be performed on a Delivery Service Request. The following actions are provided:

- **fulfill** Implement the changes captured in the Delivery Service Request
- **reject** Reject the changes captured in the Delivery Service Request
- **delete** Delete the Delivery Service Request

Delivery Service Request management includes the ability to (where applicable):

- create a new Delivery Service Request
- update an existing Delivery Service Request
- delete an existing Delivery Service Request
- update the status of a Delivery Service Request
- assign a Delivery Service Request
- reject a Delivery Service Request
- fulfill a Delivery Service Request
- complete a Delivery Service Request

See also:

*Delivery Service Requests*

**Configure**

Interfaces for managing the various components of Traffic Control and how they interact are grouped under *Configure*.

**Servers**

A table of all servers (of all kinds) across all *Delivery Services* and CDNs visible to the user. It has the following columns:
Fig. 6: The ‘Configure’ Menu

**UPD** ‘true’ when updates to the server’s configuration are pending, ‘false’ otherwise

**Host** The hostname of the server

**Domain** The server’s domain. (The FQDN of the server is given by ‘Host.Domain’)

**IP** The server’s IPv4 address

**IPv6** The server’s IPv6 address

**Status** The server’s status

See also:

*Health Protocol*

**Type** The type of server e.g. EDGE for an Edge-tier *cache server*

**Profile** The name of the server’s *Profile*

**CDN** The name of the CDN to which this server is assigned (if any)

**Cache Group** The name of the *Cache Group* to which this server belongs

**Phys Location** The name of the *Physical Location* to which this server belongs

**ILO** If not empty, this is the IPv4 address of the server’s ILO interface

See also:

Hewlett Packard ILO Wikipedia Page
Server management includes the ability to (where applicable):

- create a new server
- update an existing server
- delete an existing server
- queue/clear updates on a server
- update server status
- view server Delivery Services
- view server configuration files
- clone Delivery Service assignments
- assign Delivery Services to server(s)

**Origins**

A table of all origins. These are automatically created for the origins served by Delivery Services throughout all CDNs, but additional ones can be created at will. The table has the following columns:

- **Name** The name of the origin. If this origin was created automatically for a Delivery Service, this will be the xml_id of that Delivery Service.
- **Tenant** The name of the Tenant that owns this origin - this is not necessarily the same as the Tenant that owns the Delivery Service to which this origin belongs.
- **Primary** Either true to indicate that this is the “primary” origin for the Delivery Service to which it is assigned, or false otherwise.
- **Delivery Service** The xml_id of the Delivery Service to which this origin is assigned.
- **FQDN** The FQDN of the origin server.
- **IPv4 Address** The origin's IPv4 address, if configured.
- **IPv6 Address** The origin's IPv6 address, if configured.
- **Protocol** The protocol this origin uses to serve content. One of
  - http
  - https
- **Port** The port on which the origin server listens for incoming HTTP(S) requests.

**Note:** If this field appears blank in the table, it means that a default was chosen for the origin based on its Protocol - 80 for “http”, 443 for “https”.

---

3.1. Administrator's Guide
Coordinate The name of the geographic coordinate pair that defines the physical location of this origin server. Origins created for Delivery Services automatically will not have associated Coordinates. This can be rectified on the details pages for said origins.

Cachegroup The name of the Cache Group to which this origin belongs, if any.

Profile The name of a Profile used by this origin.

Origin management includes the ability to (where applicable):

• create a new origin
• update an existing origin
• delete an existing origin

Profiles

A table of all Profiles. From here you can see Parameters, servers and Delivery Services assigned to each Profile. Each entry in the table has these fields:

Name The name of the Profile

Type The type of this Profile, which indicates the kinds of objects to which the Profile may be assigned

Routing Disabled For Profiles applied to cache servers (Edge-tier or Mid-tier) this indicates that Traffic Router will refuse to provide routes to these machines

Description A user-defined description of the Profile, typically indicating its purpose

CDN The CDN to which this Profile is restricted. To use the same Profile across multiple CDNs, clone the Profile and change the clone’s CDN field.

Profile management includes the ability to (where applicable):

• create a new Profile
• update an existing Profile
• delete an existing Profile
• clone a Profile
• export a Profile
• view Profile Parameters
• view Profile Delivery Services
• view Profile servers

See also:

Parameters and Profiles
Parameters

This page displays a table of Parameters from all Profiles with the following columns:

- **Name** The name of the Parameter
- **Config File** The configuration file where this Parameter is stored, possibly the special value `location`, indicating that this Parameter actually names the location of a configuration file rather than its contents, or `package` to indicate that this Parameter specifies a package to be installed rather than anything to do with configuration files
- **Value** The value of the Parameter. The meaning of this depends on the value of ‘Config File’
- **Secure** When this is ‘true’, a user requesting to see this Parameter will see the value `********` instead of its actual value if the user’s permission role isn’t ‘admin’
- **Profiles** The number of Profiles currently using this Parameter

Parameter management includes the ability to (where applicable):

- create a new Parameter
- update an existing Parameter
- delete an existing Parameter
- view Parameter Profiles
- manage assignments of a Parameter to one or more Profiles and/or Delivery Services

Types

Types group Delivery Services, servers and Cache Groups for various purposes. Each entry in the table shown on this page has the following fields:

- **Name** The name of the Type
- **Use In Table** States the use of this Type, e.g. server indicates this is a Type assigned to servers
- **Description** A short, usually user-defined, description of the Type

Type management includes the ability to (where applicable):

- create a new Type
- update an existing Type
- delete an existing Type
- view Delivery Services assigned to a Type
- view servers assigned to a Type
• view *Cache Groups* assigned to a *Type*

**Statuses**

This page shows a table of *Statuses* with the following columns:

- **Name** The name of this *Status*
- **Description** A short, usually user-defined, description of this *Status*

*Status* management includes the ability to (where applicable):

- create a new *Status*
- update an existing *Status*
- delete an existing *Status*
- view *Statuses*

**Topology**

*Topology* groups views and functionality that deal with how CDNs and their Traffic Control components are grouped and distributed, both on a logical level as well as a physical level.

![Topology Menu](image)

Fig. 7: ‘Topology’ Menu

**Cache Groups**

This page is a table of *Cache Groups*, each entry of which has the following fields:

- **Name** The full name of this *Cache Group*
Cache Group management includes the ability to (where applicable):

- create a new Cache Group
- update an existing Cache Group
- delete an existing Cache Group
- queue/clear updates for all servers in a Cache Group
- view Cache Group ASN (Autonomous System Number)s

See also:

The Wikipedia page on Autonomous System Numbers

- view and assign Cache Group Parameters
- view Cache Group servers

Coordinates

Topology → Coordinates allows a label to be given to a set of geographic coordinates for ease of use. Each entry in the table on this page has the following fields:

- Name  The name of this coordinate pair
- Latitude  The geographic latitude part of the coordinate pair
- Longitude  The geographic longitude part of the coordinate pair

Coordination management includes the ability to (where applicable):

- create a new coordinate pair
- update an existing coordinate pair
- delete an existing coordinate pair

Phys Locations

A table of Physical Locations which may be assigned to servers and Cache Groups, typically for the purpose of optimizing client routing. Each entry has the following columns:

- Name  The full name of the Physical Location
- Short Name  A shorter, more human-friendly name for this Physical Location
- Address  The Physical Location’s street address (street number and name)
City The city within which the Physical Location resides
State The state within which the Physical Location's city lies
Region The Region to which this Physical Location has been assigned

Physical Location management includes the ability to (where applicable):
  • create a new Physical Location
  • update an existing Physical Location
  • delete an existing Physical Location
  • view Physical Location servers

Divisions

Each entry in the table of Divisions on this page has the following fields:

  Name The name of the Division

Division management includes the ability to (where applicable):
  • create a new Division
  • delete an existing Division
  • modify an existing Division
  • view Regions within a Division

Regions

Each entry in the table of Regions on this page has the following fields:

  Name The name of this Region
  Division The Division to which this Region is assigned

Region management includes the ability to (where applicable):
  • create a new Region
  • update an existing Region
  • delete an existing Region
  • view Physical Locations within a Region

ASNs

Manage ASNs. Each entry in the table on this page has the following fields:

  ASN The actual ASN
**Cache Group** The *Cache Group* to which this ASN is assigned

ASN management includes the ability to (where applicable):

- create a new ASN
- update an existing ASN
- delete an existing ASN

**See also:**

Autonomous System (Internet) Wikipedia Page

**Tools**

*Tools* contains various tools that don’t directly relate to manipulating Traffic Control components or their groupings.

![Tools](image)

Fig. 8: The ‘Tools’ Menu

**Invalidate Content**

Here, specific assets can be invalidated in all caches of a *Delivery Service*, forcing content to be updated from the origin. Specifically, this *doesn’t* mean that *cache server* s will immediately remove items from their caches, but rather will fetch new copies whenever a request is made matching the ‘Asset URL’ regular expression. This behavior persists until the Invalidate Content Job’s TTL expires. Each entry in the table on this page has the following fields:

*Delivery Service:* The *Delivery Service* to which to apply this Invalidate Content Job : Asset URL: A URL or regular expression which describes the asset(s) to be invalidated : Parameters: So far, the only use for this is setting a TTL over which the Invalidate Content Job shall remain active : Start: An effective start time until which the job is delayed : Created By: The user name of the person who created this Invalidate Content Job

Invalidate content includes the ability to (where applicable):

- create a new invalidate content job
Generate ISO

Generates a boot-able system image for any of the servers in the Servers table (or any server for that matter). Currently it only supports CentOS 7, but if you’re brave and pure of heart you MIGHT be able to get it to work with other Unix-like Operating Systems. The interface is mostly self-explanatory, but here is a short explanation of the fields in that form.

Copy Server Attributes From  Optional. This option lets the user choose a server from the Traffic Ops database and will auto-fill the other fields as much as possible based on that server’s properties.

OS Version  This list is populated by modifying the osversions.cfg file on the Traffic Ops server. This file maps OS names to the name of a directory under app/public/iso/ directory within the Traffic Ops install directory.

Hostname  The desired hostname of the resultant system.

Domain  The desired domain name of the resultant system.

DHCP  If this is ‘no’ the IP settings of the system must be specified, and the following extra fields will appear:

- IP Address  The resultant system’s IPv4 Address
- Network Subnet  The system’s network subnet mask
- Network Gateway  The system’s network gateway’s IPv4 Address

Network MTU  The system’s network’s MTU. Despite being a text field, this can only be 1500 or 9000 - it should almost always be 1500.

See also:

The Maximum transmission unit Wikipedia Page

Disk for OS Install  The disk on which to install the base system. A reasonable default is sda (the /dev/ prefix is not necessary).

Root Password  The password to be used for the root user. Input is MD5 hashed before being written to disk.

Confirm Root Password  Repeat the ‘Root Password’ to be sure it’s right.

Interface Name  Optional. The name of the resultant system’s network interface. Typical values are bond0, eth4, etc. If bond0 is entered, a Link Aggregation Control Protocol bonding configuration will be written.

See also:

The Link aggregation Wikipedia Page

Stream ISO  If this is ‘yes’, then the download will start immediately as the ISO is written directly to the socket connection to Traffic Ops. If this is ‘no’, then the download will begin only after the ISO has finished being generated. For almost all use cases, this should be ‘yes’. 
User Admin

This section offers administrative functionality for users and their permissions.

![User Admin Menu](image)

Fig. 9: The ‘User Admin’ Menu

User

This page lists all the users that are visible to the user (so, for ‘admin’ users, all users will appear here). Each entry in the table on this page has the following fields:

- **Full Name**  The user’s full, real name
- **Username**   The user’s username
- **Email**      The user’s email address
- **Tenant**     The user’s Tenant
- **Role**       The user’s Role

User management includes the ability to (where applicable):

- register a new user
- create a new user
- update an existing user
- view Delivery Services visible to a user

Tenants

Each entry in the table of Tenants on this page has the following entries:

- **Name**  The name of the Tenant
- **Active** If ‘true’ users of this Tenant group are allowed to login and have active Delivery Services
**Parent** The parent of this *Tenant*. The default is the ‘root’ *Tenant*, which has no users.

*Tenant* management includes the ability to (where applicable):

- create a new *Tenant*
- update an existing *Tenant*
- delete an existing *Tenant*
- view users assigned to a *Tenant*
- view *Delivery Services* assigned to a *Tenant*

**Roles**

Each entry in the table of *Roles* on this page has the following fields:

- **Name** The name of the *Role*
- **Privilege Level** The privilege level of this *Role*. This is a whole number that actually controls what a user is allowed to do. Higher numbers correspond to higher permission levels
- **Description** A short description of the *Role* and what it is allowed to do

Role management includes the ability to (where applicable):

- view all *Roles*
- create new *Role*

**Note:** *Roles* cannot be deleted through the Traffic Portal UI

**Other**

Custom menu items. By default, this contains only a link to the Traffic Control documentation.

**Docs**

This is just a link to the Traffic Control Documentation.

**Custom Menu Items**

This section is configurable in the `traffic_portal_properties.json` configuration file, in the `customMenu` section.
3.1.8 Traffic Monitor Administration

Installing Traffic Monitor

The following are hard requirements for Traffic Monitor to operate:

- CentOS 7+
- Successful install of Traffic Ops (usually on a separate machine)
- Administrative access to the Traffic Ops (usually on a separate machine)

These are the recommended hardware specifications for a production deployment of Traffic Monitor:

- 8 CPUs
- 16GB of RAM
- It is also recommended that you know the geographic coordinates and/or mailing address of the site where the Traffic Monitor machine lives for optimal performance

1. Enter the Traffic Monitor server into Traffic Portal

   **Note:** For legacy compatibility reasons, the ‘Type’ field of a new Traffic Monitor server must be ‘RASCAL’.

2. Make sure the FQDN of the Traffic Monitor is resolvable in DNS.

3. Install Traffic Monitor, either from source or by installing a `traffic_monitor-version string.rpm` package generated by the instructions in *Building Traffic Control with `yum(8)` or `rpm(8)`*

4. Configure Traffic Monitor according to *Configuring Traffic Monitor*

5. Start Traffic Monitor, usually by starting its `systemd(1)` service
6. Verify Traffic Monitor is running by e.g. opening your preferred web browser to port 80 on the Traffic Monitor host.

Configuring Traffic Monitor

Configuration Overview

Traffic Monitor is configured via two JSON configuration files, `traffic_ops.cfg` and `traffic_monitor.cfg`, by default located in the `conf` directory in the install location. `traffic_ops.cfg` contains Traffic Ops connection information. Specify the URL, username, and password for the instance of Traffic Ops of which this Traffic Monitor is a member. `traffic_monitor.cfg` contains log file locations, as well as detailed application configuration variables such as processing flush times and initial poll intervals. Once started with the correct configuration, Traffic Monitor downloads its configuration from Traffic Ops and begins polling `cache server` s. Once every `cache server` has been polled, `Health Protocol` state is available via RESTful JSON endpoints and a web browser UI.

Stat and Health Flush Configuration

The Monitor has a health flush interval, a stat flush interval, and a stat buffer interval. Recall that the monitor polls both stats and health. The health poll is so small and fast, a buffer is largely unnecessary. However, in a large CDN, the stat poll may involve thousands of `cache server` s with thousands of stats each, or more, and CPU may be a bottleneck.

The flush intervals, `health_flush_interval_ms` and `stat_flush_interval_ms`, indicate how often to flush stats or health, if results are continuously coming in with no break. This prevents starvation. Ideally, if there is enough CPU, the flushes should never occur. The default flush times are 200 milliseconds, which is suggested as a reasonable starting point; operators may adjust them higher or lower depending on the need to get health data and stop directing client traffic to unhealthy `cache server` s as quickly as possible, balanced by the need to reduce CPU usage.

The stat buffer interval, `stat_buffer_interval_ms`, also provides a temporal buffer for stat processing. Stats will not be processed except after this interval, whereupon all pending stats will be processed, unless the flush interval occurs as a starvation safety. The stat buffer and flush intervals may be thought of as a state machine with two states: the “buffer state” accepts results until the buffer interval has elapsed, whereupon the “flush state” is entered, and results are accepted while outstanding, and processed either when no results are outstanding or the flush interval has elapsed.

Note that this means the stat buffer interval acts as “bufferbloat,” increasing the average and maximum time a `cache server` may be down before it is processed and marked as unhealthy. If the stat buffer interval is non-zero, the average time a `cache server` may be down before being marked unavailable is half the poll time plus half the stat buffer interval, and the maximum time is the poll time plus the stat buffer interval. For example, if the stat poll time is 6 seconds, and the stat buffer interval is 4 seconds, the average time a `cache server` may be unhealthy before being marked is $\frac{6}{2} + \frac{4}{2} = 6$ seconds, and the maximum time is $6 + 4 = 10$ seconds. For this...
reason, if operators feel the need to add a stat buffer interval, it is recommended to start with a very low duration, such as 5 milliseconds, and increase as necessary.

It is not recommended to set either flush interval to 0, regardless of the stat buffer interval. This will cause new results to be immediately processed, with little to no processing of multiple results concurrently. Result processing does not scale linearly. For example, processing 100 results at once does not cost significantly more CPU usage or time than processing 10 results at once. Thus, a flush interval which is too low will cause increased CPU usage, and potentially increased overall poll times, with little or no benefit. The default value of 200 milliseconds is recommended as a starting point for configuration tuning.

Troubleshooting and Log Files

Traffic Monitor log files are in `/opt/traffic_monitor/var/log/`.

### 3.1.9 Traffic Router Administration

**Requirements**

- CentOS 7
- 4 CPUs
- 8GB of RAM
- Successful install of Traffic Ops (usually on another machine)
- Successful install of Traffic Monitor (usually on another machine)
- Administrative access to Traffic Ops

**Note:** Hardware requirements are generally doubled if DNSSEC is enabled

**Installing Traffic Router**

1. If no suitable Profile exists, create a new Profile for Traffic Router via the + button on the Profiles page in Traffic Portal

   **Warning:** Traffic Ops will only recognize a profile as assignable to a Traffic Router if its name starts with the prefix `ccr-`. The reason for this is a legacy limitation related to the old name for Traffic Router (Comcast Cloud Router), and will (hopefully) be rectified in the future as the old Perl parts of Traffic Ops are re-written in Go.

2. Enter the Traffic Router server into Traffic Portal on the Servers page (or via the Traffic Ops API), assign to it a Traffic Router Profile, and ensure that its status is set to ONLINE.
3. Ensure the FQDN of the Traffic Router is resolvable in DNS. This FQDN must be resolvable by the clients expected to use this CDN.

4. Install a Traffic Router server package, either from source or using a `traffic_router-version string.rpm` package generated using the instructions in Building Traffic Control.

Changed in version 3.0: As of version 3.0, Traffic Router depends upon a package called `tomcat`. This package should have been created when Traffic Router was built. If installing the `traffic_router` produces a dependency error, make sure that the `tomcat` package is available in an accessible `yum(8)` repository.

5. Edit `/opt/traffic_router/conf/traffic_monitor.properties` and specify the correct online Traffic Monitor(s) for your CDN.

See also:

Traffic Router Parameters

`traffic_monitor.properties` URL that should normally point to this file, e.g. `traffic_monitor.properties=file:/opt/traffic_monitor.properties`

`traffic_monitor.properties.reload.period` Period to wait (in milliseconds) between reloading this file, e.g. `traffic_monitor.properties.reload.period=60000`

6. Start Traffic Router. This is normally done by starting its `systemd(1)` service. `systemctl start traffic_router`, and test DNS lookups against that server to be sure it’s resolving properly, with e.g. `dig` or `curl`. Also, because previously taken CDN Snapshots will be cached, they need to be removed manually to actually be reloaded. This file should be located at `/opt/traffic_router/db/cr-config.json`. This should be done before starting or restarting Traffic Router.

# 28: Starting and Testing Traffic Router

```
[root@trafficrouter /]# systemctl start traffic_router
[root@trafficrouter /]# dig @localhost mycdn.ciab.test

; <<>> DiG 9.9.4-RedHat-9.9.4-72.el7 <<>> @localhost mycdn.ciab.test
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27109
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;mycdn.ciab.test. IN A
```

(continues on next page)
;; AUTHORITY SECTION:
mycdn.ciab.test. 30 IN SOA
  trafficrouter.infra.ciab.test. twelve_monkeys.mycdn.
  ciab.test. 2019010918 28800 7200 604800 30

;; Query time: 28 msec
;; SERVER: ::1#53(::1)
;; WHEN: Wed Jan 09 21:27:57 UTC 2019
;; MSG SIZE rcvd: 104

7. Perform a CDN Snapshot.

See also:

Snapshot CRConfig

Note: Once the Snapshot is taken, live traffic will be sent to the new Traffic Routers provided that their status has been set to ONLINE.

8. Ensure that the parent domain (e.g.: cdn.local) for the CDN’s top level domain (e.g.: ciab.cdn.local) contains a delegation (Name Server records) for the new Traffic Router, and that the value specified matches the FQDN of the Traffic Router.

Configuring Traffic Router

Changed in version 1.5: Many of the configuration files under /opt/traffic_router/conf are now only needed to override the default configuration values for Traffic Router. Most of the given default values will work well for any CDN. Critical values that must be changed are hostnames and credentials for communicating with other Traffic Control components such as Traffic Ops and Traffic Monitor. Pre-existing installations that store configuration files under /opt/traffic_router/conf will still be used and honored for Traffic Router 1.5 onward.

Changed in version 3.0: Traffic Router 3.0 has been converted to a formal Tomcat instance, meaning that is now installed separately from the Tomcat servlet engine. The Traffic Router installation package contains all of the Traffic Router-specific software, configuration and startup scripts including some additional configuration files needed for Tomcat. These new configuration files can all be found in the /opt/traffic_router/conf directory and generally serve to override Tomcat’s default settings.

For the most part, the configuration files and Parameters used by Traffic Router are used to bring it online and start communicating with various Traffic Control components. Once Traffic Router is successfully communicating with Traffic Control, configuration should mostly be performed in Traffic Portal, and will be distributed throughout Traffic Control via CDN Snapshot process. See Snapshot CRConfig for more information. Please see the Parameter documentation for Traffic Router in the Using Traffic Ops guide documented under Traffic Router Profile for Parameters that influence the behavior of Traffic Router via the Snapshot.

3.1. Administrator’s Guide
<table>
<thead>
<tr>
<th>Configuration File</th>
<th>Parameter Name</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>traffic_monitor.proPERTIES</td>
<td>traffic_monitor.bootstrap.hosts</td>
<td>Semicolon-delimited Traffic Monitor FQDNs with port numbers as necessary</td>
<td>N/A</td>
</tr>
<tr>
<td>traffic_monitor</td>
<td>traffic_monitor.bootstrap.local</td>
<td>Use only the Traffic Monitors specified in local configuration files</td>
<td>false</td>
</tr>
<tr>
<td>traffic_monitor</td>
<td>traffic_monitor.properties</td>
<td>Path to file:traffic_monitor.properties; used internally to monitor the file for changes</td>
<td>/opt/traffic_router/traffic_monitor.properties</td>
</tr>
<tr>
<td>traffic_monitor</td>
<td>traffic_monitor.properties.reload.period</td>
<td>The interval in milliseconds for Traffic Router to wait between reloading this configuration file</td>
<td>60000</td>
</tr>
<tr>
<td>dns.properties</td>
<td>dns.tcp.port</td>
<td>TCP port that Traffic Router will use for incoming DNS requests</td>
<td>53</td>
</tr>
<tr>
<td>dns.properties</td>
<td>dns.tcp.backlog</td>
<td>Maximum length of the queue for incoming TCP connection requests</td>
<td>0</td>
</tr>
<tr>
<td>dns.properties</td>
<td>dns.udp.port</td>
<td>UDP port that Traffic Router will use for incoming DNS requests</td>
<td>53</td>
</tr>
<tr>
<td>dns.properties</td>
<td>dns.max-threads</td>
<td>Maximum number of threads used to process incoming DNS requests</td>
<td>1000</td>
</tr>
<tr>
<td>dns.properties</td>
<td>dns.zones.dir</td>
<td>Path to automatically generated zone files for reference</td>
<td>/opt/traffic_router/var/auto-zones</td>
</tr>
<tr>
<td>traffic_ops.properties</td>
<td>traffic_ops.username</td>
<td>Username with which to access the Traffic Ops API (must have the admin Role)</td>
<td>admin</td>
</tr>
<tr>
<td>traffic_ops.properties</td>
<td>traffic_ops.password</td>
<td>Password for the user specified in traffic_ops.username</td>
<td>N/A</td>
</tr>
<tr>
<td>cache.properties</td>
<td>cache.geolocation.database</td>
<td>The local copy of a geographic IP mapping database (usually MaxMind’s GeoIP2)</td>
<td>/opt/traffic_router/db/GeoIP2-City.mmdb</td>
</tr>
<tr>
<td>cache.health</td>
<td>cache.health.json</td>
<td>The interval in milliseconds Traffic Router to wait between polling for a new health state file</td>
<td>1000</td>
</tr>
<tr>
<td>cache.config.json</td>
<td>cache.config.json</td>
<td>The interval in milliseconds for Traffic Router to wait between polling for a new snapshot file</td>
<td>60000</td>
</tr>
<tr>
<td>cache.config.json</td>
<td>cache.dczmap.database</td>
<td>The local copy of the deep coverage zone file</td>
<td>/opt/traffic_router/db/dczmap.json</td>
</tr>
<tr>
<td>cache.config.json</td>
<td>cache.czmap.database</td>
<td>The local copy of the coverage zone file</td>
<td>/opt/traffic_router/db/czmap.json</td>
</tr>
<tr>
<td>cache.config.json</td>
<td>cache.health.json</td>
<td>Path to the local copy of the health state file</td>
<td>/opt/traffic_router/health.json</td>
</tr>
</tbody>
</table>
DNSSEC

See also:

The Wikipedia page on Domain Name Security Extensions

Overview

DNSSEC is a set of extensions to DNS that provides a cryptographic mechanism for resolvers to verify the authenticity of responses served by an authoritative DNS server. Several RFCs (RFC 4033, RFC 4044, RFC 4045) describe the low level details and define the extensions, RFC 7129 provides clarification around authenticated denial of existence of records, and finally RFC 6781 describes operational best practices for administering an authoritative DNSSEC-enabled DNS server. The authenticated denial of existence RFC 7129 describes how an authoritative DNS server responds in NXDOMAIN and NODATA scenarios when DNSSEC is enabled. Traffic Router currently supports DNSSEC with NSEC (Next Secure Record), however, NSEC3 (Next Secure Record version 3) and more configurable options are planned for the future.

Operation

Upon startup or a configuration change, Traffic Router obtains keys from the ‘keystore’ API in Traffic Ops which returns KSKs and ZSKs for each Delivery Service that is a sub-domain of the CDN’s TLD in addition to the keys for the CDN TLD itself. Each key has timing information that allows Traffic Router to determine key validity (expiration, inception, and effective dates) in addition to the appropriate TTL to use for the DNSKEY record(s). All TTLs are configurable Parameters; see the Traffic Router Profile documentation for more information.

Once Traffic Router obtains the key data from the API, it converts each public key into the appropriate record types (DNSKEY, DS) to place in zones and uses the private key to sign zones. DNSKEY records are added to each Delivery Service’s zone (e.g.: demol.mycdn.ciab.test) for every valid key that exists, in addition to the keys for the CDN TLD itself. A DS record is generated from each zone’s KSK and is placed in the CDN TLD’s zone (e.g.: mycdn.ciab.test); the DS record for the CDN TLD must be placed in its parent zone, which is not managed by Traffic Control.

The DNSKEY to DS record relationship allows resolvers to validate signatures across zone delegation points. With Traffic Control, we control all delegation points below the CDN’s TLD, however, the DS record for the CDN TLD must be placed in the parent zone (e.g.: ciab.test), which is not managed by Traffic Control. As such, the DS record must be placed in the parent zone prior to enabling DNSSEC, and prior to generating a new CDN KSK. Based on your deployment’s DNS configuration, this might be a manual process or it might be automated. Either way, extreme care and diligence must be taken and knowledge of the management of the upstream zone is imperative for a successful DNSSEC deployment.

To enable DNSSEC for a CDN in Traffic Portal, Go to CDNs from the sidebar and click on the desired CDN, then toggle the ‘DNSSEC Enabled’ field to ‘true’, and click on the green Update button to save the changes.
Rolling Zone Signing Keys

Traffic Router currently follows the ZSK pre-publishing operational best practice described in RFC 6781#section-4.1.1.1. Once DNSSEC is enabled for a CDN in Traffic Portal, key rolls are triggered by Traffic Ops via the automated key generation process, and Traffic Router selects the active ZSKs based on the expiration information returned from the ‘keystore’ API of Traffic Ops.

Troubleshooting and Log Files

Traffic Router log files can be found under /opt/traffic_router/var/log and /opt/tomcat/logs. Initialization and shutdown logs are in /opt/tomcat/logs/catalindate.out. Application related logging is in /opt/traffic_router/var/log/traffic_router.log, while access logs are written to /opt/traffic_router/var/log/access.log.

Event Log File Format

Summary

All access events to Traffic Router are logged to the file /opt/traffic_router/var/log/access.log. This file grows up to 200MB and gets rolled into older log files, ten log files total are kept (total of up to 2GB of logged events per Traffic Router instance)

Traffic Router logs access events in a format that largely follows ATS event logging format.

Message Format

- Except for the first item, each event that is logged is a series of space-separated key/value pairs.
- The first item is always the Unix epoch in seconds with a decimal field precision of up to milliseconds.
- Each key/value pair is in the form of unquoted_string="optionally quoted string"
- Values that are quoted strings may contain whitespace characters.
- Values that are not quoted should not contains any whitespace characters.

Note: Any value that is a single dash character or a dash character enclosed in quotes represents an empty value.
Table 36: Fields Always Present

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>qtype</td>
<td>Whether the request was for DNS or HTTP</td>
<td>Always “DNS” or “HTTP”</td>
</tr>
<tr>
<td>chi</td>
<td>The IP address of the requester</td>
<td>Depends on whether this was a DNS or HTTP request, see other sections</td>
</tr>
<tr>
<td>rhi</td>
<td>The IP address of the request source address</td>
<td>Depends on whether this was a DNS or HTTP request, see other sections</td>
</tr>
<tr>
<td>ttms</td>
<td>The amount of time in milliseconds it took Traffic Router to process the request</td>
<td>A number greater than or equal to zero</td>
</tr>
<tr>
<td>rtype</td>
<td>Routing result type</td>
<td>One of ERROR, CZ, DEEP_CZ, GEO, MISS, STATIC_ROUTE, DS_REDIRECT, DS_MISS, INIT, FED</td>
</tr>
<tr>
<td>rloc</td>
<td>GeoLocation of result</td>
<td>Latitude and longitude in degrees as floating point numbers</td>
</tr>
<tr>
<td>rdtl</td>
<td>Result details Associated with unusual conditions</td>
<td>One of DS_NOT_FOUND, DS_NO_BYPASS, DS_BYPASS, DS_CZ_ONLY, DS_CZ_BACKUP_CG</td>
</tr>
<tr>
<td>rerr</td>
<td>Message about an internal Traffic Router error</td>
<td>String</td>
</tr>
</tbody>
</table>

See also:

If Regional Geo-Blocking is enabled on the Delivery Service, an additional field (rgb) will appear.

Sample Message

Items within brackets are detailed under the HTTP and DNS sections

# 29: Example Logfile Lines

```
144140678.000 qtype=DNS chi=192.168.10.11 rhi=- ttms=789 [Fields Specific to the DNS request] rtype=CZ rloc="40.252611,58.439389" rdtl=- rerr="-" [Fields Specific to the DNS result]
144140678.000 qtype=HTTP chi=192.168.10.11 rhi=- ttms=789 [Fields Specific to the HTTP request] rtype=GEO rloc="40.252611,58.439389" rdtl=- rerr="-" [Fields Specific to the HTTP result]
```

Note: These samples contain fields that are always present for every single access event to Traffic Router
### rtype Meanings

- The request was not redirected. This is usually a result of a DNS request to the Traffic Router or an explicit denial for that request

**ANON_BLOCK**  The client’s IP matched an [Anonymous Blocking](#) rule and was blocked

**CZ**  The result was derived from Coverage Zone data based on the address in the chi field

**DEEP_CZ**  The result was derived from Deep Coverage Zone data based on the address in the chi field

**DS_MISS**  *[HTTP Only]*  No HTTP [Delivery Service](#) supports either this request’s URL path or headers

**DS_REDIRECT**  The result is using the Bypass Destination configured for the matched [Delivery Service](#) when that [Delivery Service](#) is unavailable or does not have the requested resource

**ERROR**  An internal error occurred within Traffic Router, more details may be found in the rerr field

**FED**  *[DNS Only]*  The result was obtained through federated coverage zone data outside of any [Delivery Services](#)

**GEO**  The result was derived from geolocation service based on the address in the chi field

**GEO_REDIRECT**  The request was redirected based on the National Geo blocking (Geo Limit Redirect URL) configured on the [Delivery Service](#)

**MISS**  Traffic Router was unable to resolve a DNS request or find a cache for the requested resource

**RGALT**  The request was redirected to the [Regional Geo-Blocking](#) URL. Regional Geo blocking is enabled on the [Delivery Service](#) and is configured through the `regional_geoblock.polling-url` Parameter on the Traffic Router [Profile](#)

**RGDENY**  *[DNS Only]*  The result was obtained through federated coverage zone data outside of any [Delivery Service](#) - the request was regionally blocked because there was no rule for the request made

**STATIC_ROUTE**  *[DNS Only]*  No DNS [Delivery Service](#) supports the hostname portion of the requested URL

### rdtl Meanings

- The request was not redirected. This is usually a result of a DNS request to the Traffic Router or an explicit denial for that request

**DS_BYPASS**  Used a bypass destination for redirection of the [Delivery Service](#)

**DS_CLIENT_GEO_UNSUPPORTED**  Traffic Router did not find a resource supported by coverage zone data and was unable to determine the geographic location of the requesting client
DS_CZ_BACKUP.CG Traffic Router found a backup cache via fall-back (CRconfig’s edgeLocation) or via coordinates (CZF (Coverage Zone File)) configuration

DS_CZ_ONLY The selected Delivery Service only supports resource lookup based on coverage zone data

DS_NO_BYPASS No valid bypass destination is configured for the matched Delivery Service and the Delivery Service does not have the requested resource

DS_NOT_FOUND Always goes with rtypes STATIC_ROUTE and DS_MISS

GEO_NO_CACHE_FOUND Traffic Router could not find a resource via geographic location data based on the requesting client’s location

NODETAILS This entry is for a standard request

REGIONAL_GEO_ALTERNATE_WITHOUT_CACHE This goes with the rtype RGDENY. The URL is being regionally blocked

REGIONAL_GEO_NO_RULE The request was blocked because there was no rule in the Delivery Service for the request

HTTP Specifics

# 30: Sample Message

```
1452197640.936 qtype=HTTP chi=69.241.53.218 rhi=- url="http://foo.mm-test.jenkins.cdnlab.comcast.net/some/asset.m3u8" cqhm=GET cqhv=HTTP/1.1 rtype=GEO rloc="40.252611,58.439389" rdtl=- rerr="-" pssc=302 ttms=0 rurl="http://odol-atsec-sim-114.mm-test.jenkins.cdnlab.comcast.net:8090/some/asset.m3u8" rh="Accept: */*" rh="myheader: asdasdasdasfasg"
```

Table 37: Request Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Requested URL with query string</td>
<td>A URL String</td>
</tr>
<tr>
<td>cqhm</td>
<td>Http Method</td>
<td>e.g. GET, POST</td>
</tr>
<tr>
<td>cqhv</td>
<td>Http Protocol Version</td>
<td>e.g. HTTP/1.1</td>
</tr>
<tr>
<td>rh</td>
<td>One or more of these key value pairs may exist in a logged event and are controlled by the configuration of the matched Delivery Service</td>
<td>Key/value pair of the format name: value</td>
</tr>
</tbody>
</table>

Table 38: Response Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rurl</td>
<td>The resulting URL of the resource requested by the client</td>
</tr>
</tbody>
</table>
### DNS Specifics

#### # 31: Sample Message

```
144140678.000 qtype=DNS chi=192.168.10.11 rhi=- ttms=123 xn=65535, rcdn=www.example.com. type=A class=IN ttl=12345 rcode=NOERROR rloc="40.252611,58.439389" rdtl=- rerr="-" ans="192.168.1.2 192.168.3.4 0:0:0:0:0:ffff:c0a8:102 0:0:0:0:0:ffff:c0a8:304"
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>xn</td>
<td>The ID from the client DNS request header</td>
<td>A whole number between 0 and 65535 (inclusive)</td>
</tr>
<tr>
<td>rhi</td>
<td>The IP address of the resolver when ENDS0 client subnet extensions are enabled.</td>
<td>An IPv4 or IPv6 string, or dash if request is for resolver only and no client subnet is present</td>
</tr>
<tr>
<td>fqdn</td>
<td>The qname field from the client DNS request message (i.e. the FQDN the client is requesting)</td>
<td>A series of DNS labels/domains separated by '.' characters and ending with a '.' character</td>
</tr>
<tr>
<td>type</td>
<td>The qtype field from the client DNS request message (i.e. the type of resolution that’s requested such as IPv4, IPv6)</td>
<td>Examples are A (IPv4), AAAA (IPv6), NS (Name Service), SOA, and CNAME, (see qtype)</td>
</tr>
<tr>
<td>class</td>
<td>The qclass field from the client DNS request message (i.e. the class of resource being requested)</td>
<td>Either IN (Internet resource) or ANY (Traffic Router rejects requests with any other value of class)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttl</td>
<td>The ‘time to live’ in seconds for the answer provided by Traffic Router (clients can reliably use this answer for this long without re-querying traffic router)</td>
<td>A whole number between 0 and 4294967295 (inclusive)</td>
</tr>
<tr>
<td>rcode</td>
<td>The result code for the DNS answer provided by Traffic Router</td>
<td>One of NOERROR (success), NOTIMP (request is not supported), REFUSED (request is refused to be answered), or NXDOMAIN (the domain/name requested does not exist)</td>
</tr>
</tbody>
</table>

### Deep Caching

#### Overview

Deep Caching is a feature that enables clients to be routed to the closest possible “deep” Edge-tier cache server on a per-Delivery Service basis. The term “deep” is used in the networking
sense, meaning that the Edge-tier *cache server s* are located deep in the network where the number of network hops to a client is as minimal. This deep caching topology is desirable because storing content closer to the client gives better bandwidth savings, and sometimes the cost of bandwidth usage in the network outweighs the cost of adding storage. While it may not be feasible to cache an entire copy of the CDN’s contents in every deep location (for the best possible bandwidth savings), storing just a relatively small amount of the CDN’s most requested content can lead to very high bandwidth savings.

**What You Need**

1. Edge cache deployed in “deep” locations and registered in Traffic Ops
2. A DCZF (Deep Coverage Zone File) mapping these deep cache hostnames to specific network prefixes (see *The Deep Coverage Zone File* for details)
3. Deep caching parameters in the Traffic Router Profile (see *Traffic Router Profile* for details):
   - `deepcoveragezone.polling.interval`
   - `deepcoveragezone.polling.url`
4. Deep Caching enabled on one or more HTTP *Delivery Services* (i.e. ‘Deep Caching’ field on the *Delivery Service* details page (under Advanced Options) set to *ALWAYS*)

**How it Works**

Deep Coverage Zone routing is very similar to that of regular Coverage Zone routing, except that the DCZF is preferred over the regular CZF for *Delivery Services* with Deep Caching enabled. If the client requests a Deep Caching-enabled *Delivery Service* and their IP address gets a “hit” in the DCZF, Traffic Router will attempt to route that client to one of the available “deep” *cache server s* in the client’s corresponding zone. If there are no “deep” *cache server s* available for a client’s request, Traffic Router will fall back to the regular CZF and continue regular CZF routing from there.

**Steering Feature**

**Overview**

A Steering *Delivery Service* is a *Delivery Service* that is used to route a client to another *Delivery Service*. The Type of a Steering *Delivery Service* is either STEERING or CLIENT_STEERING. A Steering *Delivery Service* will have target *Delivery Services* configured for it with weights assigned to them. Traffic Router uses the weights to make a consistent hash ring which it then uses to make sure that requests are routed to a target based on the configured weights. This consistent hash ring is separate from the consistent hash ring used in cache selection.
Special regular expressions - referred to as ‘filters’ - can also be configured for target Delivery Services to pin traffic to a specific Delivery Service. For example, if the filter `*/news/.*` for a target called `target-ds-1` is created, any requests to Traffic Router with “news” in them will be routed to `target-ds-1`. This will happen regardless of the configured weights.

**Some other points of interest**

- Steering is currently only available for HTTP Delivery Services that are a part of the same CDN.
- A new role called STEERING has been added to the Traffic Ops database. Only users with the Steering Role or higher can modify steering assignments for a Delivery Service.
- Traffic Router uses the steering endpoints of the Traffic Ops API to poll for steering assignments, the assignments are then used when routing traffic.

A couple simple use-cases for Steering are:

- Migrating traffic from one Delivery Service to another over time.
- Trying out new functionality for a subset of traffic with an experimental Delivery Service.
- Load balancing between Delivery Services

**The Difference Between STEERING and CLIENT_STEERING**

The only difference between the STEERING and CLIENT_STEERING Delivery Service Types is that CLIENT_STEERING explicitly allows a client to bypass Steering by choosing a destination Delivery Service. A client can accomplish this by providing the `X-TC-Steering-Option` HTTP header with a value of the `xml_id` of the target Delivery Service to which they desire to be routed. When Traffic Router receives this header it will route to the requested target Delivery Service regardless of weight configuration. This header is ignored by STEERING Delivery Services.

**Configuration**

The following needs to be completed for Steering to work correctly:

1. Two target Delivery Services are created in Traffic Ops. They must both be HTTP Delivery Services part of the same CDN.
2. A Delivery Service with type STEERING or CLIENT_STEERING is created in Traffic Portal.
3. Target Delivery Services are assigned to the Steering Delivery Service using Traffic Portal.
4. A user with the role of Steering is created.
5. The Steering user assigns weights to the target Delivery Services.
6. If desired, the Steering user can create filters for the target Delivery Services.

See also:
For more information see Configure Delivery Service Steering.

**HTTPS for HTTP Delivery Services**

New in version 1.7: Traffic Router now has the ability to allow HTTPS traffic between itself and clients on a per-HTTP Delivery Service basis.

**Note:** As of version 3.0 Traffic Router has been integrated with native OpenSSL. This makes establishing HTTPS connections to Traffic Router much less expensive than previous versions. However establishing an HTTPS connection is more computationally demanding than an HTTP connection. Since each client will in turn get redirected to an ATS instance, Traffic Router is most always creating a new HTTPS connection for all HTTPS traffic. It is likely to mean that an existing Traffic Router may have some decrease in performance if you wish to support a lot of HTTPS traffic. As noted for DNSSEC, you may need to plan to scale Traffic Router vertically and/or horizontally to handle the new load.

The HTTPS set up process is:

1. Select one of ‘1 - HTTPS’, ‘2 - HTTP AND HTTPS’, or ‘3 - HTTP TO HTTPS’ for the Delivery Service
2. Generate private keys for the Delivery Service using a wildcard domain such as `*.my-delivery-service.my-cdn.example.com`
3. Obtain and import signed certificate chain
4. Perform a CDN Snapshot

Clients may make HTTPS requests to Delivery Services only after the CDN Snapshot propagates to Traffic Router and it receives the certificate chain from Traffic Ops.

**Protocol Options**

**HTTP** Any secure client will get an SSL handshake error. Non-secure clients will experience the same behavior as prior to 1.7

**HTTPS** Traffic Router will only redirect (send a 302 Found response) to clients communicating with a secure connection, all other clients will receive a 503 Service Unavailable response

**HTTP AND HTTPS** Traffic Router will redirect both secure and non-secure clients

**HTTP TO HTTPS** Traffic Router will redirect non-secure clients with a 302 Found response and a location that is secure (i.e. an https:// URL instead of an http:// URL), while secure clients will be redirected immediately to an appropriate target or cache server.
Certificate Retrieval

**Warning:** If you have HTTPS *Delivery Services* in your CDN, Traffic Router will not accept any connections until it is able to fetch certificates from Traffic Ops and load them into memory. Traffic Router does not persist certificates to the Java Keystore or anywhere else.

Traffic Router fetches certificates into memory:
- At startup time
- When it receives a new CDN *Snapshot*
- Once an hour starting whenever the most recent of the last of the above occurred

**Note:** To adjust the frequency at which Traffic Router fetches certificates add the Parameter `certificates.polling.interval` with the ConfigFile “CRConfig.json” and set it to the desired duration in milliseconds.

**Note:** Taking a CDN *Snapshot* may be used at times to avoid waiting the entire polling cycle for a new set of certificates.

**Warning:** If a CDN *Snapshot* is taken that involves a *Delivery Service* missing its certificates, Traffic Router will ignore ALL changes in that CDN *Snapshot* until one of the following occurs:
- It receives certificates for that *Delivery Service*
- Another CDN *Snapshot* is taken and the *Delivery Service* without certificates is changed such that its HTTP protocol is set to ‘http’

Certificate Chain Ordering

The ordering of certificates within the certificate bundle matters. It must be:

1. Primary Certificate (e.g. the one created for *.my-delivery-service.my-cdn.example.com*)
2. Intermediate Certificate(s)
3. Root Certificate from a CA (optional)
Warning: If something is wrong with the certificate chain (e.g. the order of the certificates is backwards or for the wrong domain) the client will get an SSL handshake. Inspection of /opt/tomcat/logs/catalina.log is likely to yield information to reveal this.

To see the ordering of certificates you may have to manually split up your certificate chain and use `openssl(1ssl)` on each individual certificate

**Suggested Way of Setting up an HTTPS Delivery Service**

Assuming you have already created a **Delivery Service** which you plan to modify to use HTTPS, do the following in Traffic Portal:

1. Select one of ‘1 - HTTPS’, ‘2 - HTTP AND HTTPS’, or ‘3 - HTTP TO HTTPS’ for the protocol field of a **Delivery Service** and click the **Update** button
2. Go to **More → Manage SSL Keys**
3. Click on **More → Generate SSL Keys**
4. Fill out the form and click on the green **Generate Keys** button, then confirm that you want to make these changes
5. Copy the contents of the Certificate Signing Request field and save it locally
6. Go back and select ‘HTTP’ for the protocol field of the **Delivery Service** and click **Save** (to avoid preventing other CDN **Snapshot** updates from being blocked by Traffic Router)
7. Follow your standard procedure for obtaining your signed certificate chain from a CA
8. After receiving your certificate chain import it into Traffic Ops
9. Edit the **Delivery Service**
10. Restore your original choice for the protocol field and click **Save**
11. Click **More → Manage SSL Keys**
12. Paste your key information into the appropriate fields
13. Click the green **Update Keys** button
14. Take a new CDN **Snapshot**

Once this is done you should be able to verify that you are being correctly redirected by Traffic Router using e.g. `curl(1)` commands to HTTPS destinations on your **Delivery Service**.

**Router Load Testing**

The Traffic Router load testing tool is located in the Traffic Control repository under `test/router`. It can be used to simulate a mix of HTTP and HTTPS traffic for a CDN by choosing the number of HTTP **Delivery Services** and the number HTTPS **Delivery Service** the test will exercise.

There are 2 parts to the load test:

---

3.1. **Administrator’s Guide** 113
• A web server that makes the actual requests and takes commands to fetch data from the CDN, start the test, and return current results.
• A web page that’s used to run the test and see the results.

Running the Load Tests

1. First, clone the Traffic Control repository.
2. You will need to make sure you have a CA file on your machine.
3. The web server is a Go program, set your GOPATH environment variable appropriately (we suggest $HOME/go or $HOME/src).
4. Open a terminal emulator and navigate to the test/router/server directory inside of the cloned repository.
5. Execute the server binary by running `go run server.go`.
7. Authenticate against a Traffic Ops host - this should be a nearly instantaneous operation - you can watch the output from `server.go` for feedback.
8. Enter the Traffic Ops host in the second form and click the button to get a list of CDN’s.
9. Wait for the web page to show a list of CDN’s under the above form, this may take several seconds.
10. The List of CDN’s will display the number of HTTP- and HTTPS-capable Delivery Services that may be exercised.
11. Choose the CDN you want to exercise from the drop-down menu.
12. Fill out the rest of the form, enter appropriate numbers for each HTTP and HTTPS Delivery Services.
13. Click Run Test.
14. As the test runs the web page will occasionally report results including running time, latency, and throughput.

Tuning Recommendations

The following is an example of the command line parameters set in `/opt/traffic_router/conf/startup.properties` that has been tested on a multi-core server running under HTTPS load test requests. This is following the general recommendation to use the G1 garbage collector for JVM (Java Virtual Machine) applications running on multi-core machines. In addition to using the G1 garbage collector the InitiatingHeapOccupancyPercent was lowered to run garbage collection more frequently which improved overall throughput for Traffic Router and reduced ‘Stop the World’ garbage collection. Note that any environment variable settings in this file will override those set in `/lib/systemd/system/traffic_router.service`. 
# 32: Example CATALINA_OPTS Configuration

```
CATALINA_OPTS="\n-server -Xms2g -Xmx8g \n-Dlog4j.configuration=file://$CATALINA_BASE/conf/log4j.properties \n-Djava.library.path=/usr/lib64 \n-XX:+UnlockExperimentalVMOptions \n-XX:InitiatingHeapOccupancyPercent=30"
```

**Pattern-Based Consistent Hashing Feature**

New in version 3.1: Traffic Router now has the ability to influence consisting hashing using a regular expression on a per-HTTP *Delivery Service* basis.

**Overview**

Pattern-Based Consistent Hashing is a feature to modify the request path given to Traffic Router’s consistent hasher for Cache selection (and *Delivery Service* selection for Steering Delivery Services) using a regular expression. This new regular expression field ‘Consistent Hash Regex’ is applied on a per-Delivery Service basis and is given to Traffic Router via the CDN *Snapshot*. The purpose of this feature is to increase cache efficiency by directing requests for the same asset, but with varying request paths, to the same Cache.

**Note:** Pattern-Based Consistent Hashing is only available for HTTP and Steering Delivery Services

**How it Works**

With Pattern-Based Consistent Hashing, a regular expression (Consistent Hash Regex) is applied to the request path to extract matching elements to build a new string to pass to the consistent hasher. i.e.: using the Consistent Hash Regex `/.*?(/.*?/).*?(m3u8)` given the request paths `/test/path/asset.m3u8` and `/other/path/asset.m3u8` the resulting path to hash will be `/path/m3u8`

**See also:**

See Oracle’s documentation for the `java.util.regex.Pattern` implementation in Java.

**HTTP** When routing requests for an HTTP Delivery Service, Traffic Router compiles a list of eligible caches and uses consistent hashing on the request path to select one. With Pattern-Based Consistent Hashing, the request path is rebuilt prior to consistent hashing for Cache Selection.
Steering For a Steering Delivery Service, Traffic Router uses consistent hashing on the request path to select a Target Delivery Service. In the case of Pattern-Based Consistent Hashing, the request path is rebuilt using the Consistent Hash Regex assigned to the Steering Delivery Service prior to consistent hashing. Then, the Target Delivery Service inherits the Consistent Hash Regex from the Steering Delivery Service and uses it for Cache selection.

Testing Pattern-Based Consistent Hashing

In order to test this feature without affecting the delivery of traffic through a CDN, there are several test tools in place.

Traffic Router API Several Traffic Router endpoints have been added to test regular expression application against a request path, Cache selection, and Delivery Service selection using Pattern-Based Consistent Hashing. For more information see the Traffic Router API documentation.

Traffic Ops API A Traffic Ops endpoint has been added that will proxy request data through to one of the Traffic Router endpoints in order to test regular expression application against a request path. For more information see the Traffic Ops API documentation.

Traffic Portal Test Tool On the Delivery Service (HTTP and Steering) form in Traffic Portal, a “Test Regex” link has been added so that the user can validate a regular expression before saving it to a Delivery Service.

3.1.10 Traffic Router - Migrating to 3.0

Release Notes v3.0

- Replaced custom Java SNI (Server Name Indication) implementation with a native implementation using tomcat-native, APR (Apache Portable Runtime) and OpenSSL. This should significantly improve the performance of routing HTTPS Delivery Services.

See also:
The Server Name Indication Wikipedia page, The Apache Portable Runtime project site and/or the OpenSSL project site

- Upgraded to Tomcat 8.5.30
- Separated the Traffic Router installation from the Tomcat deployment and created a new ‘tomcat’ package for installing Tomcat. Traffic Router and Tomcat can now be upgraded independently
- Converted Traffic Router to a systemd(1) service
- Modified the development test and deployment processes to be more consistent with production

System Requirements

- Centos 7.2
- OpenSSL >= 1.0.2 installed
- JDK >= 8.0 installed or available in an accessible yum(8) repository
- APR >= 1.4.8-3 installed or available in an accessible yum(8) repository
- Tomcat Native >= 1.2.16 installed or available in an accessible yum(8) repository
- tomcat >= 8.5-30 installed or available in an accessible yum(8) repository (This package is created automatically by the Traffic Router build process)

Upgrade Procedure

- upload the dist/tomcat-version string.rpm file generated as a part of the build instructions outlined in Building Traffic Control to an accessible yum(8) repository
- update the traffic_router package with yum(8)
- restore property files

Upload tomcat.rpm

The term-version string.rpm package should have been created when Traffic Router was built according to the instructions in Building Traffic Control. It must either be added to an accessible yum(8) repository, or manually copied to the servers where Traffic Router will be installed. It is generally better that it be added to a yum(8) repository because then it will be installed automatically when Traffic Router is updated.
Update the traffic_router Package

If openssl, apr, tomcat-native, java-1.8.0-openjdk, java-1.8.0-openjdk-devel and tomcat_tr packages are all in an available yum(8) repository then an upgrade can be performed by running yum update traffic_router as the root user or with sudo(8). This will first cause the apr, tomcat-native, java-1.8.0-openjdk, java-1.8.0-openjdk-devel and tomcat packages to be installed. When the tomcat package runs, it will cause any older versions of traffic_router or tomcat to be uninstalled. This is because the previous versions of the traffic_router package included an untracked installation of tomcat.

Restore Property Files

The install process does not override or replace any of the files in the /opt/traffic_router/conf directory. Previous versions of the traffic_ops.properties, traffic_monitor.properties and startup.properties should still be good. On a new install replace the Traffic Router properties files with the correct ones for the CDN.

Development Environment Upgrade

If a development environment is already set up for the previous version of Traffic Router, then openssl, apr and tomcat-native will need to be manually installed with yum(8) or rpm(8). Also, whenever either mvn clean verify or TrafficRouterStart is/are run, the location of the tomcat-native libraries will need to be made known to the JVM via command line arguments.

# 33: Example Commands Specifying a Path to the tomcat-native Library

```
mvn clean verify -Djava.library.path=[tomcat native library path on your box]
java -Djava.library.path=[tomcat native library path on your box]
TrafficRouterStart
```

3.1.11 Traffic Stats Administration

Traffic Stats consists of three separate components: Traffic Stats, InfluxDB, and Grafana. See below for information on installing and configuring each component as well as configuring the integration between the three and Traffic Ops.

Installation
Installing Traffic Stats

- See the downloads page for Traffic Control to get the latest release.
- Follow the instructions in Building Traffic Control to generate an RPM.
- Copy the RPM to your server
- Install the generated Traffic Stats RPM with `yum(8)` or `rpm(8)`

Installing InfluxDB

In order to store Traffic Stats data you will need to install InfluxDB. While not required, it is recommended to use some sort of high availability option like Influx enterprise, InfluxDB Relay, or another high availability option.

Installing Grafana

Grafana is used to display Traffic Stats/InfluxDB data in Traffic Ops. Grafana is typically run on the same server as Traffic Stats but this is not a requirement. Grafana can be installed on any server that can access InfluxDB and can be accessed by Traffic Ops. Documentation on installing Grafana can be found on the Grafana website.

Configuration

Configuring Traffic Stats

Traffic Stats’s configuration file can be found in `/opt/traffic_stats/conf/traffic_stats.cfg`. The following values need to be configured:

- **toUser** - The user used to connect to Traffic Ops
- **toPasswd** - The password to use when connecting to Traffic Ops
- **toUrl** - The URL of the Traffic Ops server used by Traffic Stats
- **influxUser** - The user to use when connecting to InfluxDB (if configured on InfluxDB, else leave default)
- **influxPassword** - That password to use when connecting to InfluxDB (if configured, else leave blank)
- **pollingInterval** - The interval at which Traffic Monitor is polled and stats are stored in InfluxDB
- **statusToMon** - The status of Traffic Monitor to poll (poll ONLINE or OFFLINE traffic monitors)
- **seelogConfig** - The absolute path of the seelog configuration file
- **dailySummaryPollingInterval** - The interval, in seconds, at which Traffic Stats checks to see if daily stats need to be computed and stored.
cacheRetentionPolicy  The default retention policy for cache stats

dsRetentionPolicy  The default retention policy for Delivery Service stats

dailySummaryRetentionPolicy  The retention policy to be used for the daily stats

influxUrls  An array of InfluxDB hosts for Traffic Stats to write stats to.

**Configuring InfluxDB**

As mentioned above, it is recommended that InfluxDB be running in some sort of high availability configuration. There are several ways to achieve high availability so it is best to consult the high availability options on the InfluxDB website.

Once InfluxDB is installed and configured, databases and retention policies need to be created. Traffic Stats writes to three different databases: cache_stats, deliveryservice_stats, and daily_stats. More information about the databases and what data is stored in each can be found in the Traffic Stats Overview.

To easily create databases, retention policies, and continuous queries, run `create_ts_databases` from the `/opt/traffic_stats/influxdb_tools` directory on your Traffic Stats server. See the InfluxDB Tools section for more information.

**Configuring Grafana**

In Traffic Portal the Other → Grafana menu item can be configured to display Grafana graphs using InfluxDB data. In order for this to work correctly, you will need two things:

1. A Parameter with the graph URL (more information below)
2. The graphs created in Grafana. See below for how to create some simple graphs in Grafana. These instructions assume that InfluxDB has been configured and that data has been written to it. If this is not true, you will not see any graphs.

To create a graph in Grafana, you can follow these basic steps:

1. Login to Grafana as an administrative user
2. Click on Data Sources → Add New
3. Enter the necessary information to configure your data source
4. Click on Home → New at the bottom
5. Click on “Collapsed Menu Icon” Button → Add Panel → Graph
6. Where it says No Title (click here) click and choose edit
7. Choose your data source at the bottom
8. You can have Grafana help you create a query, or you can create your own.
9. Once you have the graph the way you want it, click the *Save Dashboard* button at the top

10. You should now have a new saved graph

In order for Traffic Portal users to see Grafana graphs, Grafana will need to allow anonymous access. Information on how to configure anonymous access can be found on the configuration page of the Grafana Website.

Traffic Portal uses custom dashboards to display information about individual *Delivery Services* or *Cache Groups*. In order for the custom graphs to display correctly, the `traffic_ops_*.js` files need to be in the `/usr/share/grafana/public/dashboards/` directory on the Grafana server. If your Grafana server is the same as your Traffic Stats server the RPM install process will take care of putting the files in place. If your Grafana server is different from your Traffic Stats server, you will need to manually copy the files to the correct directory.

More information on custom scripted graphs can be found in the *scripted dashboards* section of the Grafana documentation.

### Configuring Traffic Portal for Traffic Stats

- The InfluxDB servers need to be added to Traffic Portal with profile = InfluxDB. Make sure to use port 8086 in the configuration.
- The traffic stats server should be added to Traffic Ops with profile = Traffic Stats.
- *Parameters* for which stats will be collected are added with the release, but any changes can be made via parameters that are assigned to the Traffic Stats profile.

### Configuring Traffic Portal to use Grafana Dashboards

To configure Traffic Portal to use Grafana Dashboards, you need to enter the following *Parameters* and assign them to the GLOBAL profile. This assumes you followed the above instructions to install and configure InfluxDB and Grafana. You will need to place ‘cdn-stats’, ‘deliveryservice-stats’, and ‘daily-summary’ with the name of your dashboards.
### Table 41: Traffic Stats Parameters

<table>
<thead>
<tr>
<th>parameter name</th>
<th>parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_graph_url</td>
<td>https://&lt;grafanaHost&gt;/dashboard/db/deliveryservice-stats</td>
</tr>
<tr>
<td>cachegroup_graph_url</td>
<td>https://&lt;grafanaHost&gt;/dashboard/script/traffic_ops_cachegroup.js?which=</td>
</tr>
<tr>
<td>deliveryservice_graph_url</td>
<td>https://&lt;grafanaHost&gt;/dashboard/script/traffic_ops_deliveryservice.js?which=</td>
</tr>
<tr>
<td>server_graph_url</td>
<td>https://&lt;grafanaHost&gt;/dashboard/script/traffic_ops_server.js?which=</td>
</tr>
<tr>
<td>visual_status_panel_1</td>
<td>https://&lt;grafanaHost&gt;/dashboard-solo/db/cdn-stats?panelId=2&amp;fullscreen&amp;from=now-24h&amp;to=now-60s</td>
</tr>
<tr>
<td>visual_status_panel_2</td>
<td>https://&lt;grafanaHost&gt;/dashboard-solo/db/cdn-stats?panelId=1&amp;fullscreen&amp;from=now-24h&amp;to=now-60s</td>
</tr>
<tr>
<td>daily_bw_url</td>
<td>https://&lt;grafanaHost&gt;/dashboard-solo/db/daily-summary?panelId=1&amp;fullscreen&amp;from=now-3y&amp;to=now</td>
</tr>
<tr>
<td>daily_served_url</td>
<td>https://&lt;grafanaHost&gt;/dashboard-solo/db/daily-summary?panelId=2&amp;fullscreen&amp;from=now-3y&amp;to=now</td>
</tr>
</tbody>
</table>

### InfluxDB Tools

Under the Traffic Stats source directory there is a directory called `influxdb_tools`. These tools are meant to be used as one-off scripts to help a user quickly get new databases and continuous queries setup in InfluxDB. They are specific for Traffic Stats and are not meant to be generic to InfluxDB. Below is an brief description of each script along with how to use it.

**create/create_ts_databases.go**

This program creates all databases, retention policies, and continuous queries required by Traffic Stats.

**Pre-Requisites**

- Go 1.7 or later
- Configured `$GOPATH` environment variable

**Options and Arguments**

---help

(Optional) Print usage information and exit (with a failure exit code for some reason)
--password password
  The password that will be used by the user defined by --user to authenticate.

--replication N
  (Optional) The number of nodes in the cluster (default: 3)

--url URL
  The InfluxDB server's root URL - including port number, if required (default: http://localhost:8086)

--user username
  The name of the user to use when connecting to InfluxDB

sync/sync_ts_databases.go

This program is used to sync one InfluxDB environment to another. Only data from continuous queries is synced as it is down-sampled data and much smaller in size than syncing raw data. Possible use cases are syncing from production to development or syncing a new cluster once brought online.

Pre-Requisites

- Go 1.7 or later
- Configured $GOPATH environment variable

Options and Arguments

--database database_name
  (Optional) Specify the name of a specific database to sync (default: all databases)

--days N
  The number of days in the past to sync. 0 means ‘all’

--help
  (Optional) Print usage information and exit

--source-password password
  The password of the user named by --source-user

--source-url URL
  (Optional) The URL of the InfluxDB instance _from_ which data will be copied (default: http://localhost:8086)

--source-user username
  The name of the user as whom the utility will connect to the source InfluxDB instance

--target-password password
  The password of the user named by --target-user
--target-url URL
(Optional) The URL of the InfluxDB instance _to_ which data will be copied (default: http://localhost:8086)

--target-user username
The name of the user as whom the utility will connect to the target InfluxDB instance

### 3.1.12 Traffic Server Administration

#### Installing Traffic Server

1. Build the Traffic Server RPM. The best way to do this is to follow the Apache Traffic Server documentation.

2. Build the astats RPM using the appropriate version number - ours are built here

   **Note:** The astats plugin is bundled as a part of Apache Traffic Server as of version 7.2.

3. Install Traffic Server and astats

   ```
   # 35: Apache Traffic Server Installation Using `yum(8)`
   ```
   ```shell
   yum -y install trafficserver-*.rpm astats_over_http*.rpm
   ```

4. Add the server using the Traffic Portal UI:

   1. Go to `Configure → Servers`
   2. Click on the + button at the top of the page.
   3. Complete the form. Be sure to fill out all fields marked ‘Required’
      - Set ‘Interface Name’ to the name of the network interface device from which Apache Traffic Server delivers content.
      - Set ‘Type’ to ‘MID’ or ‘EDGE’.
      - If you wish for the server to immediately be polled by the Health Protocol, set ‘Status’ to ‘REPORTED’.
   4. Click on the `Create` button to submit the form.
   5. Verify that the server status is now listed as `Reported`

5. Install the `ORT` script and run it in ‘BADASS’ mode to create the initial configuration

   **See also:**

   *Configuring Traffic Server*

6. Start ATS
# 36: Starting ATS with `systemd(1)`

```
systemctl start trafficserver
```

7. (Optional) Configure ATS to start automatically when the system powers on

```
# 37: Configuring ATS to Start Automatically Using `systemd(1)`

```

```
systemctl enable trafficserver
```

8. Verify that the installation is working

1. Make sure that the service is running

```
# 38: Checking that ATS is Running Using `systemd(1)`

```

```
systemctl status trafficserver
```

2. Assuming a Traffic Monitor is already installed somewhere, check the “Cache States” table in its Web UI to verify that the ATS server appears.

## Configuring Traffic Server

All of the ATS application configuration files are generated by Traffic Ops and installed by ORT. The `traffic_ops_ort.pl` file should be installed on all cache servers (See Installing the ORT Script), usually in `/opt/ort`. It is used to do the initial install of the configuration files when the cache server is being deployed, and to keep the configuration files up-to-date when the cache server is already in service.

### Installing the ORT Script

1. Build the ORT script RPM by following the instructions in Building Traffic Control and install it with `rpm(8)` or `yum(8)`.

2. Install modules required by ORT if needed

```
# 39: Example Installation of Perl Packages Occasionally Missing from Install

```

```
yum install -y perl-JSON perl-Crypt-SSLeay
```

3. For initial configuration or when major changes (like a Profile change) need to be made, run the script in “BADASS”. All required RPM packages will be installed, all ATS configuration files will be fetched and installed, and (if needed) the ATS service will be restarted.

**Note:** The first run gives a lot of state errors that are expected. The
“BADASS” mode fixes these issues. If you run it a second time, this should be cleaner. Also, note that many “ERROR” messages emitted by ORT are actually information messages. Do not panic.

4. Create a `cron(8)` entry for running ORT in “SYNCDS” mode every 15 minutes. This makes Traffic Control check periodically if the server has updates pending, and if so get the updated configuration.

**Note:** By default, running ORT on an Edge-tier cache server will cause it to first wait for its parents (usually Mid-tier cache servers) to download their configuration before downloading its own configuration. Because of this, scheduling ORT for running every 15 minutes (with 5 minutes default dispersion) means that it might take up to ~35 minutes for queued updates to affect all cache servers. To customize this dispersion time, use the command line option `--dispersion=x` where x is the number of seconds for the dispersion period. Servers will select a random number from within this dispersion period to being downloading configuration files from Traffic Ops. Another option, `--login_dispersion=x` can be used to create a dispersion period after the job begins during which ORT will wait before logging in and checking Traffic Ops for updates to the server. This defaults to 0. If `use_reval_pending`, a.k.a. “Rapid Revalidate” is enabled, Edge-tier cache servers will not wait for their parents to download their configuration before downloading their own.

**Note:** In “SYNCDS” mode, the ORT script updates only configurations that might be changed as part of normal operations, such as:

- *Delivery Services*
- SSL certificates
- Traffic Monitor IP addresses
- Logging configuration
- Revalidation requests (By default - if “Rapid Revalidate” is enabled, this will only be checked by using a separate revalidate command in ORT)

5. If “Rapid Revalidate” is enabled in Traffic Ops, create a second `cron(8)` job for revalidation checks by running ORT in “REVALIDATE” mode. ORT will not check revalidation files if “Rapid Revalidate” is enabled. This setting allows for a separate check to be performed every 60 seconds to verify if a revalidation update has been made.

### 3.1.13 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. Based on experience, version 2.0.5 of Riak is recommended, but the latest version should suffice.

Configuring Traffic Vault

The following steps were taken to configure Riak in Comcast production environments.

Self Signed Certificate configuration

**Note:** Self-signed certificates are not recommended for production use. Intended for development or learning purposes only. Modify subject as necessary.

```
# 40: Self-Signed Certificate Configuration

cd ~
mkdir certs
cd certs
openssl genrsa -out ca-bundle.key 2048
openssl req -new -key ca-bundle.key -out ca-bundle.csr -subj "/C=US/ST=CO/L=DEN/O=somecompany/OU=CDN/CN=somecompany.net/
  emailAddress=someuser@somecompany.net"
openssl x509 -req -days 365 -in ca-bundle.csr -signkey ca-bundle.key -out ca-bundle.crt
openssl genrsa -out server.key 2048
openssl req -new -key server.key -out server.csr -subj "/C=US/ST=CO/L=DEN/O=somecompany/OU=CDN/CN=somecompany.net/
  emailAddress=someuser@somecompany.net"
openssl x509 -req -days 365 -in server.csr -CA ca-bundle.crt -CAkey ca-bundle.key -CAcreateserial -out server.crt
mkdir /etc/riak/certs
mv -f server.crt /etc/riak/certs/.
mv -f server.key /etc/riak/certs/.
mv -f ca-bundle.crt /etc/pki/tls/certs/.
```

Riak Configuration File

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

1. Log into Riak server as root
2. Update the following in riak.conf to reflect your IP, hostname, and CDN domains and sub-domains:
• nodename = riak@a-host.sys.kabletown.net
• listener.http.internal = a-host.sys.kabletown.net:8098 (port can be 80 - This endpoint will not work over HTTPS)
• 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 (port can be 443)

3. Update the following in riak.conf file to point to your SSL certificate files
   • ssl.certfile = /etc/riak/certs/server.crt
   • ssl.keyfile = /etc/riak/certs/server.key
   • ssl.cacertfile = /etc/pki/tls/certs/ca-bundle.crt

4. Add a line at the bottom of the riak.conf for TLSv1 by setting tls_protocols. tlsv1 = on

5. Once the configuration file has been updated restart Riak

6. Consult the Riak documentation for instructions on how to verify the installed service

**riak-admin Configuration**

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

**See also:**

The riak-admin documentation

```
# 41: Traffic Vault Setup with riak-admin

# This script need only be run on any *one* Riak server in the cluster

# Enable security and secure access groups
riak-admin security enable
riak-admin security add-group admins
riak-admin security add-group keysusers

# User name and password should be stored in
# /opt/traffic_ops/app/conf/<environment>/riak.conf on the Traffic Ops server
# In this example, we assume the usernames 'admin' and 'riakuser' with
# respective passwords stored in the ADMIN_PASSWORD and RIAK_USER_PASSWORD environment variables
```

(continues on next page)
riak-admin security add-user admin password=$ADMIN_PASSWORD
  → groups=admins
riak-admin security add-user riakuser password=$RIAK_USER_PASSWORD
  → groups=keysusers
riak-admin security add-source riakuser 0.0.0.0/0 password
riak-admin security add-source admin 0.0.0.0/0 password

# Grant privileges to the admins group 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

# Grant privileges to keysusers group 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
riak-admin security grant riak_kv.get,riak_kv.put,riak_kv.delete on
  → default cdn_uri_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

Before a fully set-up Traffic Vault instance may be used, it must be added as a server to Traffic Ops. The easiest way to accomplish this is via Traffic Portal at Configure → Servers, though servers may also be used by low-level tools and/or scripts. The Traffic Ops configuration file /opt/traffic_ops/app/conf/environment/riak.conf for the appropriate environment must also be updated to reflect the correct username and password for accessing the Riak database.

Configuring Riak Search

In order to more effectively support retrieval of SSL certificates by Traffic Router and ORT, Traffic Vault uses Riak search. Riak Search uses Apache Solr for indexing and searching of records. This section explains how to enable, configure, and validate Riak Search.
Riak Configuration

On each Traffic Vault server follow these steps.

1. If Java (JDKv1.8+) is not already installed on your Riak server, install Java

   # 42: Check if Java is Installed, Then Install if Needed

   ```
   # Ensure that this outputs a Java version that is at least 1.8
   java -version

   # If it didn't, or produced an error because `java` doesn't exist,
   # install the correct version
   # (OpenJDK is used here because of its permissive license, though OracleJDK
   # should work with some tinkering)

   # On CentOS/RedHat/Fedora (recommended)
   yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel

   # On Ubuntu/Debian/Linux Mint
   apt install -y openjdk-8-jdk

   # Arch/Manjaro
   pacman -Sy jdk8-openjdk
   ```

2. Enable search in riak.conf by changing the search = off setting to search = on

3. Restart Riak to propagate configuration changes

   # 43: Restarting Riak on systemd(1) Systems

   ```
   systemctl restart riak
   ```

One-time Configuration

After Riak has been configured to use Riak Search, permissions still need to be updated to allow users to utilize this feature. Unlike actually setting up Riak Search, the permissions step need only be done on any one of the Riak servers in the cluster.

1. Use riak-admin to grant search.admin permissions to the “admin” user and search.query permissions to both the “admin” user and the “riakuser” user. The “admin” user will also require search.admin permissions on the schema (in addition to index) and riak_core.set_bucket permissions on any.
# 44: Setting up Riak Search Permissions

```
riak-admin security grant search.admin on schema to admin
riak-admin security grant search.admin on index to admin
riak-admin security grant search.query on index to admin
riak-admin security grant search.query on index sslkeys to admin
riak-admin security grant search.query on index to riakuser
riak-admin security grant search.query on index sslkeys to riakuser
riak-admin security grant riak_core.set_bucket on any to admin
```

2. Add the search schema to Riak. This schema is a simple Apache Solr configuration file which will index all records on CDN, hostname, and Delivery Service. The file can be found at `traffic_ops/app/config/misc/riak_search/sslkeys.xml` in the Traffic Control repository.

# 45: Adding the GitHub-hosted Search Schema to Riak

```
# Obtain the configuration file - in this example by downloading it from GitHub
wget https://raw.githubusercontent.com/apache/trafficcontrol/master/traffic_ops/app/conf/misc/riak_search/sslkeys.xml

# Upload the schema to the Riak server using its API
# Note that the assumptions made here are that the "admin" user's password is "pass"
# and the server is accessible at port 8088 on the hostname "trafficvault.infra.ciab.test"

curl -ksX PUT "https://admin:pass@trafficvault.infra.ciab.test:8088/search/schema/sslkeys" -H "Content-Type: application/xml" -d @sslkeys.xml
```

3. Add the search index to Riak.

# 46: Adding the Search Index to Riak Via its API

```
# Note that the assumptions made here are that the "admin" user's password is "pass"
# and the server is accessible at port 8088 on the hostname "trafficvault.infra.ciab.test"
curl -ksX PUT "https://admin:pass@trafficvault.infra.ciab.test:8088/search/index/sslkeys" -H "Content-Type: application/json" -d '{"schema": "sslkeys"}'
```

4. Associate the `sslkeys` index to the `ssl` bucket in Riak
# 47: Using the Riak API to Create an Index-to-Bucket Association for sslkeys

```bash
curl -kvs -XPUT "https://admin:pass@trafficvault.infra.ciab.test:8088/buckets/ssl/props" -H 'content-type: application/json' -d '{"props":{"search_index": "sslkeys"}}'
```

Adding Newly Indexed Fields to Existing Records

Riak Search (using Apache Solr) will now index all new records that are added to the ssl bucket. The cdn, deliveryservice, and hostname fields are indexed. When a search is performed Riak will return the indexed fields along with the certificate and key values for a SSL record. In order to add the indexed fields to current records and to get the current records added, the `traffic_ops/app/script/update_riak_for_search.pl` script needs to be run. This does not need to be done on new installs. The following explains how to run the script.

# 48: Example Usage of traffic_ops/app/script/update_riak_for_search.pl

```bash
#/update_riak_for_search.pl -to_url=https://trafficops.infra.ciab.test -to_un=admin -to_pw="twelve!"
```

To validate the search is working run a query against the Riak database server, or use the Traffic Ops API endpoint: `cdns/name/{[name]}/sslkeys`

# 49: Validate Riak Search is Working

```bash
#/validate Riak Search is Working
```

(continues on next page)
# the hostname "trafficvault.infra.ciab.test", $COOKIE contains a valid Mojolicious cookie for a Traffic Ops user with proper permissions, and the Traffic Ops server is available at the hostname "trafficops.infra.ciab.test"

# Verify by querying Riak directly
curl -kvs "https://admin:password@trafficvault.infra.ciab.test:8088/search/query/sslkeys?wt=json&q=cdn:CDN-in-a-Box"

# Verify using the Traffic Ops API

3.1.14 Quick How To Guides

Traffic Control is a complicated system, and documenting it is not trivial. Sometimes a picture says more than a thousand words, so here are some screenshot-based tutorials on how to use some of the more involved features.

Configure Anonymous Blocking

**Note:** Anonymous Blocking is only supported for HTTP delivery services. You will need access to a database that provides anonymous IP statistics (Maxmind’s database is recommended, as this functionality was built specifically to work with it.)

1. Prepare the Anonymous Blocking configuration file. Anonymous Blocking uses a configuration file in JSON format to define blocking rules for Delivery Services. The file needs to be put on an HTTP server accessible to Traffic Router.

```
# 50: Example Configuration JSON
{
    "customer": "YourCompany",
    "version": "1",
    "date": "2017-05-23 03:28:25",
    "name": "Anonymous IP Blocking Policy",
    "anonymousIp": {
        "blockAnonymousVPN": true,
        "blockHostingProvider": true,
        "blockPublicProxy": true,
        "blockTorExitNode": true
    },
    "ip4Whitelist": ["192.168.30.0/24", "10.0.2.0/24", "10.1.1.1/32"],
}
```

(continues on next page)
anonymousIp Contains the types of IPs which can be checked against the Anonymous IP Database. There are 4 types of IPs which can be checked: VPN (Virtual Private Network)s, Hosting Providers, Public Proxies, and TOR “Exit Nodes”. Each type of IP can be enabled or disabled. If the value is true, IPs matching this type will be blocked when the feature is enabled in the Delivery Service. If the value is false, IPs which match this type will not be blocked. If an IP matches more than 1 type and any type is enabled, the IP will be blocked.

redirectUrl The URL that will be returned to the blocked clients. Without a redirectUrl, the clients will receive an HTTP response code 403 Forbidden. With a redirectUrl, the clients will be redirected with an HTTP response code 302 Found.

ipWhiteList An optional element. It includes a list of CIDR blocks indicating the IPv4 and IPv6 subnets that are allowed by the rule. If this list exists and the value is not null, client IPs will be matched against the CIDR list, and if there is any match, the request will be allowed. If there is no match in the white list, further anonymous blocking logic will continue.

2. Add the following three Anonymous Blocking parameters in Traffic Portal into CRConfig.json:

anonymousIp.policy.configuration The HTTP URL of the Anonymous Blocking configuration file. Traffic Router will fetch the file from this URL.

anonymousIp.polling.url The HTTP URL of the Anonymous IP Database. Traffic Router will fetch the file from this URL.

anonymousIp.polling.interval The interval that Traffic Router polls the Anonymous Blocking configuration file and Anonymous IP Database.

3. Enable Anonymous Blocking for a Delivery Service
4. Go to Tools → Snapshot CRConfig, perform Diff CRConfig and click Write CRConfig.

Traffic Router Access Log

Anonymous Blocking extends the field of `rtype` and adds a new field `ANON_BLOCK` in the Traffic Router access.log file to help monitor this feature. If the `rtype` in an access log is `ANON_BLOCK` then the client’s IP was found in the Anonymous IP Database and was blocked.

See also:

*Troubleshooting and Log Files*

Configure Cache Group Fallbacks

See also:

*Cache Groups*

1. Go to ‘Topology’, click on Cache Groups, and click on your desired Cache Group or click the + button to create a new Cache Group.
2. Verify that the Cache Group is of type EDGE_LOC. Cache Group Failovers only apply to EDGE_LOC Cache Groups.

![Fig. 12: Cache Group Details Page](image)

3. Once EDGE_LOC is selected, the Failover Cache Groups section will appear at the bottom of the page. If you are editing an existing Cache Group, then the current Failovers will be listed. If creating a new Cache Group, the Fallback to Geo Failover box will default to be checked.

![Fig. 13: Failover Cache Groups Section of Cache Group Details Page](image)

4. To add a new Failover to the list, select the “Add Failover Cache Group” drop down and choose which Cache Group you would like. While in the drop down, you can also type in order to search.

![Fig. 14: Add New Failover Cache Group Section of Cache Group Details Page](image)

5. The order of the Failovers is important. If you want to reorder the Failovers, you can drag and drop them into a new position. A red line will appear to show where the Failover will be dropped.

6. To remove a Failover, click the trash can symbol on the right hand side of the list.

7. Click the Update button (if editing existing Cache Group) or the Create button (if creating new Cache Group) in order to save the Failovers to the Cache Group.
“CDN in a Box” is a name given to the time-honored tradition of new Traffic Control developers/potential users attempting to set up their own, miniature CDN to just see how it all fits together. Historically, this has been a nightmare of digging through leftover `virsh` scripts and manually configuring pretty hefty networking changes (don’t even get me started on DNS) and just generally having a bad time. For a few years now, different people had made it to various stages of merging the project into Docker for ease of networking, but certain constraints hampered progress - until now. The project has finally reached a working state, and now getting a mock/test CDN running can be a very simple task (albeit rather time-consuming).

**Getting Started**

Because it runs in Docker, the only true prerequisites are:

- Docker version $\geq$ 17.05.0-ce
- Docker Compose\(^1\) version $\geq$ 1.9.0

**Building**

The CDN in a Box directory is found within the Traffic Control repository at `infrastructure/cdn-in-a-box/`. CDN in a Box relies on the presence of pre-built `component.rpm` files for the following Traffic Control components:

- **Traffic Monitor** - at `infrastructure/cdn-in-a-box/traffic_monitor/traffic_monitor.rpm`
- **Traffic Ops** - at `infrastructure/cdn-in-a-box/traffic_ops/traffic_ops.rpm`
- **Traffic Portal** - at `infrastructure/cdn-in-a-box/traffic_portal/traffic_portal.rpm`

\(^1\) It is perfectly possible to build and run all containers without Docker Compose, but it’s not recommended and not covered in this guide.
• Traffic Router - at infrastructure/cdn-in-a-box/traffic_router/traffic_router.rpm - also requires an Apache Tomcat RPM at infrastructure/cdn-in-a-box/traffic_router/tomcat.rpm

Note: These can also be specified via the RPM variable to a direct Docker build of the component - with the exception of Traffic Router, which instead accepts JDK8_RPM to specify a Java Development Kit RPM, TRAFFIC_ROUTER_RPM to specify a Traffic Router RPM, and TOMCAT_RPM to specify an Apache Tomcat RPM.

These can all be supplied manually via the steps in Building Traffic Control (for Traffic Control component RPMs) or via some external source. Alternatively, the infrastructure/cdn-in-a-box/Makefile file contains recipes to build all of these - simply run make(1)² from the infrastructure/cdn-in-a-box/ directory. Once all RPM dependencies have been satisfied, run docker-compose build from the infrastructure/cdn-in-a-box/ directory to construct the images needed to run CDN in a Box.

Usage

In a typical scenario, if the steps in Building have been followed, all that’s required to start the CDN in a Box is to run docker-compose up - optionally with the -d flag to run without binding to the terminal - from the infrastructure/cdn-in-a-box/ directory. This will start up the entire stack and should take care of any needed initial configuration. The services within the containers are exposed locally to the host on specific ports. These are configured within the infrastructure/cdn-in-a-box/docker-compose.yml file, but the default ports are shown in Service Info. Some services have credentials associated, which are totally configurable in variables.env.

² Consider make -j to build quickly, if your computer can handle multiple builds at once.
Table 42: Service Info

<table>
<thead>
<tr>
<th>Service</th>
<th>Ports exposed and their usage</th>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>DNS name resolution on 9353</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Edge Tier Cache</td>
<td>Apache Trafficserver HTTP caching reverse proxy on port 9000</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mid Tier Cache</td>
<td>Apache Trafficserver HTTP caching forward proxy on port 9100</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mock Origin Server</td>
<td>Example web page served on port 9200</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Traffic Monitor</td>
<td>Web interface and API on port 80</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Traffic Ops</td>
<td>Main API endpoints on port 6443, with a direct route to the Perl API on port 60443(^3)</td>
<td>TO_ADMIN_USER (\text{in variables.env})</td>
<td>TO_ADMIN_PASSWORD (\text{in variables.env})</td>
</tr>
<tr>
<td>Traffic Ops PostgreSQL Database</td>
<td>PostgreSQL connections accepted on port 5432 (database name: \text{DB_NAME} (\text{in variables.env}))</td>
<td>DB_USER (\text{in variables.env})</td>
<td>DB_USER_PASS (\text{in variables.env})</td>
</tr>
<tr>
<td>Traffic Portal</td>
<td>Web interface on 443 (Javascript required)</td>
<td>TO_ADMIN_USER (\text{in variables.env})</td>
<td>TO_ADMIN_PASSWORD (\text{in variables.env})</td>
</tr>
<tr>
<td>Traffic Router</td>
<td>Web interfaces on ports 3080 (HTTP) and 3443 (HTTPS), with a DNS service on 53 and an API on 3333</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Traffic Vault</td>
<td>Riak key-value store on port 8010</td>
<td>TV_ADMIN_USER (\text{in variables.env})</td>
<td>TV_ADMIN_PASSWORD (\text{in variables.env})</td>
</tr>
<tr>
<td>Traffic Stats</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Traffic Stats Influxdb</td>
<td>Influxdbd connections accepted on port 8086 (database name: \text{cache_stats, daily_stats and deliveryservice_stats})</td>
<td>INFLUXDB_ADMIN_USER (\text{in variables.env})</td>
<td>INFLUXDB_ADMIN_PASSWORD (\text{in variables.env})</td>
</tr>
</tbody>
</table>

See also:

Traffic Router API and Traffic Monitor APIs

While the components may be interacted with by the host using these ports, the true operation of the CDN can only truly be seen from within the Docker network. To see the CDN in action, connect to a container within the CDN in a Box project and use cURL to request the URL http://video.demo1.mycdn.ciab.test which will be resolved by the DNS container to the IP of the Traffic Router, which will provide a 302 FOUND response pointing to the Edge-Tier cache. A typical choice for this is the “enroller” service, which has a very nuanced purpose not discussed here but already has the \texttt{curl(1)} command line tool installed.

\(^3\) Please do NOT use the Perl endpoints directly. The CDN will only work properly if everything hits the Go API, which will proxy to the Perl endpoints as needed.
For a more user-friendly interface into the CDN network, see VNC Server.

### # 51: Example Command to See the CDN in Action

```bash
sudo docker-compose exec enroller /usr/bin/curl -L "http://video.demo1.mycdn.ciab.test"
```

When the CDN is to be shut down, it is often best to do so using `sudo docker-compose down -v` due to the use of shared volumes in the system which might interfere with a proper initialization upon the next run.

#### variables.env

```
TLD_DOMAIN=ciab.test
INFRA_SUBDOMAIN=infra
CDN_NAME=CDN-in-a-Box
CDN_SUBDOMAIN=mycdn
DS_HOSTS=demo1 demo2 demo3
X509_CA_NAME=CIAB-CA
X509_CA_COUNTRY=US
X509_CA_STATE=Colorado
X509_CA_CITY=Denver
X509_CA_COMPANY=NotComcast
X509_CA_ORG=CDN-in-a-Box
X509_CA_ORGUNIT=CDN-in-a-Box
X509_CA_EMAIL=no-reply@infra.ciab.test
X509_CA_DIGEST=sha256
X509_CA_DURATION_DAYS=365
X509_CA_KEYTYPE=rsa
X509_CA_KEYSIZE=4096
X509_CA_UMASK=0000
X509_CA_DIR=/shared/ssl
X509_CA_PERSIST_DIR=/ca
X509_CA_PERSIST_ENV_FILE=/ca/environment
X509_CA_ENV_FILE=/shared/ssl/environment
DB_NAME=traffic_ops
DB_PORT=5432
DB_SERVER=db
DB_USER=traffic_ops
DB_USER_PASS=twelve
DNS_SERVER=dns
DBIC_TRACE=0
ENROLLER_HOST=enroller
PGPASSWORD=twelve
POSTGRES_PASSWORD=twelve
EDGE_HOST=edge
INFLUXDB_HOST=influxdb
INFLUXDB_PORT=8086
INFLUXDB_ADMIN_USER=influxadmin
```
INFLUXDB_ADMIN_PASSWORD=influxadminpassword
GRAFANA_ADMIN_USER=grafanaadmin
GRAFANA_ADMIN_PASSWORD=grafanaadminpassword
GRAFANA_PORT=443
MID_HOST=mid
ORIGIN_HOST=origin
TM_HOST=trafficmonitor
TM_PORT=80
TM_EMAIL=tmonitor@cdn.example.com
TM_PASSWORD=jhds1vhdfsuklvfhsu1hs
TM_USER=tmon
TO_ADMIN_PASSWORD=twelve
TO_ADMIN_USER=admin
TO_EMAIL=cdnadmin@example.com
TO_HOST=trafficops
TO_PORT=443
TO_PERL_HOST=trafficops-perl
TO_PERL_PORT=443
TO_SECRET=blahblah
TP_HOST=trafficportal
TP_EMAIL=tp@cdn.example.com
TR_HOST=trafficrouter
TR_DNS_PORT=53
TR_HTTP_PORT=80
TR_HTTPS_PORT=443
TR_API_PORT=3333
TP_PORT=443
TS_EMAIL=tstats@cdn.example.com
TS_HOST=trafficstats
TS_PASSWORD=trafficstatspassword
TS_USER=tstats
TV_HOST=trafficvault
TV_USER=tvault
TV_PASSWORD=mwL5GP6Ghu_uJpkfjfiBmi3l9vfgL10
TV_EMAIL=tvault@cdn.example.com
TV_ADMIN_USER=admin
TV_ADMIN_PASSWORD=riakAdmin
TV_RIAK_USER=riakuser
TV_RIAK_PASSWORD=riakPassword
TV_INT_PORT=8087
TV_HTTP_PORT=8098
TV_HTTPS_PORT=8088
ENROLLER_DIR=/shared/enroller
AUTO_SNAPQUEUE_ENABLED=true
AUTO_SNAPQUEUE_SERVERS=trafficops,trafficops-perl,trafficmonitor,
trafficrouter,trafficvault,edge,mid
AUTO_SNAPQUEUE_POLL_INTERVAL=2
AUTO_SNAPQUEUE_ACTION_WAIT=2
Note: While these port settings can be changed without hampering the function of the CDN in a Box system, note that changing a port without also changing the matching port-mapping in `infrastructure/cdn-in-a-box/docker-compose.yml` for the affected service will make it unreachable from the host.

### X.509 SSL/TLS Certificates

All components in Apache Traffic Control utilize SSL/TLS secure communications by default. For SSL/TLS connections to properly validate within the “CDN in a Box” container network a shared self-signed X.509 Root CA is generated at the first initial startup. An X.509 Intermediate CA is also generated and signed by the Root CA. Additional “wildcard” certificates are generated/signed by the Intermediate CA for each container service and demo1, demo2, and demo3 Delivery Services. All certificates and keys are stored in the `ca` host volume which is located at `infrastructure/cdn-in-a-box/traffic_ops/ca`.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
<th>X.509 CN/SAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIAB-CA-root.crt</td>
<td>Shared Root CA Certificate</td>
<td>N/A</td>
</tr>
<tr>
<td>CIAB-CA-intr.crt</td>
<td>Shared Intermediate CA Certificate</td>
<td>N/A</td>
</tr>
<tr>
<td>CIAB-CA-fullchain.crt</td>
<td>Shared CA Certificate Chain Bundle(^5)</td>
<td>N/A</td>
</tr>
<tr>
<td>infra.ciab.test.crt</td>
<td>Infrastructure Certificate</td>
<td><code>prefix.infra.ciab.test</code></td>
</tr>
<tr>
<td>demo1.mycdn.ciab.test.crt</td>
<td>Demo 1  Delivery Service Certificate</td>
<td><code>prefix.demo1.mycdn.ciab.test</code></td>
</tr>
<tr>
<td>demo2.mycdn.ciab.test.crt</td>
<td>Demo 2  Delivery Service Certificate</td>
<td><code>prefix.demo2.mycdn.ciab.test</code></td>
</tr>
<tr>
<td>demo3.mycdn.ciab.test.crt</td>
<td>Demo 3  Delivery Service Certificate</td>
<td><code>prefix.demo3.mycdn.ciab.test</code></td>
</tr>
</tbody>
</table>

### Trusting the Certificate Authority

For developer and testing use-cases, it may be necessary to have full x509 CA validation by HTTPS clients\(^6\). For x509 validation to work properly, the self-signed x509 CA certificate must be trusted either at the system level or by the client application itself.

---

\(^4\) The `ca` volume is not purged with normal `docker volume` commands. This feature is by design to allow the existing shared SSL certificate to be trusted at the system level across restarts. To re-generate all SSL certificates and keys, remove the `infrastructure/cdn-in-a-box/traffic_ops/ca` directory before startup.

\(^5\) The full chain bundle is a file that contains both the Root and Intermediate CA certificates.

\(^6\) All containers within CDN-in-a-Box start up with the self-signed CA already trusted.

\(^7\) The ‘demo1’ Delivery Service X509 certificate is automatically imported into Traffic Vault on startup.
Note: HTTP Client applications such as Google Chrome, Firefox, `curl(1)`, and `wget(1)` can also be individually configured to trust the CA certificate. Review each program’s respective documentation for instructions.

### Importing the CA Certificate on OSX

1. Copy the CIAB root and intermediate CA certificates from `infrastructure/cdn-in-a-box/traffic_ops/ca/` to the Mac.
2. Double-click the CIAB root CA certificate to open it in Keychain Access.
3. The CIAB root CA certificate appears in login.
4. Copy the CIAB root CA certificate to System.
5. Open the CIAB root CA certificate, expand Trust, select *Use System Defaults*, and save your changes.
6. Reopen the CIAB root CA certificate, expand Trust, select :guilabel:`Always Trust`, and save your changes.
7. Delete the CIAB root CA certificate from login.
8. Repeat the previous steps with the Intermediate CA certificate to import it as well.
9. Restart all HTTPS clients (browsers, etc).

### Importing the CA certificate on Windows

1. Copy the CIAB root CA and intermediate CA certificates from `infrastructure/cdn-in-a-box/traffic_ops/ca/` to Windows filesystem.
2. As Administrator, start the Microsoft Management Console.
3. Add the Certificates snap-in for the computer account and manage certificates for the local computer.
4. Import the CIAB root CA certificate into *Trusted Root Certification Authorities* → *Certificates*.
5. Import the CIAB intermediate CA certificate into *Trusted Root Certification Authorities* → *Certificates*.
6. Restart all HTTPS clients (browsers, etc).

### Importing the CA certificate on Linux/Centos7

1. Copy the CIAB full chain CA certificate bundle from `infrastructure/cdn-in-a-box/traffic_ops/ca/CIAB-CA-fullchain.crt` to path `/etc/pki/ca-trust/source/anchors/`. 

---

3.1. Administrator’s Guide 143
2. Run `update-ca-trust-extract` as the root user or with `sudo(8)`.

3. Restart all HTTPS clients (browsers, etc).

**Importing the CA certificate on Linux/Ubuntu**

1. Copy the CIAB full chain CA certificate bundle from `infrastructure/cdn-in-a-box/traffic_ops/ca/CIAB-CA-fullchain.crt` to path `/usr/local/share/ca-certificates/`.

2. Run `update-ca-certificates` as the root user or with `sudo(8)`.

3. Restart all HTTPS clients (browsers, etc).

**Advanced Usage**

This section will be amended as functionality is added to the CDN in a Box project.

**The Enroller**

The “enroller” began as an efficient way for Traffic Ops to be populated with data as CDN in a Box starts up. It connects to Traffic Ops as the “admin” user and processes files places in the docker volume shared between the containers. The enroller watches each directory within the `/shared/enroller` directory for new `filename.json` files to be created there. These files must follow the format outlined in the API guide for the `POST` method for each data type, (e.g. for a tenant, follow the guidelines for `POST api/1.4/regions`). Of note, the enroller does not require fields that reference database ids for other objects within the database.

---

**--dir** directory

Base directory to watch for data. Mutually exclusive with **--http**.

**--http** port

Act as an HTTP server for `POST` requests on this port. Mutually exclusive with **--dir**.

**--started** filename

The name of a file which will be created in the **--dir** directory when given, indicating service was started (default: “enroller-started”).

The enroller runs within CDN in a Box using **--dir** which provides the above behavior. It can also be run using **--http** to instead have it listen on the indicated port. In this case, it accepts only `POST` requests with the JSON provided in the request payload, e.g. `curl -X POST https://enroller/api/1.4/regions -d @newregion.json`. CDN in a Box does not currently use this method, but may be modified in the future to avoid using the shared volume approach.
Auto Snapshot/Queue-Updates

An automatic snapshot of the current Traffic Ops CDN configuration/toplogy will be performed once the “enroller” has finished loading all of the data and a minimum number of servers have been enrolled. To enable this feature, set the boolean AUTO_SNAPQUEUE_ENABLED to true\(^8\). The snapshot and queue-updates actions will not be performed until all servers in AUTO_SNAPQUEUE_SERVERS (comma-delimited string) have been enrolled. The current enrolled servers will be polled every AUTO_SNAPQUEUE_POLL_INTERVAL seconds, and each action (snapshot and queue-updates) will be delayed AUTO_SNAPQUEUE_ACTION_WAIT seconds\(^9\).

Mock Origin Service

The default “origin” service container provides a basic static file HTTP server as the central repository for content. Additional files can be added to the origin root content directory located at infrastructure/cdn-in-a-box/origin/content. To request content directly from the origin directly and bypass the CDN:

- Origin Service URL: http://origin.infra.ciab.test/index.html
- Docker Host: http://localhost:9200/index.html

Optional Containers

All optional containers that are not part of the core CDN-in-a-Box stack are located in the infrastructure/cdn-in-a-box/optional directory.

- infrastructure/cdn-in-a-box/optional/docker-compose.NAME.yml
- infrastructure/cdn-in-a-box/optional/NAME/Dockerfile

Multiple optional containers may be combined by using a shell alias:

```
# 52: Starting Optional Containers with an Alias

# From the infrastructure/cdn-in-a-box directory
# (Assuming the names of the optional services are stored in the \`
# `NAME1` and `NAME2` environment variables)
alias mydc="docker-compose -f $PWD/docker-compose.yml -f $PWD/
  -f optional/docker-compose.$NAME1.yml -f $PWD/optional/docker-
  compose.$NAME2.yml"
docker volume prune -f
mydc build
mydc up
```

\(^8\) Automatic Snapshot/Queue-Updates is enabled by default in infrastructure/cdn-in-a-box/variables.env.

\(^9\) Server poll interval and delay action wait are defaulted to a value of 2 seconds.
## VNC Server

The TightVNC optional container provides a basic lightweight window manager (fluxbox), Firefox browser, xterm, and a few other utilities within the CDN-In-A-Box “tcnet” bridge network. This can be very helpful for quick demonstrations of CDN-in-a-Box that require the use of real container network FQDNs and full X.509 validation.

1. Download and install a VNC client. TightVNC client is preferred as it supports window resizing, host-to-vnc copy/pasting, and optimized frame buffer compression.

2. Set your VNC console password by adding the `VNC_PASSWD` environment variable to `infrastructure/cdn-in-a-box/variables.env`. The password needs to be at least six characters long. The default password is randomized for security.

3. Start up CDN-in-a-Box stack. It is recommended that this be done using a custom bash alias

   ```bash
   # From infrastructure/cdn-in-a-box
   alias mydc="docker-compose -f $PWD/docker-compose.yml -f $PWD/optional/docker-compose.vnc.yml"
docker volume prune -f
mydc build
mydc kill
mydc rm -fv
mydc up
   ```

4. Connect with a VNC client to localhost port 9080.

5. When Traffic Portal becomes available, the Firefox within the VNC instance will subsequently be started.

6. An xterm with bash shell is also automatically spawned and minimized for convenience.

## Socks Proxy

Dante’s socks proxy is an optional container that can be used to provide browsers and other clients the ability to resolve DNS queries and network connectivity directly on the tcnet bridged interface. This is very helpful when running the CDN-In-A-Box stack on OSX/Windows docker host that lacks network bridge/IP-forward support. Below is the basic procedure to enable the Socks Proxy support for CDN-in-a-Box:

1. Start the CDN-in-a-Box stack at least once so that the x.509 self-signed CA is created.

2. On the host, import and Trust the CA for your target Operating System. See *Trusting the Certificate Authority*

3. On the host, using either Firefox or Chrome, download the FoxyProxy browser plugin which enables dynamic proxy support via URL regular expression...
4. Once FoxyProxy is installed, click the Fox icon on the upper right hand of the browser window, select Options.

5. Once in Options Dialog, Click Add New Proxy and navigate to the General tab:

6. Fill in the General tab according to the table

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy Name</td>
<td>CIAB</td>
</tr>
<tr>
<td>Color</td>
<td>Green</td>
</tr>
</tbody>
</table>

7. Fill in the Proxy Details tab according to the table

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Proxy Configuration</td>
<td>CIAB</td>
</tr>
<tr>
<td>Host or IP Address</td>
<td>localhost</td>
</tr>
<tr>
<td>Port</td>
<td>9080</td>
</tr>
<tr>
<td>Socks Proxy</td>
<td>checked</td>
</tr>
<tr>
<td>Socks V5</td>
<td>selected</td>
</tr>
</tbody>
</table>

8. Go to URL Patterns tab, click Add New Pattern, and fill out form according to

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Name</td>
<td>CIAB Pattern</td>
</tr>
<tr>
<td>URL Pattern</td>
<td><em>.test/</em></td>
</tr>
<tr>
<td>Whitelists</td>
<td>selected</td>
</tr>
<tr>
<td>Wildcards</td>
<td>selected</td>
</tr>
</tbody>
</table>

9. Enable dynamic ‘pre-defined and patterns’ mode by clicking the fox icon in the upper right of the browser. This mode only forwards URLs that match the wildcard *.*.test/* to the Socks V5 proxy.

10. On the docker host start up CDN-in-a-Box stack. It is recommended that this be done using a custom bash alias

```bash
# From infrastructure/cdn-in-a-box
alias mydc="docker-compose -f $PWD/docker-compose.yml -f $PWD/optional/docker-compose.socksproxy.yml"
docker volume prune -f
mydc build
mydc kill
mydc rm -fv
mydc up
```
11. Once the CDN-in-a-box stack has started, use the aforementioned browser to access Traffic Portal via the socks proxy on the docker host.

See also:
The official Docker Compose documentation CLI reference for complete instructions on how to pass service definition files to the `docker-compose` executable.

**Static Subnet**

Since `docker-compose` will randomly create a subnet and it has a chance to conflict with your network environment, using static subnet is a good choice.

```
# 55: CIAB Startup with Static Subnet
# From the infrastructure/cdn-in-a-box directory
alias mydc="docker-compose -f $PWD/docker-compose.yml -f $PWD/optional/docker-compose.static-subnet.yml"
docker volume prune -f
mydc build
mydc up
```

**VPN Server**

This container provides an OpenVPN service. It’s primary use is to allow users and developers to easily access CIAB network.

**How to use it**

1. It is recommended that this be done using a custom bash alias.

```
# 56: CIAB Startup with VPN
# From infrastructure/cdn-in-a-box
mydc down -v
mydc build
mydc up
```

2. All certificates, keys, and client configuration are stored at `infrastructure/cdn-in-a-box/optional/vpn/vpnca`. You just simply change `REALHOSTIP` and `REALPORT` of `client.ovpn` to fit your environment, and then you can use it to connect to this OpenVPN server.
The proposed VPN client

On Linux, we suggest openvpn. On most Linux distributions, this will also be the name of the package that provides it.

```
# 57: Install openvpn on ubuntu/debian
apt-get update && apt-get install -y openvpn
```

On OSX, it only works with brew installed openvpn client, not the OpenVPN GUI client.

```
# 58: Install openvpn on OSX
brew install openvpn
```

If you want a GUI version of VPN client, we recommend Tunnelblick.

Private Subnet for Routing

Since docker-compose randomly creates a subnet, this container prepares 2 default private subnets for routing:

- 172.16.127.0/255.255.240.0
- 10.16.127.0/255.255.240.0

The subnet that will be used is determined automatically based on the subnet prefix. If the subnet prefix which docker-compose selected is 192. or 10., this container will select 172.16.127.0/255.255.240.0 for its routing subnet. Otherwise, it selects 10.16.127.0/255.255.240.0.

Of course, you can decide which routing subnet subnet by supplying the environment variables PRIVATE_NETWORK and PRIVATE_NETMASK.

Pushed Settings

Pushed settings are shown as follows:

- DNS
- A routing rule for the CIAB subnet

**Note:** It will not change your default gateway. That means apart from CDN in a Box traffic and DNS requests, all other traffic will use the standard interface bound to the default gateway.
Grafana

This container provides a Grafana service. It’s an open platform for analytics and monitoring. This container has prepared necessary datasources and scripted dashboards. Please refer to Configuring Grafana for detailed Settings.

How to start it

It is recommended that this be done using a custom bash alias.

```
# 59: CIAB Startup with Grafana
mydc down -v
mydc build
mydc up
```

Apart from start Grafana, the above commands also expose port 3000 for it.

Check the charts

There are some scripted dashboards can show beautiful charts. You can display different charts by passing in different query string

- [https://<grafanaHost>/dashboard/script/traffic_ops_cachegroup.js?which=](https://<grafanaHost>/dashboard/script/traffic_ops_cachegroup.js?which=). The query parameter `which` in this particular URL should be the `cachegroup`. Take CIAB as an example, it can be filled in with **CDN_in_a_Box_Edge** or **CDN_in_a_Box_Edge**.

- [https://<grafanaHost>/dashboard/script/traffic_ops_deliveryservice.js?which=](https://<grafanaHost>/dashboard/script/traffic_ops_deliveryservice.js?which=). The query parameter `which` in this particular URL should be the `xml_id` of the desired Delivery Service.

- [https://<grafanaHost>/dashboard/script/traffic_ops_server.js?which=](https://<grafanaHost>/dashboard/script/traffic_ops_server.js?which=). The query parameter `which` in this particular URL should be the `hostname` (not FQDN). It can be filled in with `edge` or `mid` in CIAB.

Configure DNSSEC

See also:

**DNSSEC**

**Note:** In order for Traffic Ops to successfully store keys in Traffic Vault, at least one Riak
Server needs to be configured in Traffic Ops. See the Traffic Vault admin page for more information.

**Note:** Currently DNSSEC is only supported for DNS Delivery Services.

1. Go to CDNs and click on the desired CDN.

   ![CDNs Page](image)

   **Fig. 17: CDNs Page**

2. Click on More → Manage DNSSEC Keys.

   ![CDN Details Page](image)

   **Fig. 18: CDN Details Page**

3. Click on the Generate DNSSEC Keys button.

4. A modal will pop up asking you to confirm that you want to proceed.

5. Input the required information (reasonable defaults should be generated for you). When done, click on the green Generate button.

   **Note:** Depending upon the number of Delivery Services in the CDN, generating DNSSEC keys may take several seconds.
Fig. 19: DNSSEC Key Management Page

Fig. 20: Confirmation Modal

Fig. 21: DNSSEC Key Generation Page
6. You will be prompted to confirm the changes by typing the name of the CDN into a text box. After doing so, click on the red **Confirm** button.

![DNSSEC Key Change Confirmation](image)

**Fig. 22: DNSSEC Key Change Confirmation**

7. In order for DNSSEC to work properly, the DS (Delegation of Signing) Record information needs to be added to the parent zone of the CDN’s domain (e.g. If the CDN’s domain is ‘ciab.cdn.local’ the parent zone is ‘cdn.local’). If you control your parent zone you can enter this information yourself, otherwise you will need to work with your DNS team to get the DS Record added to the parent zone.

8. Once DS Record information has been added to the parent zone, DNSSEC needs to be activated for the CDN so that Traffic Router will sign responses. Go back to the CDN details page for this CDN, and set the ‘DNSSEC Enabled’ field to ‘true’, then click the green **Update** button.

![Change ‘DNSSEC Enabled’ to ‘true’](image)

**Fig. 23: Change ‘DNSSEC Enabled’ to ‘true’**

9. DNSSEC should now be active on your CDN and Traffic Router should be signing responses. This should be tested e.g. with this `dig(1)` command: `dig edge.cdn.local. +dnssec`.

10. When KSK expiration is approaching (default 365 days), it is necessary to manually generate a new KSK for the TLD and add the DS Record to the parent zone. In order to avoid signing errors, it is suggested that an effective date is chosen which allows time for the DS Record to be added to the parent zone before the new KSK becomes active.

**Delivery Service Requests**

When enabled in `traffic_portal_properties.json`, Delivery Service Requests are created when *all* users attempt to create, update or delete a *Delivery Service*. This allows users...
with higher level permissions (“operations” or “admin”) to review the changes for completeness and accuracy before deploying the changes. In addition, most Delivery Service changes require cache configuration updates (aka queue updates) and/or a CDN Snapshot. Both of these actions are reserved for users with elevated permissions.

A list of the Delivery Service requests associated with your Tenant can be found under Services → Delivery Service Requests

Who Can Create a Delivery Service Request and How?

Users with the Portal Role (or above) can create Delivery Service Requests by doing one of three things:

- Creating a new Delivery Service
- Updating an existing Delivery Service
- Deleting an existing Delivery Service

By performing one of these actions, a Delivery Service Request will be created for you with a status of ‘draft’ or ‘submitted’. You determine the status of your request upon submission. Only change the status of your request to ‘submitted’ once the request is ready for review and deployment.

Who Can Fulfill a Delivery Service Request and How?

Users with elevated permissions (Operations or above) can fulfill (apply the changes) or reject the Delivery Service Request. In fact, they can do all of the following:

Update the contents of the Delivery Service Request  This will update the “Last Edited By” field to indicate who last updated the request.

Assign or Unassign the Delivery Service Request  Assignment is currently limited to current user. This is optional as fulfillment will auto-assign the request to the user doing the fulfillment.

Reject the Delivery Service Request  Rejecting a Delivery Service Request will set status to ‘rejected’ and the request can no longer be modified. This will auto-assign the request to the user doing the rejection.

Fulfill the Delivery Service Request  Fulfilling a Delivery Service Request will show the requested changes and, once committed, will apply the desired changes and set status to
‘pending’. The request is pending because many types of changes will require cache server configuration updates (aka queue updates) and/or a CDN snapshot. Once queue updates and/or CDN snapshot is complete, the request should be marked ‘complete’.

**Complete the Delivery Service Request**  Only after the Delivery Service Request has been fulfilled and the changes have been applied can a Delivery Service Request be marked as ‘complete’. Marking a Delivery Service Request as ‘complete’ is currently a manual step because some changes require cache server configuration updates (aka queue updates) and/or a CDN Snapshot. Once that is done and the changes have been deployed, the request status should be changed from ‘pending’ to ‘complete’.

**Delete the Delivery Service request** Delivery Service Requests with a status of ‘draft’ or ‘submitted’ can always be deleted entirely if appropriate.

**Configure Federations**

1. Create a user with a federations role (*User Admin → Users → ‘+’ button*). This user will need the ability to:
   - Create/edit/delete federations
   - Add IPV4 resolvers
   - Add IPV6 resolvers
2. As a user with administrative privileges, create a Federation Mapping by going to Services → More → Federations and then clicking Add Federation Mapping.

3. Choose the Delivery Service to which the federation will be mapped and assign it to the Federation-role user; click Add.

4. After the Federation is added, Traffic Ops will display the Federation. Changes can be made at this time or the Federation can be deleted. Notice that no resolvers have been added to the Federation yet. This can only be done by the Federation-role user to whom the Federated Delivery Service was assigned. If no further action is necessary, the Close button will close the window and display the list of all Federations.

5. The federation user logs into either the Traffic Ops API or the Traffic Portal UI and stores the Mojolicious cookie. The Mojolicious cookie can be obtained manually using the debug tools on a web browser or via a command line utility like curl(1).

   # 60: Example cURL Command

   ```bash
curl -i -XPOST "http://localhost:3000/api/1.1/user/login" -H "Content-Type: application/json" -d '{ "u": "federation_user1", "p": "password" }'
   ```

   # 61: Example API Response

   ```
   HTTP/1.1 200 OK
   Date: Wed, 02 Dec 2015 21:12:06 GMT
   Content-Length: 65
   ```

   (continues on next page)
6. The federation user sends a request to Traffic Ops to add IPV4 and/or IPV6 resolvers

---

**# 62: Example cURL Command**

```
curl -ki -H "Cookie: mojolicious=eyJleHBpcmVzIjoiNDQ5MTA1MTI2LCJhbGwiOiJmZWRlcmF0aW9uX3VzZXIxIn0--06b4f870d809d82a91433e92eae8320875c3e8b0; expires=Thu, 03 Dec 2015 01:12:06 GMT; path=/; HttpOnly"

'http://localhost:3000/api/1.2/federations' -d '{
"federations": [
{
"deliveryService": "images-c1",
"mappings":
{
"resolve4": ["8.8.8.8/32", "8.8.4.4/32"],
"resolve6": ["2001:4860:4860::8888/128", "2001:4860:4860::8844"]
}
]
}'
```

---

**# 63: Example API Response**

```
HTTP/1.1 200 OK
Set-Cookie: mojolicious=eyJleHBpcmVzIjoiNDQ5MTA1MTI2LCJhbGwiOiJmZWRlcmF0aW9uX3VzZXIxIn0--b42be0749415cefd1d14ela91bb214845b4de556; expires=Thu, 03 Dec 2015 01:25:42 GMT; path=/; HttpOnly
Server: Mojolicious (Perl)
Date: Wed, 02 Dec 2015 21:25:42 GMT
```
7. The resolvers added by the Federation-user will now visible in Traffic Portal.
Any requests made from a client that resolves to one of the federation resolvers will now be given a CNAME Record from Traffic Router.

### Example DNS request (via `dig`)

```plaintext
dig @tr.kabletown.net foo.images-c1.kabletown.net
```

### Example Resolver Response

```plaintext
; <<>> DiG 9.7.3-RedHat-9.7.3-2.el6 <<>> @tr.kabletown.net
;foo.images-c1.kabletown.net
;(1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: recursion requested but not available
;; QUESTION SECTION:
;foo.images-c1.kabletown.net. IN A

;; ANSWER SECTION:
foo.images-c1.kabletown.net. 30 IN CNAME img.mega-cdn.net.

;; Query time: 9 msec
;; SERVER: 10.10.10.10#53(10.10.10.10)
;; MSG SIZE  rcvd: 84
```

### Configure Multi-Site Origin

The following steps will take you through the procedure of setting up an MSO.

1. Create `Cache Groups` for the origin locations, and assign the appropriate parent-child relationship between the Mid-tier `Cache Group(s)` and origin `Cache Groups`. Each Mid-tier `Cache Group` can be assigned a primary and secondary origin parent `Cache Group`. When the `Cache Group` parent configuration is generated, origins in the primary `Cache Groups` will be listed first, followed by origins in the secondary `Cache Group`. Origin servers assigned to the `Delivery Service` that are assigned to neither the primary nor secondary `Cache Groups` will be listed last.
1. Create a profile to assign to each of the origins:

![Profile creation](image)

2. Create server entries for the origination vips:

![Server creation](image)

3. Check the multi-site check box in the **Delivery Services** screen:

![Multi-site check](image)

4. Assign the org servers to the **Delivery Service** that will have the multi site feature. Origin servers assigned to a **Delivery Service** with multi-site checked will be assigned to be the origin servers for this **Delivery Service**.

**Note:** “Origin Server Base URL” uniqueness: In order to enable Mid-tier **Cache Group** to distinguish **Delivery Services** by different MSO algorithms while performing parent fail-over, it requires that OSBU (Origin Server Base
URL) for each MSO-enabled Delivery Service is unique. This means that the OSBU of an MSO-enabled Delivery Service should be different from the OSBUs of any other Delivery Service, regardless of whether they are MSO-enabled or not. The exceptions to this rule are:

- If there are multiple CDNs created on the same Traffic Ops, Delivery Services across different CDNs may have the same OSBU configured.
- If several Delivery Services in the same CDN have the same MSO algorithm configured, they may share the same OSBU.
- If delivery services are assigned with different Mid-tier Cache Groups respectively, they can share the same OSBU.
- This OSBU must be valid - ATS will perform a DNS lookup on this FQDN even if IPs, not DNS, are used in the parent.config.
- The OSBU entered as the “Origin Server Base URL” will be sent to the origins as a host header. All origins must be configured to respond to this host.

5. Create a delivery service profile. This must be done to set the MSO algorithm. Also, as of ATS 6.x, multi-site options must be set as parameters within the parent.config. Header rewrite parameters will be ignored. See ATS parent.config for more details. These parameters are now handled by the creation of a Delivery Service profile.

   a) Create a profile of the type DS_PROFILE for the Delivery Service in question.

   b) Click Show profile parameters to bring up the parameters screen for the profile. Create parameters for the following:
c) In the Delivery Service page, select the newly created DS_PROFILE and save the Delivery Service.

6. Turn on parent_proxy_routing in the MID profile.

**Note:** Support for multisite configurations with single-layer CDNs is now available. If a Cache Groups defined parents are either blank or of the type ORG_LOC, that cache server’s parent.config will be generated as a top layer cache, even if it is an edge. In the past, parent.config generation was strictly determined by cache type. The new method examines the parent Cache Group definitions and generates the parent.config accordingly.
Configure Regional Geo-blocking (RGB)

Note: RGB (Regional Geographic-based Blocking) is only supported for HTTP Delivery Services.

1. Prepare an RGB configuration file. RGB uses a configuration file in JSON format to define regional geographic blocking rules for Delivery Services. The file needs to be put on an HTTP server accessible to Traffic Router.

```json
{
  "deliveryServices":
    [
      {
        "deliveryServiceId": "hls-live",
        "urlRegex": ".*live4\./m3u8",
        "geoLocation": {
          "includePostalCode": ["N0H", "L9V", "L9W"]
        },
        "redirectUrl": "http://third-party.com/blacked_out.html"
      },
      {
        "deliveryServiceId": "hls-live",
        "urlRegex": ".*live5\./m3u8",
        "ipWhiteList": ["185.68.71.9/22", "142.232.0.79/24"],
        "geoLocation": {
          "excludePostalCode": ["N0H", "L9V"]
        },
        "redirectUrl": "/live5_low_bitrate.m3u8"
      }
    ]
}
```

deliveryServiceId Should be equal to the ID or xml_id field of the intended Delivery Service as configured in Traffic Portal

urlRegex A regular expression to be used to determine to what URLs the rule shall apply; a URL that matches it is subject to the rule

geoLocation An object that currently supports only the keys includePostalCode and excludePostalCode (mutually exclusive). When the includePostalCode key is used, only the clients whose FSA (Forward Sortation Areas) - the first three postal characters of Canadian postal codes - are in the includePostalCode list are able to view the content at URLs matched by the urlRegex. When excludePostalCode is used, any client whose FSA is not in the excludePostalCode list will be allowed to view the content.
**redirectUrl**  The URL that will be returned to the blocked clients. Without a domain name in the URL, the URL will still be served in the same *Delivery Service*. Thus Traffic Router will redirect the client to a chosen *cache server* assigned to the *Delivery Service*. If the URL includes a domain name, Traffic Router simply redirects the client to the defined URL. In the latter case, the redirect URL must not match the *urlRegex* value, or an infinite loop of HTTP 302 *Found* responses will occur at the Traffic Router.

**ipWhiteList**  An optional element that is an array of CIDR blocks indicating the IPv4 subnets that are allowed by the rule. If this list exists and the value is not empty, client IP will be matched against the CIDR list, bypassing the value of *geoLocation*. If there is no match in the white list, Traffic Router defers to the value of *geoLocation* to determine if content ought to be blocked.

1. Add RGB parameters in Traffic Portal to the *Delivery Service*’s Traffic Router(s)’s profile(s). The *configFile* field should be set to *CRConfig.json*, and the following two parameter name/values need to be specified:

   **regional_geoblocking.polling.url**  The URL of the RGB configuration file. Traffic Router will fetch the file from this URL using an HTTP GET request.

   **regional_geoblocking.polling.interval**  The interval on which Traffic Router polls the RGB configuration file.

2. Enable RGB for a *Delivery Service*

3. Go to *Tools → Snapshot CRConfig*, perform *Diff CRConfig* and click *Write CRConfig*.

---

**Traffic Router Access Log**

See also:

*Troubleshooting and Log Files*
Regular expressions for this delivery service:

Type: HOST_REGEXP  Order: 0  Regular Expression:

Add a new Regex:
Select...

Save  Close
RGB extends the rtype field and adds a new field rgb in Traffic Router access.log to help to monitor this feature. A value of RGALT in the rtype field indicates that a request is redirected to an alternate URL by RGB; a value of RGDENY indicates that a request is denied by RGB because there is no matching rule in the RGB configuration file for this request. When RGB is enabled, the RGB field will be non-empty with following format:

```
{FSA}:{allowed/disallowed}:{include/exclude postal}:{fallback config}:{allowed by whitelist}
```

**FSA** FSA part of the client’s postal code, which is retrieved from a geographic location database. If this field is empty, a dash (“-“) is filled in.

**allowed/disallowed** This flag shows if a request was allowed or disallowed by RGB (1 for yes, and 0 for no).

**include/exclude postal** This shows that when a rule in JSON is matched for a request, it’s value is “I” if the rule matched because of an includePostalCode rule, “X” if the rule matched because of an excludePostalCode rule, or “-” if no rule matched.

**fallback config** When Traffic Router fails to parse an RGB configuration file as JSON, Traffic Router will handle requests with latest valid configuration that it had, but will set the fallback config flag to 1. If no fall-back occurred, then the flag is set to 0.

**allowed by whitelist** If a request is allowed by a whitelist field in the configuration, this flag is set to 1; for all other cases, it is 0.

### # 67: Example

```
1446442214.685 qtype=HTTP chi=129.100.254.79 url="http://foo.geo2.cdn.com/live5.m3u8" cqhm=GET cqhv=HTTP/1.1 rtype=GEO rloc="-" rdtl=- rerr="-" rgb="N6G:1:X:0:0" pssc=302 ttms=3 rurl=http://cent6-44.geo2.cdn.com/live5.m3u8 rh="-"
```

```
1446442219.181 qtype=HTTP chi=184.68.71.9 url="http://foo.geo2.cdn.com/live5.m3u8" cqhm=GET cqhv=HTTP/1.1 rtype=RGALT rloc="-" rdtl=- rerr="-" rgb="-:0:X:0:0" pssc=302 ttms=3 rurl=http://cent6-44.geo2.cdn.com/low_bitrate.m3u8 rh="-"
```

```
1446445521.677 qtype=HTTP chi=24.114.29.79 url="http://foo.geo2.cdn.com/live51.m3u8" cqhm=GET cqhv=HTTP/1.1 rtype=RGDENY rloc="-" rdtl=- rerr="-" rgb="L4S:0:-:0:0" pssc=520 ttms=3 rurl="-" rh="-"
```

### Configuring Static DNS Entries

Static DNS records (historically “entries”) can be configured within the DNS subdomain of a given Delivery Service. In a typical scenario, the Delivery Service will have DNS records automatically generated based on its “xml_id” and “routing name”, and the name and subdomain of the CDN to which it belongs. For example, in the CDN in a Box default environment, the “demo1” Delivery Service has an automatically generated DNS record for video.demo1.mycdn.ciab.test. Configuring a static DNS record allows for further extension of this,
for example, one could create an A record that enforces lookups of the name foo.demo1.mycdn.ciab.test resolve to the IPv4 address 192.0.2.1.

**Note:** It’s only possible to create static DNS records within a *Delivery Service*’s subdomain. That is, one could not create an A record for foo.bar.mycdn.ciab.test on the *CDN in a Box Delivery Service* “demo1”, since “demo1”’s subdomain is demo1.mycdn.ciab.test.

**See also:**

This guide covers how to set up static DNS records using Traffic Portal. It’s also possible to do so directly using the *Traffic Ops API* endpoint `staticdnsentries`.

**Example**

To set up the aforementioned rule, follow these steps.

1. In Traffic Portal, expand the *Services* sidebar menu and select *Delivery Services*.
2. From the now-displayed table of *Delivery Services*, select the desired one for static DNS record configuration.
3. From the *More* drop-down menu, select *Static DNS Entries*. The displayed table will probably be empty.

![Fig. 25: The Static DNS Entries Table Page](image)

4. Click on the blue + button to add a new static DNS Entry
5. Fill in all of the fields.

**Host**  This is the lowest-level DNS label that will be used in the DNS record. In the CDN in a Box scenario, for example, entering foo here will result in a full DNS name of foo.demo1.mycdn.ciab.test.

**Type**  Indicates the type of DNS record that will be created. The available types are

- A
- AAAA
- CNAME
- TXT

**TTL**  The TTL of the DNS record, after which clients will be expected to re-request name resolution.

**Address**  The meaning of this field depends on the value of the “Type” field.

- If the “Type” is A, this must be a valid IPv4 address
- If the “Type” is AAAA, this must be a valid IPv6 address
- If the “Type” is CNAME, this must be a valid DNS name - not an IP address at all
- If the “Type” is TXT, no restrictions are placed on the content whatsoever

![Fig. 26: An Example Static DNS Entry Form](image)

6. Click on the green *Create* button to finalize the changes.
7. At this point, although the static DNS record has been created, it will have no effect until a new CDN Snapshot is taken. Once that is done (and enough time has passed for Traffic Router to poll for the changes), the new DNS record should be usable through the CDN’s designated Traffic Router.

# 68: Example DNS Query to Test a New Static DNS Entry within CDN in a Box

```bash
$ docker exec cdninabox_enroller_1 dig +noall +answer foo.demo1.mycdn.ciab.test
foo.demo1.mycdn.ciab.test. 42 IN A 192.0.2.1
```

Configure Delivery Service Steering

1. Create two target Delivery Services in Traffic Portal. They must both be HTTP Delivery Services that are part of the same CDN.

![Fig. 27: Target Delivery Services](image)

2. Create a Delivery Service with Type STEERING or CLIENT_STEERING in Traffic Ops.

![Fig. 28: Creating a STEERING Delivery Service](image)
3. Click More → View Targets and then use the blue + button to assign targets.

4. If desired, a ‘steering’ Role user can create filters for the target Delivery Services using steering/{{ID}}/targets

   **Note:** This is only available via the Traffic Ops API; no functionality for manipulating steering targets is offered by Traffic Portal. This feature has been requested and is tracked by GitHub Issue #2811

5. Any requests to Traffic Router for the steering Delivery Services should now be routed to target Delivery Services based on configured weight or order.

   **Note:** This example assumes that the Traffic Ops instance is running at to.cdn.local and the administrative username and password are admin and twelve, respectively. This is not recommended in production, but merely meant to replicate the default CDN in a Box environment!
A guide to the various internal and external APIs, and an 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 Building Traffic Control

The build steps for Traffic Control components are all pretty much the same, despite that they are written in a variety of different languages and frameworks. This is accomplished by using Docker.

**Note:** Currently, both listed methods of building Traffic Control components will produce *.rpm files, meaning that the support of these components is limited to RedHat-based distributions - and none of them are currently tested (or guaranteed to work) outside of CentOS7, specifically.

**Downloading Traffic Control**

If any local work on Traffic Monitor, Traffic Router Golang, Grove or Traffic Ops is to be done, it is highly recommended that the Traffic Control repository be downloaded inside
the $GOPATH directory. Specifically, the best location is $GOPATH/src/github.com/apache/trafficcontrol. Cloning the repository outside of this location will require either linking the actual directory to that point, or moving/copying it there.

See also:
The Golang project’s GOPATH wiki page

Build Using pkg

This is the easiest way to build all the components of Traffic Control; all requirements are automatically loaded into the image used to build each component.

Requirements

- Docker
- Docker Compose

Usage

./pkg [options] [projects]

Note: The pkg script often needs to be run as sudo, as certain privileges are required to run Docker containers

Options

- `-q` Quiet mode. Supresses output.
- `-v` Verbose mode. Lists all build output.
- `-l` List available projects.

If present, `projects` should be one or more project names. When no specific project or project list is given the default projects will be built. Valid projects:

- `docs`
- `grove_build`
- `grovetc cfg_build`
- `source`
- `traffic_monitor_build`
- `traffic_ops_build`

---

1 This is optional, but recommended. If a docker-compose executable is not available the pkg script will automatically download and run it using a container. This is noticeably slower than running it natively.

2 This is a default project, which will be built if pkg is run with no projects argument.
• traffic_portal_build
• traffic_router_build
• traffic_stats_build
• weasel

Output component-version.rpm files, build logs and source tarballs will be output to the dist/ directory at the root of the Traffic Control repository directory.

Build Using docker-compose

If the pkg script fails, docker-compose can still be used to build the projects directly. The compose file can be found at infrastructure/docker/build/docker-compose.yml under the repository’s root directory. It can be passed directly to docker-compose, either from the infrastructure/docker/build/ directory or by explicitly passing a path to the infrastructure/docker/build/docker-compose.yml file via -f. It is recommended that between builds docker-compose down -v is run to prevent caching of old build steps. The service names are the same as the project names described above in Usage, and similar to the pkg script, the build results, logs and source tarballs may all be found in the dist directory after completion.

Note: Calling docker-compose in the way described above will build _all_ projects, not just the default projects.

See also:
The Docker Compose command line reference

Building Individual Components

Each Traffic Control component can be individually built, and the instructions for doing so may be found in their respective component’s development documentation.

Building This Documentation

This documentation uses the Sphinx documentation build system, and as such requires a Python3 version that is at least 3.4.1. It also has dependency on Sphinx, and Sphinx extensions and themes. All of these can be easily installed using pip by referencing the requirements file like so:


# 69: Run from the Repository’s Root Directory

```bash
python3 -m pip install --user -r docs/source/requirements.txt
```

Once all dependencies have been satisfied, build using the Makefile at `docs/Makefile`. Alternatively, it is also possible to `Build Using pkg` or to `Build Using docker-compose`, both of which will output a documentation “tarball” to `dist/`.

## 4.1.2 Traffic Ops

### Introduction

Traffic Ops uses a PostgreSQL database to store the configuration information, and the Mojo-licious framework to generate the user interface and REST APIs.

### Software Requirements

To work on Traffic Ops you need a CentOS 7+ environment that has the following installed:

- **Carton 1.0.12**
  - libpcap (plus development library - usually “libpcap-dev” or “libpcap-devel”)
  - libpq (plus development library - usually “libpq-dev” or “libpq-devel”)
  - cpan JSON
  - cpan JSON::PP
- **Go 1.8.3**
- **Perl 5.10.1**
- **Git**
- **PostgreSQL 9.6.6**
- **Goose**

### Traffic Ops Project Tree Overview

traffic_ops/ - The root of the Traffic Ops project

- **app/** - Holds most of the Perl code base
  - **bin/** - Directory for scripts, `cron(8)` jobs, etc
  - **conf/**
    - `development/` - Development (local) specific configuration files.
    - `misc/` - Miscellaneous configuration files.
* production/ - Production specific configuration files.
* test/ - Test (unit test) specific configuration files.

- db/ - Database related area.
  * migrations/ - Database Migration files.

- lib/
  * API/ - Mojolicious Controllers for the Traffic Ops API
  * Common/ - Common Code between both the Traffic Ops API and the deprecated Traffic Ops UI
  * Extensions/ - Contains Data Source Extensions
  * Fixtures/ - Test Case fixture data for the ‘to_test’ database.
    - Integration/ - Integration Tests.
  * Schema/ - Database Schema area.
    - /Result - DBIx ORM related files.
  * /Test - Common Test.
  * UI/ - Mojolicious Controllers for the deprecated Traffic Ops UI.
  * Utils/
    - Helper/ - Common utilities for the Traffic Ops application.

- log/ - Log directory where the development and test files are written
- public/
- css/ - Stylesheets
- images/ - Images
- js/ - Javascripts
- script/ - Mojolicious Bootstrap scripts.
- t/ - Unit Tests for the UI.
- api/ - Unit Tests for the API.
- t_integration/ - High level tests for Integration level testing.
- templates/ - Mojolicious Embedded Perl (*.ep) files for the UI.

- bin/ - holds executables related to Traffic Ops, but not actually a part of the Traffic Ops server’s operation
- build/ - contains files that are responsible for packaging Traffic Ops into an RPM file
- client/ - API endpoints handled by Go
- client_tests/ - lol
• doc/ - contains only a `coverage-zone.json` example (?) file

• etc/ - configuration files needed for the Traffic Ops server
  – cron.d/ - holds specifications for `cron(8)` jobs that need to be run periodically on Traffic Ops servers
  – init.d/ - contains the old initscripts-based job control for Traffic Ops
  – logrotate.d/ - specifications for the Linux `logrotate(8)` utility for Traffic Ops log files
  – profile.d/traffic_ops.sh - sets up common environment variables for working with Traffic Ops

• experimental/ - includes all kinds of prototype and/or abandoned tools and extensions
  – ats_config/ - an attempt to provide an easier method of obtaining and/or writing configuration files for ATS `cache servers`
  – auth/ - a simple authentication server that mimics the authentication process of Traffic Ops, and provides a detailed view of a logged-in user's permissions and capabilities
  – goto/ - an Angular (1.x) web page backed by a Go server that provides a ReST API interface for mySQL servers
  – postgrest/ - originally probably going to be a web server that provides a ReST API for PostgreSQL servers, this only contains a simple - albeit unfinished - Docker container specification for running PostgreSQL client tools and/or server(s)
  – server/ - a living copy of the original attempt at re-writing Traffic Ops in Go
  – traffic_ops_auth/ - proof-of-concept for authenticating, creating and deleting users in a Traffic Ops schema.
  – url-rewriter-nginx/ - Docker container specification for a modification to the NginX web server, meant to make it suitable for use as a caching server at the Edge-tier or Mid-tier levels of the Traffic Control architecture
  – webfront/ - a simple HTTP caching server written from the ground-up, meant to be suitable as a caching server at the Edge-tier or Mid-tier levels of the Traffic Control architecture

• install/ - contains all of the resources necessary for a full install of Traffic Ops
  – bin/ - binaries related to installing Traffic Ops, as well as installing its prerequisites, certificates, and database
  – data/ - almost nothing
  – etc/ - this directory left empty; it's used to contain post-installation extensions and resources
  – lib/ - contains libraries used by the various installation binaries

• testing/ - holds utilities for testing the `Traffic Ops API`, as well as comparing two separate API instances (for e.g. comparing a new build to a known-to-work build)
• traffic_ops_golang/ - has all of the functionality that has been re-written from Perl into Go
• vendor/ - contains “vendored” packages from third party sources

Perl Formatting Conventions

Perltidy is for use in code formatting.

```bash
# 70: Example Perltidy Configuration (usually in ~/.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=""
```

app/db/admin

The **app/db/admin** binary is for use in managing the Traffic Ops database tables. This essentially serves as a front-end for **Goose**.

**Note:** For proper resolution of configuration and SOL statement files, it’s recommended that this binary be run from the **app** directory

**Usage**

db/admin [options] command
Options and Arguments

**--env** ENVIRONMENT

An optional environment specification that causes the database configuration to be read out of the corresponding section of the `app/db/dbconf.yml` configuration file. One of:

- development
- integration
- production
- test

(Default: development)

**MOJO_MODE**

`admin` sets this to the value of the environment as specified by `--env` (Default: development)

**command**

The `command` specifies the operation to be performed on the database. It must be one of:

- `createdb` Creates the database for the current environment
- `create_user` Creates the user defined for the current environment
- `dbversion` Displays the database version that results from the current sequence of migrations
- `down` Rolls back a single migration from the current version
- `drop` Drops the database for the current environment
- `drop_user` Drops the user defined for the current environment
- `load_schema` Sets up the database for the current environment according to the SQL statements in `app/db/create_tables.sql`
- `migrate` Runs a migration on the database for the current environment
- `patch` Patches the database for the current environment using the SQL statements from `app/db/patches.sql`
- `redo` Rolls back the most recently applied migration, then run it again
- `reset` Creates the user defined for the current environment, drops the database for the current environment, creates a new one, loads the schema into it, and runs a single migration on it
- `reverse_schema` Reverse engineers the `app/lib/Schema/Result/*` files from the environment database
- `seed` Executes the SQL statements from the `app/db/seeds.sql` file for loading static data
show_users  Displays a list of all users registered with the PostgreSQL server
status  Prints the status of all migrations
upgrade  Performs a migration on the database for the current environment, then seeds it and patches it using the SQL statements from the app/db/patches.sql file

# 71: Example Usage

db/admin --env=test reset

The environments are defined in the app/db/dbconf.yml file, and the name of the database generated will be the name of the environment for which it was created.

Installing The Developer Environment

To install the Traffic Ops Developer environment:

1. Clone the Traffic Control repository from GitHub.
2. Install the local dependencies using Carton.

# 72: Install Development Dependencies

cd traffic_ops/app
carton

3. Set up a role (user) in PostgreSQL

See also:

PostgreSQL instructions on setting up a database.

4. Use the reset and upgrade commands of admin (see app/db/admin for usage) to set up the traffic_ops database(s).

5. (Optional) To load the ‘KableTown’ example/testing data set into the tables, use the app/bin/db/setup_kabletown.pl script.

   Note: To ensure proper paths to Perl libraries and resource files, setup_kabletown.pl should be run from within the app/ directory.

6. Run the postinstall script, located in install/bin/

7. To start Traffic Ops, use the start.pl script located in the app/bin directory. If the server starts successfully, the STDOUT of the process should contain the line [<date and time>] [INFO] Listening at "http://*:3000", followed by the line Server available at http://127.0.0.1:3000 (using default settings for port number and listening address, and where <date and time> is an actual date and time in ISO format).
Note: To ensure proper paths to Perl libraries and resource files, the `start.pl` script should be run from within the app/ directory.

8. Using a web browser, navigate to the given address: http://127.0.0.1:3000

9. A prompt for login credentials should appear. Assuming default settings are used, the initial login credentials will be

   **User name** admin  
   **Password** password

10. Change the login credentials.

    See also:  
    *Traffic Ops - Using*

    **Test Cases**

    Use `prove` (should be installed with Perl) to execute test cases. Execute after a carton install of all required dependencies:

    - To run the Unit Tests: `prove -qrp app/t/`
    - To run the Integration Tests: `prove -qrp app/t_integration/`

Note: As progress continues on moving Traffic Ops to run entirely in Go, the number of passing tests has steadily decreased. This means that the tests are not a reliable way to test Traffic Ops, as they are expected to fail more and more as functionality is stripped from the Perl codebase.

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. Be sure the integration tests have been run

2. Start the Traffic Ops server. The `MOJO_MODE` environment variable should be set to the name of the environment that has been loaded.

   ```
   # 73: Example Startup
   export MOJO_MODE=integration
   cd app/
   bin/start.pl
   ```
3. Using a web browser, navigate to the address Traffic Ops is serving, e.g. http://127.0.0.1:3000 for default settings

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 customizable manner. There are two types of extensions:

**Check Extensions** These allow you to add custom checks to the Monitor → Cache Checks view.

**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

See also:

For more information see *Managing Traffic Ops Extensions*.

**Extensions at Runtime**

The search path for Data Source Extensions depends on the configuration of the PERL5LIB environment variable, which is pre-configured in the Traffic Ops start scripts. All Check Extensions must be located in $TO_HOME/bin/checks

```bash
# 74: Example PERL5LIB Configuration

export PERL5LIB=/opt/traffic_ops_extensions/private/lib/
  Extensions:/opt/traffic_ops/app/lib/Extensions/
  TrafficStats
```

To prevent Data Source Extensions namespace collisions within Traffic Ops all Data Source Extensions should follow the package naming convention ‘Extensions::<ExtensionName>’

**TrafficOpsRoutes.pm**

Traffic Ops accesses each extension through the addition of a URL route as a custom hook. These routes will be defined in a file called TrafficOpsRoutes.pm that should be present in the top directory of your Extension. The routes that are defined should follow the Mojolicious route conventions.
Development Configuration

To incorporate any custom *Data Source Extensions* during development set your `PERL5LIB` environment variable with any number of colon-separated directories with the understanding that the `PERL5LIB` search order is from left to right through this list. Once Perl locates your custom route or Perl package/class it ‘pins’ on that class or Mojolicious Route and doesn’t look any further, which allows for the developer to override Traffic Ops functionality.

4.1.3 Traffic Ops - Go

Traffic Ops is currently in the process of being re-written in Go, which has its own dependencies, build process and source tree. To develop for the Go version of Traffic Ops you **must have cloned the Traffic Control repository into** `GOPATH/src/github.com/apache/trafficcontrol/` **or** linked your downloaded repository to that path. Either way, you must build from there.

**Building**

To fetch dependencies, it should suffice to run `go get -v` from the `traffic_ops/traffic_ops/golang` directory. Then simply run `go build .` from the same directory to actually run the build\(^1\).

4.1.4 traffic_ops_ort package

This package is meant to fully implement the Traffic Ops Operational Readiness Test - which was originally written in a single, chickenscratch Perl script. When the `main()` function is run, it acts (more or less) exactly like that legacy script, with the ability to set system configuration files and start, stop, and restart HTTP cache servers etc.

This package provides an executable script named `traffic_ops_ort`

**Usage**

There are two main ways to invoke `traffic_ops_ort`. The first method uses what’s referred to as the “legacy call signature” and is meant to match the Perl command line arguments.

\(^1\) This should also run the `go get` command if this has not already been done, but it is recommended to do this in two different steps for ease of debugging.
The second method - called the “new call signature” - aims to reduce the complexity of the ORT command line. Rather than require a URL and “login string” for connecting and authenticating with the Traffic Ops server, these pieces of information are optional and may be provided by the --to_url, -u/--to_user, and -p/--to_password options, respectively. If they are NOT provided, then their values will be obtained from the TO_URL, TO_USER, and TO_PASSWORD environment variables, respectively. Note that traffic_ops_ort cannot be run using the new call signature without providing a definition for each of these, either on the command line or in the execution environment.

# 76: New call signature

```
traffic_ops_ort [-k] [-h] [-v] [--dispersion DISP] [--login_˓→ dispersion DISP]
   [--retries RETRIES] [--wait_for_parents INT] [--rev_proxy_
   → disable]
   → url URL] MODE
```

These two call signatures should not be mixed, and traffic_ops_ort will exit with an error if they are.

### Arguments and Flags

- **-h, --help**
  
  Print usage information and exit

- **-v, --version**
  
  Print version information and exit

- **-k, --insecure**
  
  An optional flag which, when used, disables the checking of SSL certificates for validity

- **--dispersion DISP**
  
  Wait a random number between 0 and DISP seconds before starting. This option only has any effect if MODE is SYNCDS. (Default: 300)

- **--login Dispersion DISP**
  
  Wait a random number between 0 and DISP seconds before authenticating with Traffic Ops. (Default: 0)

- **--retries RETRIES**
  
  If connection to Traffic Ops fails, retry RETRIES times before giving up (Default: 3).

- **--wait_for_parents INT**
  
  If INT is anything but 0, do not apply updates if parents of this server have pending updates. This option requires an integer argument for legacy compatibility reasons; 0 is considered False, anything else is True. (Default: 1)

- **--rev_proxy_disable**
  
  Make requests directly to the Traffic Ops server, bypassing a reverse proxy if one exists.
--ts_root PATH
An optional flag which, if present, specifies the absolute path to the install directory of Apache Traffic Server. A common alternative to the default is /opt/trafficserver. (Default: /)

MODE
Specifies traffic_ops_ort’s mode of operation. Must be one of:

REPORT Runs as though the mode was BADASS, but doesn’t actually change anything on the system.

Tip: This is normally useful with a verbose LOG_LEVEL to check the state of the system

INTERACTIVE Runs as though the mode was BADASS, but asks the user for confirmation before making changes

REVALIDATE Will not restart Apache Traffic Server, install packages, or enable/disable system services and will exit immediately if this server does not have revalidations pending. Also, the only configuration file that will be updated is regex_revalidate.config.

SYNCDS Will not restart Apache Traffic Server, and will exit immediately if this server does not have updates pending. Otherwise, the same as BADASS

BADASS Applies all pending configuration in Traffic Ops, and attempts to solve encountered problems when possible. This will install packages, enable/disable system services, and will start or restart Apache Traffic Server as necessary.

LOG_LEVEL, -l LOG_LEVEL, --log_level LOG_LEVEL
Sets the verbosity of output provided by traffic_ops_ort. This argument is positional in the legacy call signature, but optional in the new call signature, wherein it has a default value of “WARN”. Must be one of (case-insensitive):

NONE Will output nothing, not even fatal errors.

CRITICAL Will only output error messages that indicate an unrecoverable error.

FATAL A synonym for “CRITICAL”

ERROR Will output more general errors about conditions that are causing problems of some kind.

WARN In addition to error information, will output warnings about conditions that may cause problems, or possible misconfiguration.

INFO Outputs informational messages about what traffic_ops_ort is doing as it progresses.

DEBUG Outputs detailed debug information, including stack traces.

Note: Not all stack traces indicate problems with traffic_ops_ort. Stack traces are printed whenever an exception is encountered, whether or not it could be
handled.

**TRACE** A synonym for “DEBUG”

**ALL** A synonym for “DEBUG”

**Note:** All logging is sent to STDERR. INTERACTIVE MODE prompts are printed to STDOUT

---

**TO_URL, --to_url TO_URL**

This must be at minimum an FQDN that resolves to the Traffic Ops server, but may optionally include the schema and/or port number. E.g. `https://trafficops.infra.ciab.test:443`, `https://trafficops.infra.ciab.test`, `trafficops.infra.ciab.test:443`, and `trafficops.infra.ciab.test` are all acceptable, and in fact are all equivalent. When given a value without a schema, HTTPS will be the assumed protocol, and when a port number is not present, 443 will be assumed except in the case that the schema is provided and is `http://` (case-insensitive) in which case 80 will be assumed.

This argument is positional in the legacy call signature, but is optional in the new call signature. When the new call signature is used and this option is not present on the command line, its value will be obtained from `TO_URL`. Note that `traffic_ops_ort` cannot be run using the new call signature unless this value is defined, either on the command line or in the execution environment.

**LOGIN**

The information used to authenticate with Traffic Ops. This must consist of a username and a password, delimited by a colon (:). E.g. `admin:twelve`. This argument is not used in the new call signature, instead `-u/-to_user` and `-p/-to_password` are used to separately set the authentication user and password, respectively.

**Warning:** The first colon found in this string is considered the delimiter. There is no way to escape the delimiter. This effectively means that usernames containing colons cannot be used to authenticate with Traffic Ops, though passwords containing colons should be fine.

**-u USER, --to_user USER**

Specifies the username of the user as whom to authenticate when connecting to Traffic Ops. This option is only available using the new call signature. If not provided when using said new call signature, the value will be obtained from the `TO_USER` environment variable. Note that `traffic_ops_ort` cannot be run using the new call signature unless this value is defined, either on the command line or in the execution environment.

**-p PASSWORD, --to_password PASSWORD**

Specifies the password of the user identified by `TO_USER` (or `-u/-to_user` if overridden) to use when authenticating to Traffic Ops. This option is only available using the new call signature. If not provided when using said new call signature,
the value will be obtained from the `TO_PASSWORD` environment variable. Note that `traffic_ops_ort` cannot be run using the new call signature unless this value is defined, either on the command line or in the execution environment.

**Environment Variables**

**TO_URL**

Should be set to the URL of a Traffic Ops server. This doesn’t need to be a full URL, an FQDN will do just as well. It may also omit the port number on which the Traffic Ops server listens for incoming connections - port 443 will be assumed unless `TO_URL` is prefixed by `http://` (case-insensitive), in which case port 80 will be assumed. The value of this environment variable will only be considered if `traffic_ops_ort` was invoked using the new call signature, which allows it to be overridden on the command line by the value of `--to_url`.

**TO_USER**

The username to use when authenticating to the Traffic Ops server. The value of this environment variable will only be considered if `traffic_ops_ort` was invoked using the new call signature, which allows it to be overridden on the command line by the value of `--to_user`.

**TO_PASSWORD**

The password to use when authenticating to the Traffic Ops server. The value of this environment variable will only be considered if `traffic_ops_ort` was invoked using the new call signature, which allows it to be overridden on the command line by the value of `--to_password`.

**Strings with Special Meaning to ORT**

When processing configuration files, if `traffic_ops_ort` encounters any of the strings in the *Replacement Strings* table it will perform the indicated replacement. This means that these strings can be used to create templates in *Profile Parameters* and certain *Delivery Service* configuration fields.
Table 47: Replacement Strings

<table>
<thead>
<tr>
<th>String</th>
<th>Replaced With</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CACHE_IPV4</strong></td>
<td>The IPv4 address of the cache server on which traffic_ops_ort is running.</td>
</tr>
<tr>
<td>__FULL_HOSTNAME</td>
<td>The full hostname (i.e. including the full domain to which it belongs) of the cache server on which traffic_ops_ort is running.</td>
</tr>
<tr>
<td><strong>HOSTNAME</strong></td>
<td>The (short) hostname of the cache server on which traffic_ops_ort is running.</td>
</tr>
<tr>
<td><strong>RETURN</strong></td>
<td>A newline character (\n).</td>
</tr>
<tr>
<td>__SERVER_TCP_PORT</td>
<td>If the cache server on which traffic_ops_ort is being run has a TCP port configured to something besides 80, this will be replaced with that TCP port value. If it is set to “80”, this string will simply be removed, NOT replaced with ANYTHING.</td>
</tr>
<tr>
<td>##OVERRIDE##</td>
<td>This string is only valid in the content of files named “remap.config”. It is further described in Remap Override</td>
</tr>
</tbody>
</table>

Deprecated since version ATCv4.0: The use of __RETURN__ in lieu of a true newline character is (finally) no longer necessary, and the ability to do so will be removed in the future.

Note: There is currently no way to indicate that a server’s IPv6 address should be inserted - only IPv4 is supported.

Remap Override

Warning: The ANY_MAP ##OVERRIDE## special string is a temporary solution and will be deprecated once Delivery Service Versioning is implemented. For this reason it is suggested that it not be used unless absolutely necessary.

The ##OVERRIDE## template string allows the Delivery Service Raw Remap Text field to override to fully override the Delivery Service’s line in the remap.config ATS configuration file, generated by Traffic Ops. The end result is the original, generated line commented out, prepended with ##OVERRIDDEN## and the ##OVERRIDE## rule is activated in its place. This behavior is used to incrementally deploy plugins used in this configuration file. Normally, this entails cloning the Delivery Service that will have the plugin, ensuring it is assigned to a subset of the cache servers that serve the Delivery Service content, then using this ##OVERRIDE## rule to create a remap.config rule that will use the plugin, overriding the normal rule. Simply grow the subset over time at the desired rate to slowly deploy the plugin. When it encompasses all cache servers that serve the original Delivery Service’s content, the “override Delivery Service” can be deleted and the original can use a non-##OVERRIDE## Raw Remap Text to add the plugin.
# 77: Example of Remap Override

```plaintext
# This is the original line as generated by Traffic Ops
map http://from.example.com/ http://to.example.com/

# This is the raw remap text as configured on the delivery service
##OVERRIDE##
map http://from.example.com/ http://to.example.com/
→ some_plugin.so

# The resulting content is what actually winds up in the remap.
##OVERRIDE##
config file:
# ##OVERRIDDEN##
map http://from.example.com/ http://to.example.com/
→ some_plugin.so

##OVERRIDE##
map http://from.example.com/ http://to.example.com/

##OVERRIDDEN##
map http://from.example.com/ http://to.example.com/
```

**Warning:** The “from” URL must exactly match for this to properly work (e.g. including trailing URL ‘/’), otherwise ATS may fail to initialize or reload while processing `remap.config`.

**Tip:** To assist in troubleshooting, it is strongly recommended that any `##OVERRIDE##` rules in use should be documented on the original `Delivery Service`.

---

### Module Contents

**traffic_ops_ort**

`doMain(args)`

Runs the main routine based on the parsed arguments to the script

**Parameters**

- `args` (Namespace) – A parsed argument list as returned from `argparse.ArgumentParser.parse_args()`

**Return type**

`int`

**Returns**

An exit code for the script.

**Raises**

`AttributeError` – when the namespace is missing required arguments

`traffic_ops_ort.main()`

The ORT entrypoint, parses argv before handing it off to `doMain()`.

**Return type**

`int`

**Returns**

An exit code for `traffic_ops_ort`

---

### Submodules

190 Chapter 4. Developer's Guide
traffic_ops_ort.configuration module

This module is responsible for holding information related to the configuration of the ORT script; it has constants that hold and set up the log level, run modes, Traffic Ops login credentials etc.

```python
class traffic_ops_ort.configuration.Configuration(args)
    Bases: object

    Represents a configured state for traffic_ops_ort.

class Modes
    Bases: enum.IntEnum

    Enumerated representations for run modes for valid configurations.

    BADASS = 4
        Apply all settings specified in Traffic Ops, with no restrictions

    INTERACTIVE = 1
        Ask for user confirmation before modifying the system

    REPORT = 0
        Do nothing, only report what would be done

    REVALIDATE = 2
        Only check for configuration file changes and content revalidations

    SYNCDS = 3
        Check for and apply Delivery Service changes

ServerInfo = None
    Holds a reference to a to_api.ServerInfo object used by this configuration - must be set manually.

TOURL
    Convenience function to construct a full URL out of whatever information was given at runtime

    Return type str
    Returns The configuration’s URL which points to its Traffic Ops server instance

Note: This is totally constructed from information given on the command line; the resulting URL may actually point to a reverse proxy for the Traffic Ops server and not the server itself.
```

```python
api = None
    Holds a reference to a to_api.API object used by this configuration - must be set manually.

fullHostname
    Convenience accessor for the full hostname of this server
```
Return type str

Returns The hostname of this server as detected by platform.

node()

shortHostname
Convenience accessor for the short hostname of this server

Return type str

Returns The (short) hostname of this server as detected by platform.

node()

traffic_ops_ort.configuration.DISTRO = 'ubuntu'
contains identifying information about the host system’s Linux distribution

traffic_ops_ort.configuration.LOG_FORMAT = '%(levelname)s: %(asctime)s line %(lineno)d in %(module)s.%(funcName)s: %(message)s'
A format specifier for logging output. Propagates to all imported modules.

class traffic_ops_ort.configuration.LogLevels
    Bases: enum.IntEnum

Enumerated Log levels

ALL = 0
    Outputs all logging information

CRITICAL = 50
    Synonym for FATAL

DEBUG = 10
    Outputs debugging information as well as all higher log levels

ERROR = 40
    Errors - but not more verbose warnings - will be output

FATAL = 50
    Outputs only reasons why the script exited prematurely

INFO = 20
    Outputs informational messages as well as all higher log levels

NONE = 51
    Silent mode - no output at all

TRACE = 0
    Synonym for ALL

WARN = 30
    Outputs warnings, errors and fatal messages

traffic_ops_ort.configuration.parseTOURL(url, verify)
Parses and verifies the passed URL and breaks it into parts for the caller

Parameters

* url (str) – At minimum an FQDN for a Traffic Ops server, but can include schema and port number
• **verify** *(bool)* – Whether or not to verify the server’s SSL certificate

**Return type** Tuple[bool, str, int]

**Returns** Whether or not the Traffic Ops server uses SSL (http vs https), the server’s FQDN, and the port on which it listens

**Raises** ValueError – if *url* does not point at a valid HTTP server or is incorrectly formatted

`traffic_ops_ort.configuration.parseTSRoot(tsroot)`

Parses and validates a given path as a path to the root of an Apache Traffic Server installation

**Parameters**

*tsroot* *(str)* – The relative or absolute path to the root of this server’s ATS installation

**Raises** ValueError – if *tsroot* is not an existing path, or does not contain the ATS binary

**Return type** str

`traffic_ops_ort.configuration.setLogLevel(level)`

Parses a string to return the requested LogLevels member, to which it will then set the global logging level.

**Parameters**

*level* *(str)* – the name of a LogLevels enum constant

**Raises** ValueError – if *level* cannot be parsed to an actual LogLevel

**traffic_ops_ort.main_routines module**

This module is meant to run the main routines of the script, and performs a variety of operations based on the run mode.

**exception** `traffic_ops_ort.main_routines.ORTException`

Bases: Exception

Signifies an ORT related error

`traffic_ops_ort.main_routines.STATUS_FILE_DIR = '/opt/ort/status'`

A constant that holds the absolute path to the status file directory

`traffic_ops_ort.main_routines.deleteOldStatusFiles(myStatus, conf)`

Attempts to delete any and all old status files

**Parameters**

*myStatus* *(str)* – the current status - files by this name will not be deleted

*conf*(Configuration) – An object containing the configuration of `traffic_ops_ort`
Raises

- `ConnectionError` – if there’s an issue retrieving a list of statuses from Traffic Ops
- `OSError` – if a file cannot be deleted for any reason

`traffic_ops_ort.main_routines.processConfigurationFiles(conf)`
Updates and backs up all of a server’s configuration files.

Parameters `conf` (*Configuration*) – An object containing the configuration for `traffic_ops_ort`

Return type `bool`

Returns whether or not the configuration changes were successful

`traffic_ops_ort.main_routines.processPackages(conf)`
Manages the packages that Traffic Ops reports are required for this server.

Parameters `conf` (*Configuration*) – An object containing the configuration of `traffic_ops_ort`

Return type `bool`

Returns whether or not the package processing was successfully completed

`traffic_ops_ort.main_routines.processServices(conf)`
Manages the running processes of the server, according to an ancient system known as ‘chkconfig’

Parameters `conf` (*Configuration*) – An object containing the configuration for `traffic_ops_ort`

Return type `bool`

Returns whether or not the service processing was completed successfully

`traffic_ops_ort.main_routines.revalidateState(conf)`
Checks the revalidation status of this server in Traffic Ops

Parameters `conf` (*Configuration*) – The script’s configuration

Return type `bool`

Returns whether or not this server has a revalidation pending

Raises `ConnectionError` – when something goes wrong communicating with Traffic Ops

`traffic_ops_ort.main_routines.run(conf)`
This function is the entrypoint into the script’s main flow from `traffic_ops_ort.doMain()` It runs the appropriate actions depending on the run mode.

Parameters `conf` (*Configuration*) – An object that holds the script’s configuration

Return type `int`
Returns an exit code for the script

traffic_ops_ort.main_routines.setStatusFile(conf)
Attempts to set the status file according to this server’s reported status in Traffic Ops.

Warning: This will create the directory ‘/opt/ORTstatus’ if it does not exist, and may delete files there without warning!

Parameters conf (Configuration) – An object that contains the configuration for traffic_ops_ort

Return type bool

Returns whether or not the status file could be set properly

traffic_ops_ort.main_routines.syncDSState(conf)
Queries Traffic Ops for the Delivery Service’s sync state

Parameters conf (Configuration) – The script’s configuration

Return type bool

Returns whether or not an update is needed

Raises ConnectionError – when something goes wrong communicating with Traffic Ops

traffic_ops_ort.packaging module

This module deals with managing packages installed on the host system. It attempts to do so in a distribution-agnostic way, but actually only support a strict set of distributions with well-known package managers.

class traffic_ops_ort.packaging.Package(pkg)
Bases: object

Represents a package installed (or about to be installed) on the host system

checkInstallList()

install(conf)
Installs this package.

Parameters conf (Configuration) – An object containing the configuration for traffic_ops_ort

Return type int

Returns the exit code of the install process

installArgs = ['/usr/bin/apt-get', 'install', '-y']

isInstalled()
Checks if this package is already present on the system
Return type bool
Returns whether or not the package could be found

name = None
The package’s name

uninstall (conf)
Uninstalls this package. I have no idea how one would make use of this from within ATC...

Return type int
Returns the exit code of the uninstall process

uninstallArgs = ['/usr/bin/apt-get', 'purge', '-y']

version = None
Optionally, a specific version of the package

traffic_ops_ort.to_api module

This module contains functionality for dealing with the Traffic Ops ReST API. It extends the class provided by the official Apache Traffic Control Client.

class traffic_ops_ort.to_api.API (conf)
Bases: trafficops.tosession.TOSession

This class extends trafficops.tosession.TOSession to provide some ease-of-use functionality for getting things needed by ORT.

VERSION = '1.4'
This should always be the latest API version supported - note this breaks compatibility with older ATC versions. Go figure.

getMyChkconfig()
Fetches the ‘chkconfig’ for this server

Return type List[dict]
Returns An iterable list of ‘chkconfig’ entries

Raises ConnectionError – when something goes wrong communicating with Traffic Ops

getMyConfigFiles (conf)
Fetches configuration files constructed by Traffic Ops for this server

Note: This function will set the serverInfo attribute of the object passed as the conf argument to an instance of ServerInfo with the provided information.

Parameters conf (Configuration) – An object that represents the configuration of traffic_ops_ort
Return type List[dict]

Returns A list of constructed config file objects

Raises ConnectionError – when something goes wrong communicating with Traffic Ops

getMyPackages()
 Fetches a list of the packages specified by Traffic Ops that should exist on this server.

Return type List[Package]

Returns all of the packages which this system must have, according to Traffic Ops.

Raises ConnectionError – if fetching the package list fails

getMyStatus()
 Fetches the status of this server as set in Traffic Ops

Raises ConnectionError – if fetching the status fails

Return type str

Returns the name of the status to which this server is set in the Traffic Ops configuration

getMyUpdateStatus()
 Gets the update status of a server.

Raises ConnectionError – if something goes wrong communicating with the server

Return type dict

Returns An object representing the API’s response

getRaw (path)
 This gets the API response to a “raw” path, meaning it will queried directly without a /api/1.x prefix. Because the output structure of the API response is not known, this returns the response body as an unprocessed string rather than a Python object via e.g. Munch.

Parameters path (str) – The raw path on the Traffic Ops server

Return type str

Returns The API response payload

Raises ConnectionError – When something goes wrong communicating with the Traffic Ops server

updateTrafficOps (mode)
 Updates Traffic Ops’s knowledge of this server’s update status.

Parameters mode (Modes) – The current run-mode of traffic_ops_ort
traffic_ops_ort.to_api.API_LOGGERS = {'error': <function <lambda>>, 'info': <function <lambda>>, 'success': <function <lambda>>, 'warning': <function <lambda>>}
Maps Traffic Ops alert levels to logging levels

traffic_ops_ort.to_api.CACHED_STATUSES = []
Caches the names of statuses supported by Traffic Ops

class traffic_ops_ort.to_api.ServerInfo(raw)
    Bases: object
    Holds information about a server, as returned by the Traffic Ops API api/1.x/
    servers/<hostname>/configfiles/ats endpoint

    cdnId = -1
    A database primary key for the CDN to which this server is assigned

    cdnName = ''
    The name of the CDN to which this server is assigned

    profileId = -1
    A database primary key for this server’s profile’s information

    profileName = ''
    The name of the profile in use by this server

    sanitize(fmt, hostname)
    Sanitizes an input string with the passed hostname information

    Parameters
    • fmt (str) – The string to be sanitized
    • hostname (Tuple[str, str]) – A tuple containing the short and
      full hostnames of the server

    Return type str
    Returns The string fmt after sanitization

    serverId = -1
    A database primary key for this server’s information

    serverIpv4 = ''
    This server’s IPv4 address

    serverName = ''
    This server’s short hostname

    serverTcpPort = 80
    The port on which the caching proxy of this server listens

    toRevProxyUrl = ''
    This specifies the url of a reverse proxy that should be used for future requests to
    the Traffic Ops API - if present.

    toUrl = ''
    The Traffic Ops URL... not sure what that’s for...
traffic_ops_ort.utils module

This module contains miscellaneous utilities, typically dealing with string manipulation or user input/output.

traffic_ops_ort.utils.getJSONResponse(uri, cookies=None, verify=True)

Retrieves a JSON object from some HTTP API.

Parameters

- **uri** (str) – The URI to fetch.
- **cookies** (Optional[dict]) – A dictionary of cookie names mapped to values.
- **verify** (bool) – If this is True, the SSL keys will be verified during handshakes with ‘https’ URIs.

Return type **dict**

Returns The decoded JSON object.

Raises

- **ConnectionError** – when an error occurs trying to communicate with the server.
- **ValueError** – when the request completes successfully, but the response body does not represent a JSON-encoded object.

traffic_ops_ort.utils.getTextResponse(uri, cookies=None, verify=True)

Gets the plaintext response body of an HTTP GET request.

Parameters

- **uri** (str) – The full path to a resource for the request.
- **cookies** (Optional[dict]) – An optional dictionary of cookie names mapped to values.
- **verify** (bool) – If True, the SSL keys used to communicate with the full URI will be verified.

Raises

- **ConnectionError** – when an error occurs trying to communicate with the server.
- **ValueError** – if the server’s response cannot be interpreted as a UTF-8 string - e.g. when the response body is raw binary data but the response headers claim it’s UTF-16.

Return type **str**

traffic_ops_ort.utils.getYesNoResponse(prompt, default=None)

Utility function to get an interactive yes/no response to the prompt `prompt`.

4.1. Developer’s Guide
Parameters

- **prmpt** *(str)* – The prompt to display to users

- **default** *(Optional[str])* – The default response; should be one of ’y’, ”yes”, ’n’ or ”no” (case insensitive)

**Raises** `AttributeError` – if ‘prmpt’ and/or ‘default’ is/are not strings

**Return type** bool

**Returns** the parsed response as a boolean

---

### 4.1.5 Traffic Portal

**Introduction**

Traffic Portal is an AngularJS client served from a Node.js web server designed to consume the Traffic Ops API. Traffic Portal is the official replacement for the legacy Traffic Ops UI.

**Software Requirements**

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

- Ruby Devel 2.0.x or above
- Compass 1.0.x or above
- Node.js 6.0.x or above
- Bower 1.7.9 or above
- Grunt CLI 1.2.0 or above
- Access to a working instance of Traffic Ops

**Note:** The Traffic Portal consumes the Traffic Ops API. Modify traffic_portal/conf/config.js to specify the location of Traffic Ops.

**Traffic Portal Project Tree Overview**

- `traffic_control/traffic_portal/app/src` - contains HTML, JavaScript and SCSS (Sassy CSS) source files.

**Installing The Traffic Portal Developer Environment**

1. Clone the Traffic Control Repository
2. Navigate to the `traffic_portal` subdirectory of your cloned repository.

3. Run `npm install` to install application dependencies into `traffic_portal/node_modules`. Only needs to be done the first time unless `traffic_portal/package.json` changes.

4. Run `bower install` to install client-side dependencies into `traffic_portal/app/bower_components`. Only needs to be done the first time unless `traffic_portal/bower.json` changes.

5. Run `grunt` to package the application into `traffic_portal/app/dist`, start a local HTTPS server (Express), and start a file watcher.

6. Valid SSL certificates and keys are needed for Traffic Portal to run. Generate these (e.g. using this SuperUser answer) and edit the `traffic_portal/conf/config.js` file to reflect their names and locations.

7. Navigate to `http(s)://localhost:[port|sslPort defined in traffic_portal/conf/config.js]`

### 4.1.6 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 server availability is determined by Traffic Monitor; consequently Traffic Router polls Traffic Monitor for its configuration and cache server 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 response to redirect to the nearest cache server. 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 `foo.deliveryservice.somecdn.net`. Traffic Router is comprised of seven separate Maven modules:

- **shared**: A reusable utility JAR for defining Delivery Service Certificates
- **configuration**: A reusable JAR defining the ConfigurationListener interface
- **connector**: A JAR that overrides Tomcat’s standard Htp11Protocol Connector class and allows Traffic Router to delay opening listen sockets until it is in a state suitable for routing traffic
- **geolocation**: Submodule for defining geolocation services
- **neustar**: A JAR that provides a bean “neustarGeolocationService” that implements the GeolocationService interface defined in the geolocation maven submodule, which can optionally be added to the build of Traffic Router
- **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 the API
• build - A simple Maven project which gathers the artifacts from the 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 >= 8.0 (OpenJDK suggested, but not required)
• OpenSSL >= 1.0.2
• APR >= 1.4.8-3
• Tomcat Native >= 1.2.16
• Not Tomcat - You do not need a Tomcat installation for development. An embedded version is launched for development testing instead.

Traffic Router Project Tree Overview

• traffic_control/traffic_traffic_router/ - base directory for Traffic Router
  – 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
      · lib/systemd/system/traffic_router.service - Unit script for launching the Traffic Router with Tomcat
      · conf/ - All of the required configuration files for running the traffic_router web application, including those needed for Tomcat
      · java/ - Java source code for Traffic Router Core
      · resources/ - Resources pulled in during an RPM build
      · scripts/ - Scripts used by the RPM build process
      · webapp/ - Java “webapp” resources
      · var/log/ - location of all the Traffic Router runtime logs
    * src/test - Test source directory for Traffic Router Core
      · conf/ - Minimal Configuration files that make it possible to run JUnit 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 Router Developer environment:

1. Clone the traffic_control repository using Git.
2. Change directories into traffic_control/traffic_router.
3. Set the environment variable TRAFFIC_MONITOR_HOSTS to be a semicolon delimited list of Traffic Monitors that can be accessed during integration tests OR install the traffic_monitor.properties file.
4. Additional configuration is set using the below files:
   • copy core/src/test/conf/dns.properties from core/src/main/conf/
   • copy core/src/test/conf/http.properties from core/src/main/conf/
   • copy core/src/test/conf/log4j.properties from core/src/main/conf/
   • copy core/src/test/conf/traffic_monitor.properties from core/src/main/conf/ and then edit the traffic_monitor.bootstrap.hosts property
   • core/src/test/conf/traffic_ops.properties holds the credentials for accessing Traffic Ops. Copy it from core/src/main/conf/ and then edit the credentials as appropriate for the Traffic Ops instance you will be using.
   • Default configuration values now reside in core/src/main/webapp/WEB-INF/applicationContext.xml

Note: These values may be overridden by creating and/or modifying the property files listed in core/src/main/resources/applicationProperties.xml
Note: Pre-existing properties files are still honored by Traffic Router. For example traffic_monitor.properties may contain the FQDN and port of the Traffic Monitor instance(s), separated by semicolons as necessary (do not include scheme e.g. http://)

1. Import the existing git repository as projects into your IDE (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

2. From the terminal or your IDE, run mvn clean verify from the traffic_router directory. This will run a series of integration tests and will temporarily start and embedded version of Traffic Router and a ‘fake’ simulated instance of Traffic Monitor.

3. Start the embedded Tomcat instance for Core from within your IDE by following these steps:
   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

Once running, the Traffic Router API is available at http://localhost:3333, the HTTP routing interface is available on http://localhost:8888 and HTTPS is available on http://localhost:8443. The DNS server and routing interface is available on localhost:53 via TCP and UDP.
Manual Testing

Look up the URL for a test HTTP Delivery Service in Traffic Ops and then make a request. When Traffic Router is running and used as a resolver for the host in the Delivery Service URL, the requested origin content should be found through an Edge-tier cache server.

# 78: Example Test for an HTTP Delivery Service

```bash
root@enroller:/shared/enroller# curl -skvL http://video.demo1.mycdn.ciab.test/
* Trying fc01:9400:1000:8::60...
* TCP_NODELAY set
* Connected to video.demo1.mycdn.ciab.test (fc01:9400:1000:8::60) port 80 (#0)
> GET / HTTP/1.1
> Host: video.demo1.mycdn.ciab.test
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 302 Found
< Location: http://edge.demo1.mycdn.ciab.test/
< Content-Length: 0
< Date: Wed, 16 Jan 2019 21:52:14 GMT
<
* Curl_http_done: called premature == 0
* Connection #0 to host video.demo1.mycdn.ciab.test left intact
* Issue another request to this URL: 'http://edge.demo1.mycdn.ciab.test/
* Trying fc01:9400:1000:8::100...
* TCP_NODELAY set
* Connected to edge.demo1.mycdn.ciab.test (fc01:9400:1000:8::100) port 80 (#1)
> GET / HTTP/1.1
> Host: edge.demo1.mycdn.ciab.test
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html
< Accept-Ranges: bytes
< ETag: "1473249267"
< Cache-Control: public, max-age=300
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Accept, Origin, Content-Type
< Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS
< Content-Length: 1881
< Date: Wed, 16 Jan 2019 21:52:15 GMT
< Server: ATS/7.1.4
< Age: 1
```

(continues on next page)
< Via: http/1.1 mid.infra.ciab.test (ApacheTrafficServer/7.1.4 [uScMsSfWpSeN:t cCMi p sS]), http/1.1 edge.infra.ciab.test (ApacheTrafficServer/7.1.4 [uScMsSfWpSeN:t cCMi p sS])< Connection: keep-alive

<!DOCTYPE html>
<!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. -->
<html lang="en">
<head>
<title>CDN In a Box</title>
<meta charset="utf-8"/>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width; height=device-height; initial-scale=1"/>
<link rel="shortcut-icon" href="/tc_logo.svg"/>
<meta name="author" content="Apache"/>
<meta name="creator" content="Apache"/>
<meta name="publisher" content="Apache"/>
<meta name="description" content="A simple test origin for Apache Traffic Control"/>
<style type="text/css">
html {
  height: 100vh;
  width: 100vw;
}

body {
  text-align: center;
  background-image: url(/tc_logo.svg);
  background-color: black;
  background-position: center;
  background-repeat: no-repeat;
  background-size: 25%;
  font-family: "Ubuntu Mono", "Consolas", sans-serif;
} (continues on next page)
Test Cases

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

RPM Packaging

Running `mvn package` on a Linux-based distribution will trigger the build process to create the Traffic Router RPM and the Traffic Router `.war` file, but will not run the integration tests, so it is a good way to update those artifacts quickly during development. But the preferred way to build the Traffic Router RPMs is by following the instructions in *Building Traffic Control*.

API

*Traffic Router API*
Traffic Router API

By default, Traffic Router serves its API via HTTP (not HTTPS) on port 3333. This can be configured in /opt/traffic_router/conf/server.xml or by setting a Parameter named api.port with configfile server.xml on the Traffic Router’s Profile.

Traffic Router API endpoints only respond to GET requests.

/crs/stats

General stats.

Request Structure

# 79: Request Example

```
GET /crs/stats HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
```

Response Structure

# 80: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 15 Jan 2019 21:02:09 GMT

{
    "app": {
        "buildTimestamp": "2019-01-10",
        "name": "traffic_router",
        "deploy-dir": "/opt/traffic_router",
        "git-revision": "437e9df81",
        "version": "3.0.0"
    },
    "stats": {
        "dnsMap": {},
        "httpMap": {
            "video.demo1.mycdn.ciab.test": {
                "czCount": 0,
                "geoCount": 0,
                "deepCzCount": 0,
                "missCount": 0,
                "dsrCount": 0,
            }
        }
    }
}
```


```
"errCount": 0,
"staticRouteCount": 0,
"fedCount": 0,
"regionalDeniedCount": 0,
"regionalAlternateCount": 0
}
),
"totalDnsCount": 0,
"totalHttpCount": 1,
"totalDsMissCount": 0,
"appStartTime": 1547584831677,
"averageDnsTime": 0,
"averageHttpTime": 1547584863270,
"updateTracker": {
    "lastHttpsCertificatesCheck": 1547586068932,
    "lastGeolocationDatabaseUpdaterUpdate": 1547584858917,
    "lastCacheStateCheck": 1547586128932,
    "lastCacheStateChange": 1547584867102,
    "lastNetworkUpdaterUpdate": 1547584857484,
    "lastHttpsCertificatesUpdate": 1547586071079,
    "lastSteeringWatcherUpdate": 1547584923514,
    "lastConfigCheck": 1547586127344,
    "lastConfigChange": 1547584863406,
    "lastNetworkUpdaterCheck": 1547584857465,
    "lastGeolocationDatabaseUpdaterCheck": 1547584858906,
    "lastFederationsWatcherUpdate": 1547584863433,
    "lastHttpsCertificatesFetchSuccess": 1547586069070,
    "lastSteeringWatcherCheck": 1547586124630,
    "lastFederationsWatcherCheck": 1547586124584,
    "lastHttpsCertificatesFetchAttempt": 1547586068932
}
}}

/crs/stats/ip/{{IP}}

Geolocation information for an IPv4 or IPv6 address.

**Request Structure**

Table 48: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>The IP address for which statics will be returned. May be IPv4 or IPv6</td>
</tr>
</tbody>
</table>
# 81: Request Example

```
GET /crs/stats/ip/255.255.255.255 HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
```

## Response Structure

# 82: Response Example

```
HTTP/1.1 200 OK
Content-Disposition: inline;filename=f.txt
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 15 Jan 2019 21:06:09 GMT

{
   "locationByGeo": {
      "city": "Woodridge",
      "countryCode": "US",
      "latitude": "41.7518",
      "postalCode": "60517",
      "countryName": "United States",
      "longitude": "-88.0489"
   },
   "locationByFederation": "not found",
   "requestIp": "69.241.118.34",
   "locationByCoverageZone": "not found"
}
```

## /crs/locations

A list of configured Cache Groups to which the Traffic Router is capable of routing client traffic.

## Request Structure

# 83: Request Example

```
GET /crs/locations HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
```
Response Structure

locations An array of the names of Cache Groups to which this Traffic Router is capable of routing client traffic

# 84: Response Example

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 15 Jan 2019 21:12:17 GMT

{ "locations": [
    "CDN_in_a_Box_Edge"
]}

/crs/locations/caches

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

Request Structure

# 85: Request Example

GET /crs/locations/caches HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*

Response Structure

# 86: Response Example

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 15 Jan 2019 21:15:53 GMT

{ "locations": {
    "CDN_in_a_Box_Edge": [
        {
            "cacheId": "edge",
            "fqdn": "edge.infra.ciab.test",
            "ipAddresses": [
                "172.16.239.100",
                "172.16.239.101",
                "172.16.239.102"
            ]
        }
    ]
}
"fc01:9400:1000:8:0:0:0:100",
    "port": 0,
    "adminStatus": null,
    "lastUpdateHealthy": false,
    "lastUpdateTime": 0,
    "connections": 0,
    "currentBW": 0,
    "availBW": 0,
    "cacheOnline": true
  ]
],
"port": 0,
"adminStatus": null,
"lastUpdateHealthy": false,
"lastUpdateTime": 0,
"connections": 0,
"currentBW": 0,
"availBW": 0,
"cacheOnline": true
]
]
}

/crs/locations/{{cachegroup}}/caches

A list of cache servers for this Cache Group only.

Request Structure

Table 49: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachegroup</td>
<td>The name of a Cache Group of which a list of constituent cache servers will be retrieved</td>
</tr>
</tbody>
</table>

# 87: Request Example

GET /crs/locations/CDN_in_a_Box_Edge/caches HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*

Response Structure

# 88: Response Example

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 15 Jan 2019 21:18:25 GMT

{ "caches": [
  
]
"cacheId": "edge",
"fqdn": "edge.infra.ciab.test",
"ipAddresses": [
  "172.16.239.100",
  "fc01:9400:1000:8:0:0:0:100"
],
"port": 0,
"adminStatus": null,
"lastUpdateHealthy": false,
"lastUpdateTime": 0,
"connections": 0,
"currentBW": 0,
"availBW": 0,
"cacheOnline": true
}

/crs/consistenthash/cache/coveragezone

The resulting cache of the consistent hash using coverage zone file for a given client IP, Delivery Service, and request path.

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>yes</td>
<td>The IP address of a potential client</td>
</tr>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?/xml_id?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
<tr>
<td>requestPath</td>
<td>yes</td>
<td>The... request path?</td>
</tr>
</tbody>
</table>

Response Structure

TBD

/crs/consistenthash/cache/deep/coveragezone

The resulting cache of the consistent hash using deep coverage zone file (deep caching) for a given client IP, Delivery Service, and request path.
Request Structure

Table 51: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>yes</td>
<td>The IP address of a potential client</td>
</tr>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?/xml_id?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
<tr>
<td>requestPath</td>
<td>yes</td>
<td>The... request path?</td>
</tr>
</tbody>
</table>

Response Structure

TBD

/crs/consistenthash/cache/geolocation

The resulting cache of the consistent hash using geographic location for a given client IP, Delivery Service, and request path.

Request Structure

Table 52: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>yes</td>
<td>The IP address of a potential client</td>
</tr>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?/xml_id?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
<tr>
<td>requestPath</td>
<td>yes</td>
<td>The... request path?</td>
</tr>
</tbody>
</table>

Response Structure

TBD

/crs/consistenthash/deliveryservice/

The resulting Delivery Service of the consistent hash for a given Delivery Service and request path – used to test STEERING Delivery Services.
Request Structure

Table 53: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?&quot;xml_id&quot;?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
<tr>
<td>requestPath</td>
<td>yes</td>
<td>The... request path?</td>
</tr>
</tbody>
</table>

# 89: Request Example

```
GET /crs/consistenthash/deliveryservice?deliveryServiceId=demo1&
    requestPath=/ HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
```

Response Structure

# 90: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 15 Jan 2019 21:40:51 GMT

{
  "id": "demo1",
  "coverageZoneOnly": false,
  "geoRedirectUrl": null,
  "geoRedirectFile": null,
  "geoRedirectUrlType": "INVALID_URL",
  "routingName": "video",
  "missLocation": {
    "latitude": 42,
    "longitude": -88,
    "postalCode": null,
    "city": null,
    "countryCode": null,
    "countryName": null,
    "defaultLocation": false,
    "properties": {
      "city": null,
      "countryCode": null,
      "latitude": "42.0",
      "postalCode": null,
      "countryName": null,
```
(continues on next page)
/crs/coveragezone/caches

A list of caches for a given Delivery Service and Cache Group.

Request Structure

Table 54: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?xml_id?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
<tr>
<td>cacheLocationId</td>
<td>yes</td>
<td>The name of a Cache Group to which this Traffic Router is capable of routing client traffic</td>
</tr>
</tbody>
</table>

Response Structure

TBD
The resulting Cache Group for a given client IP and Delivery Service.

**Request Structure**

Table 55: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>yes</td>
<td>The IP address of a potential client</td>
</tr>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?&quot;xml_id&quot;?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
</tbody>
</table>

**Response Structure**

TBD

The resulting Cache Group using deep coverage zone file (deep caching) for a given client IP and Delivery Service.

**Request Structure**

Table 56: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>yes</td>
<td>The IP address of a potential client</td>
</tr>
<tr>
<td>delivery-ServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?&quot;xml_id&quot;?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
</tbody>
</table>

**Response Structure**

TBD

The resulting path that will be used for consistent hashing when the given regex is applied to the given request path.
Request Structure

Table 57: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>regex</td>
<td>yes</td>
<td>The (URI encoded) regular expression to be used to test pattern based consistent hashing</td>
</tr>
<tr>
<td>request-Path</td>
<td>yes</td>
<td>The (URI encoded) request path to use to test pattern based consistent hashing</td>
</tr>
</tbody>
</table>

# 91: Request Example

GET /crs/consistenthash/patternbased/regex?regex=%2F.*%3F%28%2F.*%3F%2F%29.*%3F%28%5C.m3u8%29&requestPath=%2Ftext1234%2Fname%2Fasset.m3u8 HTTP/1.1
Host: localhost:3333
User-Agent: curl/7.54.0
Accept: */*

Response Structure

# 92: Response Example

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 15 Feb 2019 22:06:53 GMT

{
  "resultingPathToConsistentHash": "/name/.m3u8",
  "consistentHashRegex": "/.*?(/.*?/).*?(\.m3u8)",
  "requestPath": "/text1234/name/asset.m3u8"
}

/crs/consistenthash/patternbased/deliveryservice

The resulting path that will be used for consistent hashing for the given delivery service and the given request path.
Request Structure

Table 58: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestPath</td>
<td>yes</td>
<td>The (URI encoded) request path to use to test pattern based consistent hashing</td>
</tr>
<tr>
<td>deliveryServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?xml_id?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
</tbody>
</table>

# 93: Request Example

```
GET /crs/consistenthash/patternbased/deliveryservice?deliveryServiceId=asdf&requestPath=%2Fsometext1234%2Fstream_name HTTP/1.1
Host: localhost:3333
User-Agent: curl/7.54.0
Accept: */*
```

Response Structure

# 94: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 15 Feb 2019 22:12:38 GMT

{
"resultingPathToConsistentHash": "/sometext1234/stream_name/asset_name.m3u8",
"deliveryServiceId": "asdf",
"requestPath": "/sometext1234/stream_name/asset_name.m3u8"
}
```

/crs/consistenthash/cache/coveragezone/steering

The resulting cache of the consistent hash using coverage zone for a given client IP, delivery service and, request path – used to test cache selection for steering delivery services.
Request Structure

Table 59: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestPath</td>
<td>yes</td>
<td>The (URI encoded) request path to use to test pattern based consistent hashing</td>
</tr>
<tr>
<td>deliveryServiceId</td>
<td>yes</td>
<td>The integral, unique identifier?/xml_id?/name? of a Delivery Service served by this Traffic Router</td>
</tr>
<tr>
<td>ip</td>
<td>yes</td>
<td>The IP address of a potential client</td>
</tr>
</tbody>
</table>

Response Structure

TBD

4.1.7 Traffic Monitor

Introduction

Traffic Monitor is an HTTP service application that monitors cache servers, provides health state information to Traffic Router, and collects statistics for use in tools such as Traffic Portal and Traffic Stats. The health state provided by Traffic Monitor is used by Traffic Router to control which cache servers are available on the CDN.

Software Requirements

To work on Traffic Monitor you need a Unix-like (MacOS and Linux are most commonly used) environment that has a working install of Go version 1.7+.

Project Tree Overview

traffic_monitor/ - base directory for Traffic Monitor.

- cache/ - Handler for processing cache results.
- config/ - Application configuration; in-memory objects from traffic_monitor.cfg.
- crconfig/ - data structure for deserializing the CDN Snapshot (historically named “CRConfig”) from JSON.
- deliveryservice/ - aggregates Delivery Service data from cache results.
- deliveryservicedata/ - Delivery Service data structures. This exists separate from deliveryservice to avoid circular dependencies.
• **enum/** - enumerations and name alias types.

• **health/** - functions for calculating *cache server* health, and creating health event objects.

• **manager/** - manager goroutines (microthreads).
  - **health.go** - Health request manager. Processes health results, from the health poller -> fetcher -> manager. The health poll is the “heartbeat” containing a small amount of stats, primarily to determine whether a *cache server* is reachable as quickly as possible. Data is aggregated and inserted into shared thread-safe objects.
  - **manager.go** - Contains `Start` function to start all pollers, handlers, and managers.
  - **monitorconfig.go** - Monitor configuration manager. Gets data from the monitor configuration poller, which polls Traffic Ops for changes to which caches are monitored and how.
  - **opsconfig.go** - Ops configuration manager. Gets data from the ops configuration poller, which polls Traffic Ops for changes to monitoring settings.
  - **peer.go** - Peer manager. Gets data from the peer poller -> fetcher -> handler and aggregates it into the shared thread-safe objects.
  - **stat.go** - Stat request manager. Processes stat results, from the stat poller -> fetcher -> manager. The stat poll is the large statistics poll, containing all stats (such as HTTP response status codes, transactions, *Delivery Services* statistics, and more). Data is aggregated and inserted into shared thread-safe objects.
  - **statecombiner.go** - Manager for combining local and peer states, into a single combined states thread-safe object, for serving the `CrStates` endpoint.

• **datareq/** - HTTP routing, which has thread-safe health and stat objects populated by stat and health managers.

• **peer/** - Manager for getting and populating peer data from other Traffic Monitors

• **srvhttp/** - HTTP(S) service. Given a map of endpoint functions, which are lambda closures containing aggregated data objects. If HTTPS, the HTTP service will redirect to HTTPS.

• **static/** - Web interface files (markup, styling and scripting)

• **threadsafe/** - Thread-safe objects for storing aggregated data needed by multiple goroutines (typically the aggregator and HTTP server)

• **trafficopsdata/** - Data structure for fetching and storing Traffic Ops data needed from the CDN *Snapshot*. This is primarily mappings, such as *Delivery Services* servers, and server types.

• **trafficopswrapper/** - Thread-safe wrapper around the Traffic Ops client. The client used to not be thread-safe, however, it mostly (possibly entirely) is now. But, the wrapper also serves to overwrite the Traffic Ops `monitoring.json` values, which are live, with values from the CDN *Snapshot*.
Architecture

At the highest level, Traffic Monitor polls cache servers, aggregates their data and availability, and serves it at HTTP endpoints in JSON format. In the code, the data flows through microthread (goroutine) pipelines. All stages of the pipeline are independently running microthreads\(^1\). The pipelines are:

**stat poll** Polls caches for all statistics data. This should be a slower poll, which gets a lot of data.

**health poll** Polls caches for a tiny amount of data, typically system information. This poll is designed to be a heartbeat, determining quickly whether the cache server is reachable. Since it’s a small amount of data, it should poll more frequently.

**peer poll** Polls Traffic Monitor peers for their availability data, and aggregates it with its own availability results and that of all other peers.

**monitor config** Polls Traffic Ops for the list of Traffic Monitors and their info.

**ops config** Polls for changes to the Traffic Ops configuration file `traffic_ops.cfg`, and sends updates to other pollers when the configuration file has changed.

- The ops config manager also updates the shared Traffic Ops client, since it’s the actor which becomes notified of configuration changes requiring a new client.
- The ops config manager also manages, creates, and recreates the HTTP server, since Traffic Ops configuration changes necessitate restarting the HTTP server.

All microthreads in the pipeline are started by `manager/manager.go:Start()`.

![Fig. 30: Pipeline Overview](image.png)

Stat Pipeline

**poller** `common/poller/poller.go:HttpPoller.Poll()`. Listens for configuration changes (from the Ops Configuration Manager), and starts its own, internal mi-

\(^1\) Technically, some stages which are one-to-one simply call the next stage as a function. For example, the Fetcher calls the Handler as a function in the same microthread. But this isn’t architecturally significant.
crothreads - one for each cache to poll. These internal microthreads call the Fetcher at each cache’s poll interval.

**Fetcher** [common/fetcher/fetcher.go:HttpFetcher.Fetch()]{fetcher}. Fetches the given URL, and passes the returned data to the Handler, along with any errors.

**Handler** [traffic_monitor/cache/cache.go:Handler.Handle()]{handler}. Takes the given result and does all data computation possible with the single result. Currently, this computation primarily involves processing the de-normalized ATS data into Go structs, and processing System data into ‘OutBytes’, KBPS, etc. Precomputed data is then passed to its result channel, which is picked up by the Manager.

**Manager** [traffic_monitor/manager/stat.go:StartStatHistoryManager()]{manager}. Takes preprocessed results, and aggregates them. Aggregated results are then placed in shared data structures. The major data aggregated are Delivery Service statistics, and cache server availability data. See Aggregated Stat Data and Aggregated Availability Data.

### Health Pipeline

**Poller** [common/poller/poller.go:httpPoller.Poll()]{poller}. Same poller type as the Stat Poller pipeline, with a different handler object.

**Fetcher** [common/fetcher/fetcher.go:HttpFetcher.Fetch()]{fetcher}. Same fetcher type as the Stat Poller pipeline, with a different handler object.

**Handler** [traffic_monitor/cache/cache.go:Handler.Handle()]{handler}. Same handler type as the Stat Poller pipeline, but constructed with a flag to not pre-compute anything. The health endpoint is of the same form as the stat endpoint, but doesn’t return all stat data. So, it doesn’t pre-compute like the Stat Handler, but only processes the system data, and passes the processed result to its result channel, which is picked up by the Manager.

**Manager** [traffic_monitor/manager/health.go:StartHealthResultManager()]{manager}. Takes preprocessed results, and aggregates them. For the Health pipeline, only health availability data is aggregated. Aggregated results are then placed in shared data
structures (lastHealthDurationsThreadsaf, lastHealthEndTimes, etc). See Aggregated Availability Data.

Peer Pipeline

poller common/poller/poller.go:HttpPoller.Poll(). Same poller type as the Stat and Health Poller pipelines, with a different handler object. Its configuration changes come from the Monitor Configuration Manager, and it starts an internal microthread for each peer to poll.

fetcher common/fetcher/fetcher.go:HttpFetcher.Fetch(). Same fetcher type as the Stat and Health Poller pipeline, with a different handler object.

handler traffic_monitor/cache/peer.go:Handler.Handle(). Decodes the JSON result into an object, and without further processing passes to its result channel, which is picked up by the Manager.

manager traffic_monitor/manager/peer.go:StartPeerManager(). Takes JSON peer Traffic Monitor results, and aggregates them. The availability of the Peer Traffic Monitor itself, as well as all cache server availability from the given peer result, is stored in the shared peerStates object. Results are then aggregated via a call to the combineState() lambda, which signals the State Combiner microthread (which stores the combined availability in the shared object combinedStates; See State Combiner).

Monitor Config Pipeline

poller common/poller/poller.go:MonitorConfigPoller.Poll(). The Monitor Configuration poller, on its interval, polls Traffic Ops for the Monitor configuration, and writes the polled value to its result channel, which is read by the Manager.

manager traffic_monitor/manager/monitorconfig.go:StartMonitorConfigManager(). Listens for results from the poller, and processes them. Cache changes are written to channels read by the Health, Stat, and
Peer pollers. In the Shared Data objects, this also sets the list of new Delivery Services and removes ones which no longer exist, and sets the list of peer Traffic Monitors.

**Ops Config Pipeline**

![Ops Configuration Pipeline Diagram](image)

**poller** common.poller/poller.go:FilePoller.Poll(). Polls for changes to the Traffic Ops configuration file traffic_ops.cfg, and writes the changed configuration to its result channel, which is read by the Handler.

**handler** common.handler/handler.go:OpsConfigFileHandler.Listen(). Takes the given raw configuration, un-marshals the JSON into an object, and writes the object to its channel, which is read by the Manager, along with any error.

**manager** traffic_monitor/manager/monitorconfig.go:StartMonitorConfigManager(). Listens for new configurations, and processes them. When a new configuration is received, a new HTTP dispatch map is created via traffic_monitor/datareq/datareq.go:MakeDispatchMap(), and the HTTP server is restarted with the new dispatch map. The Traffic Ops client is also recreated, and stored in its shared data object. The Ops Configuration change subscribers and Traffic Ops Client change subscribers (the Monitor Configuration poller) are also passed the new Traffic Ops configuration and new Traffic Ops client.

**Events**

The events shared data object is passed to each pipeline microthread which needs to signal events. Most of them do. Events are then logged, and visible in the UI as well as an HTTP JSON endpoint. Most events are cache server becoming available or unavailable, but include other things such as peer availability changes.

**State Combiner**

The State Combiner is a microthread started in traffic_monitor/manager/manager.go:Start() via traffic_monitor/manager/statecombiner.go:StartStateCombiner(), which listens for signals to combine states. It should be signaled by any pipeline which updates the local or peer availability shared data objects, localStates and peerStates. It holds the thread-safe shared data objects for local states and peer states, so no data is passed or returned, only a signal. When a signal is received, it combines the local and peer states optimistically. That is, if a cache server is marked available locally or by any peer, that cache server is marked available in the combined states. There exists a variable to combine pessimistically, which may be set at compile time (it's...
unusual for a CDN to operate well with pessimistic cache server availability). Combined data is stored in the thread-safe shared data object combinedStates.

**Aggregated Stat Data**

The Stat pipeline Manager is responsible for aggregating stats from all cache servers, into Delivery Services statistics. This is done via a call to traffic_monitor/deliveryservice/stat.go:CreateStats().

**Aggregated Availability Data**

Both the Stat and Health pipelines aggregate availability data received from caches. This is done via a call to traffic_monitor/deliveryservice/health.go:CalcAvailability() followed by a call to combineState(). The CalcAvailability function calculates the availability of each cache server from the result of polling it, that is, local availability. The combineState() function is a lambda passed to the Manager, which signals the State Combiner microthread, which will combine the local and peer Traffic Monitor availability data, and insert it into the shared data combinedStates object.

**HTTP Data Requests**

Data is provided to HTTP requests via the thread-safe shared data objects (see Shared Data). These objects are closed in lambdas created via traffic_monitor/datareq/datareq.go:MakeDispatchMap(). This is called by the Ops Configuration Manager when it recreates the HTTP(S) server. Each HTTP(S) endpoint is mapped to a function which closes around the shared data objects it needs, and takes the request data it needs (such as query parameters). Each endpoint function resides in its own file in traffic_monitor/datareq/. Because each Go HTTP routing function must be a http.HandlerFunc, wrapper functions take the endpoint functions and return http.HandlerFunc functions which call them, and which are stored in the dispatch map, to be registered with the HTTP(S) server.

**Shared Data**

Processed and aggregated data must be shared between the end of the stat and health processing pipelines, and HTTP requests. The CSP (Communicating Sequential Processes) paradigm of idiomatic Go does not work efficiently with storing and sharing state. While not idiomatic Go, shared mutexed data structures are faster and simpler than CSP manager microthreads for each data object. Traffic Monitor has many thread-safe shared data types and objects. All shared data objects can be seen in manager/manager.go:Start(), where they are created and passed to the various pipeline stage microthreads that need them. Their respective types all include the word Threadsafe, and can be found in traffic_monitor/threadsafe/ as well as, for dependency reasons, various appropriate directories. Currently, all thread-safe
shared data types use mutexes. In the future, these could be changed to lock-free or wait-free structures, if the performance needs outweighed the readability and correctness costs. They could also easily be changed to internally be manager microthreads and channels, if being idiomatic were deemed more important than readability or performance.

Disk Backup

The Traffic Monitor configuration and CDN Snapshot are both stored as backup files (tmconfig.backup and crconfig.backup or whatever you set the values to in the configuration file). This allows the monitor to come up and continue serving even if Traffic Ops is down. These files are updated any time a valid configuration is received from Traffic Ops, so if Traffic Ops goes down and Traffic Monitor is restarted it can still serve the previous data. These files can also be manually edited and the changes will be reloaded into Traffic Monitor so that if Traffic Ops is down or unreachable for an extended period of time manual updates can be done. If on initial startup Traffic Ops is unavailable then Traffic Monitor will continue through its exponential back-off until it hits the max retry interval, at that point it will create an unauthenticated Traffic Ops session and use the data from disk. It will still poll Traffic Ops for updates though and if it successfully gets through then it will login at that point.

Formatting Conventions

Go code should be formatted with gofmt. See also CONTRIBUTING.md.

Installing The Developer Environment

To install the Traffic Monitor Developer environment:

1. Install Go version 1.7 or greater
2. Clone the Traffic Control repository using git, into $GOPATH/src/github.com/apache/trafficcontrol
3. Change directories into $GOPATH/src/github.com/apache/trafficcontrol/traffic_monitor
4. Run ./build.sh

Test Cases

Tests can be executed by running go 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.

**Note:** Unlike Traffic Ops API endpoints, no authentication is required for any of these, and as such there can be no special role requirements for a user.

/publish/EventLog

Gets a log of recent changes in the availability of polled caches.

**GET**

**Response Type** Array (key ‘events’ contains an array of all data)

**Response Structure**

- **event** an entry in the top-level events array
  - **description** A string containing short description of the event
  - **hostname** A string containing the server’s full hostname
  - **index** A serial integer that is incremented for each sequential event
  - **isAvailable** A boolean value indicating whether the server is available following this event
  - **name** The server’s short hostname as a string
  - **time** A UNIX timestamp as an integer
  - **type** The type of the server as a string

# 95: Example Response

```json
{
  "events": [
    {
      "time": 1538417713,
      "index": 67848,
      "description": "REPORTED - loadavg too high (36.37 \u003e 25.00) (health)",
      "name": "edge",
      "hostname": "edge",
      "type": "EDGE",
    }
  ]
}
```

(continues on next page)
"isAvailable":false
]
]

/publish/CacheStats

Statistics gathered for each cache.

GET

Response Type Object

Request Structure

Table 60: Request Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc</td>
<td>integer</td>
<td>The history count, number of items to display.</td>
</tr>
<tr>
<td>stats</td>
<td>string</td>
<td>A comma separated list of stats to display.</td>
</tr>
<tr>
<td>wildcard</td>
<td>boolean</td>
<td>Controls whether specified stats should be treated as partial strings.</td>
</tr>
</tbody>
</table>

Response Structure

pp Stores any provided request parameters provided as a string

date A ctime-like string representation of the time at which the response was served

caches An object with keys that are the names of monitored cache servers

<server name> Each server’s object is a collection of keys that are the names of statistics

<statistic name> The name of the statistic which this array represents. Each value in the array is one (and usually only one) object with the following structure:

value The statistic’s value. This is always a string, even if that string only contains a number.

time An integer UNIX timestamp indicating the start time for this value of this statistic
span The span of time - in milliseconds - for which this value is valid. This is determined by the polling interval for the statistic

# 96: Example Response

```{}
```

publish/CacheStats/{{cache}}

Statistics gathered for only a single cache.

GET

Response Type Object

Request Structure

Table 61: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>string</td>
<td>The name of the cache to inspect</td>
</tr>
</tbody>
</table>

Table 62: Request Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc</td>
<td>integer</td>
<td>The history count, number of items to display.</td>
</tr>
<tr>
<td>stats</td>
<td>string</td>
<td>A comma separated list of stats to display.</td>
</tr>
<tr>
<td>wildcard</td>
<td>boolean</td>
<td>Controls whether specified stats should be treated as partial strings.</td>
</tr>
</tbody>
</table>

Response Structure

pp Stores any provided request parameters provided as a string
date A ctime-like string representation of the time at which the response was served
caches An object with keys that are the names of monitored cache servers - only the cache named by the cache request path parameter will be shown

<server name> The requested server’s object is a collection of keys that are the names of statistics

<statistic name> The name of the statistic which this array represents. Each value in
the array is one (and usually only one) object with the following structure:

- **value** The statistic’s value. This is *always* a string, even if that string only contains a number.
- **time** An integer UNIX timestamp indicating the start time for this value of this statistic
- **span** The span of time - in milliseconds - for which this value is valid. This is determined by the polling interval for the statistic

# 97: Example Response

```json
{}
```

`/publish/DsStats`

Statistics gathered for *Delivery Services*

GET

**Response Type** Object

**Request Structure**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc</td>
<td>int</td>
<td>The history count, number of items to display.</td>
</tr>
<tr>
<td>stats</td>
<td>string</td>
<td>A comma separated list of stats to display.</td>
</tr>
<tr>
<td>wildcard</td>
<td>boolean</td>
<td>Controls whether specified stats should be treated as partial strings.</td>
</tr>
</tbody>
</table>

**Response Structure**

TODO

`/publish/DsStats/{{deliveryService}}`

Statistics gathered for this *Delivery Service* only.
GET

Response Type  ?

Request Structure

Table 64: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deliveryService</td>
<td>string</td>
<td>The name of the Delivery Service to inspect</td>
</tr>
</tbody>
</table>

Table 65: Request Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc</td>
<td>integer</td>
<td>The history count, number of items to display.</td>
</tr>
<tr>
<td>stats</td>
<td>string</td>
<td>A comma separated list of stats to display.</td>
</tr>
<tr>
<td>wildcard</td>
<td>boolean</td>
<td>Controls whether specified stats should be treated as partial strings.</td>
</tr>
</tbody>
</table>

Response Structure

TODO

/publish/CrStates

The current state of this CDN per the ref:health-proto.

GET

Response Type  ?

Response Structure

TODO

..???. raw

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

/publish/CrConfig

The CDN Snapshot (historically named a “CRConfig”) served to and consumed by Traffic Router.
**GET**

Response Type ?

Response Structure

TODO

/publish/PeerStates

The health state information from all peer Traffic Monitors.

**GET**

Response Type ?

Response Structure

Table 66: Request Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hc</td>
<td>integer</td>
<td>The history count, number of items to display.</td>
</tr>
<tr>
<td>stats</td>
<td>string</td>
<td>A comma separated list of stats to display.</td>
</tr>
<tr>
<td>wildcard</td>
<td>boolean</td>
<td>Controls whether specified stats should be treated as partial strings.</td>
</tr>
</tbody>
</table>

Response Structure

TODO

/publish/Stats

The general statistics about Traffic Monitor.

**GET**

Response Type ?

Response Structure

TODO
The Apache Traffic Control Documentation, Release 3.0.0

/publish/StatSummary

The summary of cache server statistics.

GET

Response Type ?

Request Structure

Table 67: Request Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>number</td>
<td>Window start. The number of milliseconds since the epoch.</td>
</tr>
<tr>
<td>endTime</td>
<td>number</td>
<td>Window end. The number of milliseconds since the epoch.</td>
</tr>
<tr>
<td>hc</td>
<td>integer</td>
<td>The history count, number of items to display.</td>
</tr>
<tr>
<td>stats</td>
<td>string</td>
<td>A comma separated list of stats to display.</td>
</tr>
<tr>
<td>wildcard</td>
<td>boolean</td>
<td>Controls whether specified stats should be treated as partial strings.</td>
</tr>
<tr>
<td>cache</td>
<td>string</td>
<td>Summary statistics for just this cache.</td>
</tr>
</tbody>
</table>

Response Structure

TODO

/publish/ConfigDoc

The overview of configuration options.

GET

Response Type ?

Response Structure

TODO

4.1.8 Traffic Stats
Introduction

Traffic Stats is a utility written in Go that is used to acquire and store statistics about CDNs controlled by Traffic Control. Traffic Stats mines metrics from the Traffic Monitor APIs and stores the data in InfluxDB. Data is typically stored in InfluxDB on a short-term basis (30 days or less) and is used to drive graphs created by Grafana which are linked from Traffic Ops. Traffic Stats also calculates daily statistics from InfluxDb and stores them in the Traffic Ops database.

Software Requirements

- A *nix (MacOS and Linux are most commonly used) environment
- Go 1.7.x or above
- Access to a working instance of Traffic Ops
- Access to a working instance of Traffic Monitor
- InfluxDB version 1.0.0 or greater

Traffic Stats Project Tree Overview

- traffic_stats/ - contains Go source files and files used to create the Traffic Stats RPM.
  - grafana/ - contains a javascript file which is installed on the Grafana server. This allows Traffic Ops to create custom dashboards for Delivery Services, cache server's, etc.
  - influxdb_tools/ - contains sync/sync_ts_databases.go and create/create_ts_databases.go which are helpful if you have multiple instances and they get out of sync with data.

Go Formatting Conventions

In general Go fmt is the standard for formatting Go code. It is also recommended to use Go lint.

Installing The Developer Environment

1. Clone the traffic_control repository using Git into a location accessible by your $GOPATH
2. Navigate to the traffic_ops/client directory of your cloned repository. (This is the directory containing Traffic Ops client code used by Traffic Stats)
3. From the traffic_ops/client directory run go test to test the client code. This will run all unit tests for the client and return the results. If there are missing dependencies you will need to run go get <dependency name> to get the dependency.

4. Once the tests pass, run go install to build and install the Traffic Ops client package. This makes it accessible to Traffic Stats.

5. Navigate to your cloned repository under Traffic Stats.

6. Run go build traffic_stats.go to build traffic_stats. You will need to run go get for any missing dependencies.

4.1.9 Documentation Guidelines

The Apache Traffic Control documentation is written in RST (reStructuredText) and uses the Sphinx documentation build system to convert these into the desired output document format. This collection of guidelines does not aim to be a primer in RST, but merely a style guide regarding how the components of the document ought to be formatted and structured. It may also point out some features of the markup language of which a writer may not be aware.

See also:

The docutils RST reference.

Building

To build the documentation, see Building This Documentation.

Writing

When writing documentation, the most important things to remember are:

- Spell Check. Most text editors have this built in (e.g. F6 in Sublime Text) or have plugins that will do this for you.

- Proof-Read. Spell-checkers won’t catch grammatical errors or poor wording, so it’s very important to actually proof-read all documentation before submitting it in a Pull Request.

- Make Sure the Documentation Actually Builds. Please actually verify the documentation not only builds, but builds correctly. That means there probably shouldn’t be any warnings, no malformed tables etc. and it also means that new documentation is actually accessible once built. It’s not enough to create a new something.rst file, that file must actually be linked to from some other, already included document. Some warnings may be considered acceptable, but do be prepared to defend them.

- Traffic Ops UI is Dead. Do not ever create documentation that references or includes images of the Traffic Ops UI. That is officially dead now, and if the documentation being created is best made with references to a user-friendly UI, such references, examples and/or images should all be to/of Traffic Portal.
Formatting

Whenever possible, avoid specifying manual line breaks, except as required by RST syntax. Extremely long lines will be wrapped by the user-agent, compiler, or output format as necessary. A single blank line may be used to separate paragraphs. This means that the ‘flow break’ character should never need to be used, i.e. no line in the documentation should ever match the regular expression `^\|\$`.

Abbreviations

When using an abbreviation, acronym or initialism for the first time on a page, it **must** be named fully and followed by the abbreviation in parentheses e.g. “Fully Qualified Domain Name (FQDN)”. Strictly speaking, the *best* way to create an abbreviation is to always fully name it in parentheses immediately following the abbreviation using the `:abbr:` RST text role e.g. `:abbr:` FQDN (Fully Qualified Domain Name)`, but it's not reasonable to expect that of everyone. Some abbreviations can be assumed to be understood by the documentation’s target audience, and do not need full naming; they are general, basic networking and computing terms including (though not strictly limited to):

- API
- CSS
- DNS
- HTML
- HTTP
- HTTPS
- IP/IPv4/IPv6
- ISO
- JPG
- JSON
- PDF
- PNG
- RPM
- SQL
- SSL
- SVG
- TCP
- TLS
- UDP
Floating Objects

“Floating objects” are images, tables, source code listings, and equations. These may not be placed relative to other content exactly as shown in the source RST document, as it may be necessary to move them for e.g. page breaks in PDF documents so they are not split across pages.

Figures

Images should always be included inside of a .. figure directive. Always caption figures to make their purpose clear, as well as to make them directly link-able inside of the document and include them in figure listings. Though not syntactically required, figures should, in general be sized explicitly. The size should not be absolute, however; i.e. use 70% not 540px. Figures should, in general, be centered on the page. When drawings, graphs, or diagrams are included they should ideally be provided in both SVG and PNG formats, and included using globbing as filename.*. This will use the appropriate format for the output type.

Source Code Listings

Do not ever use the double-colon (:: ) directive to mark a section of text as a source code listing. This not only doesn’t support direct linking or provide a caption, but also uses the default “domain” - which is Python - for syntax highlighting. Instead, use .. code-block to include source code in the documentation. Source code must always be left-aligned, so do not provide any configuration options that would alter the default.
Tables

Tables should be included in .. table directive bodies, never as a floating, block-quoted tabular environment. This ensures that all tables will be captioned, which makes their purpose clear and makes them directly link-able in the output as well as includes them in table listings. Tables should avoid wrapping lines until they reach 215 characters in width in the source RE-STRUCTUREDTEXT document (including indentation which should be counted as 4 characters per TAB). No table may ever exceed 215 characters in width. Tables should, in general be left-aligned (which is the default configuration). For the usage or command-line flags or options of a utility, use an “option list” or the .. program and .. option directives instead of a table.

Indentation

Firstly, indentation of a text paragraph is rarely required. Doing so does not “link” the text with a heading in any way, text falls beneath a section or sub-section purely by being literally beneath that heading. When placing source code into a source code listing, any indentation may be used for the source code (typically should represent the actual preferred indentation of the code’s respective project were it included in the repository), but to avoid ambiguity in indentation used in the documentation versus indentation used in the source code, all documentation indentation should be done using one (1) TAB character and never spaces.

Lists

When making a list, consider first what kind of list it is. A list only needs to be enumerated if the enumeration has meaning e.g. a list of hierarchically ordered data or a sequential list of steps to accomplish a task or desired state. When enumerating list items, use #. instead of literal numbers, as this will enumerate automatically which makes modification of the list much easier at a later date. Unordered lists may use - or * for each item. Lists do not need to be indented beyond the current paragraph level. If a list is an unordered list of terms and their definitions, use a “definition list” instead of any other kind of list. If a list is a list of fields in a document or object, use a “field list” instead of any other kind of list. If a list is a list of parameters, arguments, or flags used by a command-line utility, use an “option list” instead of any other kind of list.

Notes and Footnotes

Instead of **NOTE** or similar, consider using the .. note directive, or one of the appropriate admonitions supported by RST:

attention The default admonition that calls attention to something without any specific semantics or attached context. Use when none of the others seem appropriate.

caution Includes cautionary information. Should be used when advising the reader that the containing section includes instructions or information that frequently confuse people/trip people up, and how to avoid these pitfalls.
danger Advises the reader of potential security risks or system damage that could occur as a result of following instructions in the containing section, or as a result of making assumptions about and/or improperly utilizing information in the containing section.

error Denotes an error. This has limited uses in the ATC documentation.

hint Offers a hint to nudge readers in the direction of a solution. This has limited uses in the ATC documentation.

impl-detail Contains information describing the details of an implementation of a feature described in the containing section. For an example, see the DSCP section of the Delivery Services page.

Note: This is an extension provided by the ATC documentation build system, and will not appear in the Sphinx project documentation nor the reStructuredText standard.

important Contains information that is important to consider while reading the containing section. Typically content that is related to a section’s content in an important way should appear as content of that section, but if a section is in danger of readers “skimming” it for information this can be useful to catch their eye.

note Used to segregate content that is only tangentially related to the containing section, but is noteworthy nonetheless. Historically the most used admonition, containing caveats, exceptions etc.

tip Provides the reader with information that can be helpful - particularly to users/administrators/developers new to ATC - but not strictly necessary to understand the containing section.

warning Warns the reader about possible unintended consequences of following instructions/utilizing information in the containing section. If the behavior warned about constitutes a security risk and/or serious damage to one or more systems - including clients and origins - please use .. danger instead.

In a similar vein, instead of e.g. “(See also: some-link-or-reference)” please use the special .. seealso admonition. If the same admonition is required more than twice on the same page, it most likely ought to be a footnote instead. Footnotes should ideally use human-readable labels or, failing that, be labeled sequentially in the order of appearance. Footnotes should appear at the end of the major section in which they first or last appear. In practice, however, placement of the footnote is left to the writer’s discretion.

Section Headings

When deciding on the name of a section heading, it is important to select a name that both accurately reflects the content it contains and is suitable for reference later e.g. “Health Protocol” is good, but “Configuring Multi-Site Origin” as the title of a page which not only explains MSO configuration but also the concept is not good. Section headings follow a hierarchy, and for consistency’s sake should follow this particular hierarchy:
1. Document title. There should only be one of these per page, and it should be the first heading on the page. This will also make the contained text the “Page Title” in HTML output (i.e. `<title>Page Title</title>` in the `<head>`).

# 99: Document Title

**************

Document Title

**************

2. Section header. This should represent a main topic of the page

# 100: Section Header

Section Header
==============

3. Subsection header. This should represent a key piece of a main topic on the page

# 101: Subsection Header

Subsection Header
-----------------

4. Sub-Subsection header. This represents a group of content logically separate from the rest of the subsection but still related to the content thereof. It is also acceptable to use this as an “aside” containing information only tangentially related to the subsection content.

# 102: Sub-Subsection Header

Sub-Subsection Header
"""""""""""""

5. Aside or Sub-Sub-Subsection header. This is the lowest denomination of header, and should almost always be used exclusively for “asides” which contain information only tangentially related to the sub-subsection content.

# 103: Aside

Aside
:

Section headings should always follow this order exactly, and never skip levels (which will generally cause a failure to compile properly). These can be thought of as the equivalents of the HTML tags `<h1>` through `<h5>`, respectively. Sectioning should never require more specificity than can be provided by an “Aside”. Please do not use bold text in lieu of a proper section heading. When referencing information in another section on the same page, please do not refer to the current placement of the referenced content relative to the referencing content. For example, instead of “as discussed below”, use “as discussed in Terms”.

4.1. Developer’s Guide
Terms

Please always spell out the entire name of any Traffic Control terms used in the definition. For example, a collection of cache servers associated with a certain physical location is called a “Cache Group”, not a “CG”, “cachegroup”, “cache location” etc. A subdomain and collection of cache servers responsible collectively for routing traffic to a specific origin is called a Delivery Service”, not a “DS”, “deliveryservice” etc. Similarly, always use full permissions role names e.g. “operations” not “oper”. This will ensure the Glossary is actually helpful. To link a term to the glossary, use the :term: role. This should be done for virtually every use of a Traffic Control term, e.g. :term:`Cache Group` will render as: Cache Group. Generally speaking, be wary of using the word “cache”. To most people that means the actual cache on a hard disk somewhere. This word is frequently confused with ”cache server”, which - when accurate - is always preferred over “cache”.

Documenting API Routes

Follow all of the formatting conventions in Formatting. Maintain the structural format of the API documentation as outlined in the Traffic Ops API section. API routes that have variable paths e.g. profiles/{{ID}} should use mustache templates not the Mojolicious-specific :param syntax. This keeps the templates generic, familiar, and reflects the inability of a request path to contain procedural instructions or program logic. Please do not include the /api/1.x/ part of the request path for Traffic Ops API endpoints. If an endpoint is unavailable prior to a specific version, use the .. versionadded directive to indicate that version. Likewise, do not make a new page for an endpoint when it changes across versions, instead call out the changes using the .. versionchanged directive. If an endpoint should not be used because newer endpoints provide the same functionality in a better way, use the .. deprecated directive to link to them and explain why they are better.

When documenting an API route, be sure to include all methods, request/response JSON payload fields, path parameters, and query parameters, whether they are optional or not. When describing a field in a JSON payload, remember that JSON does not have “hashes” it has “objects” or even “maps”. When documenting path parameters such as profile ID in profiles/{{ID}}, consider that the endpoint path cannot be formed without defining all path parameters, and so to label them as “required” is superfluous.

The “Response Example” must always exist. “TODO” is not an acceptable Response Example for new endpoints. The “Request Example” must only exist if the request requires data in the body (most commonly this will be for PATCH, POST and PUT methods). It is, however, strongly advised that a request example be given if the endpoint takes Query Parameters or Path Parameters, and it is required if the Response Example is a response to a request that used a query or path parameter. If the Request Example is present, then the Response Example must be the appropriate response to that request. When generating Request/Response Examples, attempt to use the CDN in a Box environment whenever possible to provide a common basis and familiarity to new users who likely set up “CDN in a Box” as a primer for understanding CDNs/Traffic Control. Responses are sometimes hundreds of lines long, and in those cases only as much as is required for an understanding of the structure needs to be included in the example - along with a note mentioning that the output was trimmed. Also always attempt to place structure explanations before any example so that the content of the example can be
understood by the reader (though in general the placement of a floating environment like a code listing is not known at compile-time). Whenever possible, the Request and Response examples should include the complete HTTP stack, which captures behavior like Query Parameters, Path Parameters and HTTP cookie operations like those used by e.g. logs. A few caveats to the “include all headers” rule:

- The Host header ought to reflect the actual hostname of the Traffic Ops server - which should be “trafficops.infra.ciab.test” for the CDN in a Box environment. This can be polluted when requests are made to a remotely running CDN in a Box on a different server.

- The “mojolicious” cookie is extremely long and potentially insecure to publicly show. As such, a placeholder should be used for its value, preferably “…”.

- The Content-Type header sent by curl(1) (and possibly others) is always application/x-www-form-urlencoded regardless of the actual content (unless overridden). Virtually all payloads accepted by the API must be JSON, so this should be modified to reflect that when appropriate e.g. application/json.

- API output is often beautified by inserting line breaks and indentation, which will make the Content-Length header (if any) incorrect. Don’t worry about fixing that - just try to leave the output as close as possible to what will actually be returned by leaving it the way it is.

File names should reflect the request path of the endpoint, e.g. a file for an endpoint of the Traffic Ops API /api/1.7/foo/{{fooID}}/bar/{{barID}} should be named foo_fooID_bar_barID.rst. Similarly, reference labels linking to the document title for API route pages should follow the convention: <component>-api-<path> in all lowercase where <component> is an abbreviated Traffic Control component name e.g. to and <path> is the request path e.g. foo_bar. So a label for an endpoint of the Traffic Ops API at /api/1.7/foo_bar/{{ID}} should be to-api-foo_bar-id.

Extension

The ATC documentation provides an extension to the standard roles and directives offered by Sphinx, located at docs/source/_ext/atc.py. It provides the following roles and directives:

impl-detail An admonition directive used to contain implementation-specific notes on a subject.

# 104: Example impl-detail usage

```impl-detail::
Implementation-specific information here.
```

This example usage renders like so:

**Implementation Detail**

Implementation-specific information here.
atc-file  Creates a link to the specified file on the master branch of the ATC repository. For example, `:atc-file:`docs/source/development/documentation_guidelines:` renders as a link to the source of this documenting section like so: docs/source/development/documentation_guidelines. You can also link to directories as well as files.

issue  A text role that can be used to easily link to GitHub Issues for the ATC repository. For example, `:issue:`1:` renders as: Issue #1.

pr  A text role that can be used to easily link to GitHub Pull Requests for the ATC repository. For example, `:pr:`1:` renders as Pull Request 1.

pull-request  A synonym for pr
A guide to external RESTful APIs for Traffic Ops

5.1 Traffic Ops API

The Traffic Ops API provides programmatic access to read and write Traffic Control data which allows for the monitoring of CDN performance and configuration of Traffic Control settings and parameters.

5.1.1 API Routes

api_capabilities

Deals with the capabilities that may be associated with API endpoints and methods. These capabilities are assigned to “roles”, of which a user may have one or more. Capabilities support “wildcarding” or “globbing” using asterisks to group multiple routes into a single capability

GET

Get all API-capability mappings.

Auth. Required Yes

Roles Required None

Response Type Array
Request Structure

Table 68: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capability</td>
<td>no</td>
<td>string</td>
<td>Capability name</td>
</tr>
</tbody>
</table>

# 105: Request Example

GET /api/1.1/api_capabilities?capability=types-write HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

**capability**  Capability name

**httpMethod**  An HTTP request method, practically one of:

- GET
- POST
- PUT
- PATCH
- DELETE

**httpRoute**  The request route for which this capability applies - relative to the Traffic Ops server’s URL

**id**  An integer which uniquely identifies this capability

**lastUpdated**  The time at which this capability was last updated, in ISO format

# 106: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 01 Nov 2018 14:45:24 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 01 Nov 2018 18:45:24 GMT; path=/; HttpOnly

(continues on next page)
POST

Create an API-capability mapping.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

capability  Capability name


httpRoute  The API endpoint for which to create capabilities
# 107: Request Example

```
POST /api/1.1/api_capabilities HTTP/1.1
Host: ipcdn-cache-51.cdnlab.comcast.net:6443
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 94
Content-Type: application/x-www-form-urlencoded

{
    "capability": "types-write",
    "httpRoute": "/api/1.1/api_capabilities",
    "httpMethod": "PATCH"
}
```  

## Response Structure

- **capability**  
  Capability name

- **httpMethod**  
  An HTTP request method, practically one of:
  - GET
  - POST
  - PUT
  - PATCH
  - DELETE

- **httpRoute**  
  The request route for which this capability applies - relative to the Traffic Ops server’s URL

- **id**  
  An integer which uniquely identifies this capability

- **lastUpdated**  
  The time at which this capability was last updated, in ISO format

# 108: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 01 Nov 2018 14:53:58 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 01 Nov 2018 18:53:58 GMT; path=/; HttpOnly
```
api_capabilities/{{ID}}

Manages a specific API capability.

GET

Get an API-capability mapping by id.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 69: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>yes</td>
<td>integer</td>
<td>A unique identifier for this capability</td>
</tr>
</tbody>
</table>

# 109: Request Example

GET /api/1.1/api_capabilities/273 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Response Structure

capability  Capability name

httpMethod  An HTTP request method, practically one of:
  • GET
  • POST
  • PUT
  • PATCH
  • DELETE

httpRoute  The request route for which this capability applies - relative to the Traffic Ops server’s URL

id  An integer which uniquely identifies this capability

lastUpdated  The time at which this capability was last updated, in an ISO-like format

# 110: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 01 Nov 2018 16:14:09 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 01 Nov 2018 20:14:09 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
Content-Length: 162

{ "response": [
  {
    "httpMethod": "PATCH",
    "lastUpdated": "2018-11-01 14:53:58.853356+00",
  }
]
```
PUT

Edit an API-capability mapping.

**Auth. Required** Yes

**Roles Required** "admin" or "operations"

**ResponseType** Object

**Request Structure**

- `capability` Capability name
- `httpMethod` An HTTP request method, practically one of:
  - GET
  - POST
  - PUT
  - PATCH
  - DELETE
- `httpRoute` The request route for which this capability applies - relative to the Traffic Ops server’s URL

**Table 70: Request Path Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>integer</td>
<td>A unique identifier for this capability</td>
</tr>
</tbody>
</table>

**# 111: Request Example**

```
PUT /api/1.1/api_capabilities/273 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 98
Content-Type: application/x-www-form-urlencoded
```
Response Structure

**capability**  Capability name

**httpMethod**  An HTTP request method, practically one of:

- GET
- POST
- PUT
- PATCH
- DELETE

**httpRoute**  The request route for which this capability applies - relative to the Traffic Ops server’s URL

**id**  An integer which uniquely identifies this capability

**lastUpdated**  The time at which this capability was last updated, in ISO format

# 112: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 01 Nov 2018 18:28:38 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 01 Nov 2018 22:28:38 GMT;
Vary: Accept-Encoding
Whole-Content-Sha512: zQuDrqpJt02Fh2fN26K7/
XmVJ49ZqGTnSbsaR7nOyoxbkmLM17XJV1rtef/SAows2M4j4YjcDbEP4WM/
jCtww==
Content-Length: 241

{  
"alerts": [
```
DELETE

Delete a capability.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  undefined

Request Structure

Table 71: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>integer</td>
<td>A unique identifier for this capability</td>
</tr>
</tbody>
</table>

# 113: Request Example

DELETE /api/1.1/api_capabilities/273 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure
# 114: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 07 Nov 2018 15:44:14 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 07 Nov 2018 19:44:14 GMT;
  path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: eTFJkB2Bh8SCT2A29e21e8goEdNzFGfuT5a3tDG7u8vwz/
  JHntQRRR8554ali65733uWoj1WM65bLSDNmmNqQ==
Content-Length: 73

{ "alerts": [  
  {  
    "level": "success",  
    "text": "API-capability mapping deleted."
  }
]}
```

**asns**

See also:


**GET**

List all Autonomous System Numbers (ASNs). :Auth. Required: Yes :Roles Required: None :Response Type: Array

**Request Structure**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachegroup</td>
<td>no</td>
<td>An integral, unique identifier for a Cache Group - only ANSs for this Cache Group will be returned.</td>
</tr>
</tbody>
</table>
Response Structure

- **lastUpdated**: The Time / Date this server entry was last updated in ISO format
- **id**: An integer which uniquely identifies the ASN
- **asn**: Autonomous System Numbers per APNIC for identifying a service provider
- **cachegroup**: Related Cache Group name
- **cachegroupId**: Related Cache Group ID

Changed in version 1.2: Used to contain the array in the `response.asns` object, changed so that `response` is an actual array.

# 115: Response Example

```json
{
  "response": [
    {
      "asn": 1,
      "cachegroup": "TRAFFIC_ANALYTICS",
      "cachegroupId": 1,
      "id": 1,
      "lastUpdated": "2018-11-01 18:55:39+00"
    }
  ]
}
```

**POST**

Creates a new Autonomous System Number (ASN).

- **Auth. Required**: Yes
- **Roles Required**: “admin” or “operations”
- **Response Type**: Object

5.1. Traffic Ops API
Request Structure

asn  The value of the new ASN

cachegroupId  The integral, unique identifier of a Cache Group to which this ASN will be assigned

cachegroup  An optional field which, if present, specifies the name of a Cache Group to which this ASN will be assigned

**Note:** While this endpoint accepts the cachegroup field, sending this in the request payload has no effect except that the response will (erroneously) name the Cache Group to which the ASN was assigned. Any subsequent requests will reveal that, in fact, the Cache Group name is set by the cachegroupId field.

# 116: Request Example

```
POST /api/1.1/asns HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 60
Content-Type: application/x-www-form-urlencoded

{"asn": 1, "cachegroupId": 1}
```

Response Structure

lastUpdated  The Time / Date this server entry was last updated in ISO format

id  An integer which uniquely identifies the ASN

asn  Autonomous System Numbers per APNIC for identifying a service provider

cachegroup  Related Cache Group name

cachegroupId  Related Cache Group ID

# 117: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
```
asns/{{id}}

See also:


GET

Deal with a specific Autonomous System Number (ASN). :Auth. Required: Yes :Roles Required: None :Response Type: Array

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the desired ASN</td>
</tr>
</tbody>
</table>

Response Structure

- **asn**  Autonomous System Numbers per APNIC for identifying a service provider
- **cachegroup**  Related Cache Group name
cachegroupId  Related Cache Group ID

id  An integer which uniquely identifies the ASN

lastUpdated  The time and date at which this server entry was last updated in an ISO-like format

# 118: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: oeifOX6ImUlKVyCmyswh4uddhbNP2qXliMuNw+1Nealq/SJOYKXpaKnYqVjRm9QqJ7gH3vqeBxCftMLtb3sAWg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 07 Nov 2018 18:44:31 GMT
Content-Length: 120

{
    "response": [
        {
            "lastUpdated": "2012-09-17 21:41:22",
            "id": "28",
            "asn": "7016",
            "cachegroup": "us-pa-pittsburgh",
            "cachegroupId": "13"
        }
    ]
}
```

PUT

Allows user to edit an existing Autonomous System Number (ASN).

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

Request Structure

**asn**  The value of the new ASN

**cachegroupId**  The integral, unique identifier of a Cache Group to which this ASN will be assigned
**cachegroup** An optional field which, if present, specifies the name of a Cache Group to which this ASN will be assigned.

**Note:** While this endpoint accepts the `cachegroup` field, sending this in the request payload has no effect except that the response will (erroneously) name the Cache Group to which the ASN was assigned. Any subsequent requests will reveal that, in fact, the Cache Group name is set by the `cachegroupId` field.

### Table 74: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the desired ASN</td>
</tr>
</tbody>
</table>

### # 119: Request Example

```bash
PUT /api/1.1/asns/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 29
Content-Type: application/x-www-form-urlencoded

{"asn": 2, "cachegroupId": 1}
```

### Response Structure

- **asn** Autonomous System Numbers per APNIC for identifying a service provider
- **cachegroup** Related Cache Group name
- **cachegroupId** Related Cache Group ID
- **id** An integer which uniquely identifies the ASN
- **lastUpdated** The date and time at which this server entry was last updated in an ISO-like format

### # 120: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-\Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
```

(continues on next page)
DELETE

Deletes an Autonomous System Number (ASN).

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type undefined

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the desired ASN</td>
</tr>
</tbody>
</table>

Response Structure

# 121: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
```
cache_stats

**Caution:** This page is a stub! Much of it may be missing or just downright wrong - it needs a lot of love from people with the domain knowledge required to update it.

Retrieves detailed, aggregated statistics for caches configured in Traffic Ops.

New in version 1.2.

**See also:**
This gives an aggregate of statistics for *all caches* within a particular CDN and time range. For statistics basic statistics from all caches regardless of CDN and at the current time, use `caches/stats`.

**GET**

Retrieves statistics about the caches within the CDN

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Object
Request Structure

Table 76: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdnName</td>
<td>yes</td>
<td>The name of a CDN. Results will represent caches within this CDN</td>
</tr>
<tr>
<td>metric-Type</td>
<td>yes</td>
<td>The metric type (valid metric types: ‘ats.proxy.process.http.current_client_connections’, ‘bandwidth’, ‘maxKbps’)</td>
</tr>
<tr>
<td>start-Date</td>
<td>yes</td>
<td>The begin date for data aggregation in ISO format, e.g. ‘2015-08-11T12:30:00-06:00’</td>
</tr>
<tr>
<td>end-Date</td>
<td>yes</td>
<td>The end date for data aggregation in ISO format, e.g. ‘2015-08-12T12:30:00-06:00’</td>
</tr>
</tbody>
</table>

Response Structure

- **series** A collection of tabular data and its descriptors
  - **columns** An array of names, in order, of the columns of the table. The first is a label for the first entry in each “value”, and so on.
  - **count** The total number of data points in the “values” array
  - **name** The name of the metric which was aggregated
  - **values** An array of the actual data points. Each of which is itself an array of properties, which are labeled by the “columns” array. This can be thought of as the data’s rows
    - **time** The time in ISO format at which this datum was collected
    - **value** The value of the datum. Its meaning is dependent upon “name” - and by extension the metricType request query parameter
  - **summary** A summary of the data contained in the “series” object
    - **average** The arithmetic mean across all data point values
    - **count** The total number of data points in the “series.values” array
    - **fifthPercentile** The right-hand threshold value for the 5th percentile
    - **max** The maximum of the requested metric values
    - **min** The minimum of the requested metric values
ninetyEighthPercentile  The right-hand threshold value for the 98th percentile

ninetyFifthPercentile  The right-hand threshold value for the 95th percentile

# 122: Response Example

```json
{
    "response": {
        "series": {
            "columns": [
                "time",
                ""
            ],
            "count": 29,
            "name": "bandwidth",
            "tags": {
                "cdn": "over-the-top"
            },
            "values": [
                ["2015-08-10T22:40:00Z", 229340299720],
                ["2015-08-10T22:41:00Z", 224309221713.334],
                ["2015-08-10T22:42:00Z", 229551834168.334],
                ["2015-08-10T22:43:00Z", 225179658876.667],
                ["2015-08-10T22:44:00Z", 230443968275]
            ],
            "summary": {
                "average": 970410.295,
                "count": 1376041798,
                "fifthPercentile": 202.03,
                "max": 3875441.02,
                "min": 0,
                "ninetyEighthPercentile": 2957940.93,
                "ninetyFifthPercentile": 2366728.63
            }
        }
    }
}
```

(continues on next page)
cachegroup_fallbacks

GET

Retrieve fallback-related configurations for a Cache Group.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheGroupId</td>
<td>yes</td>
<td>The integral, unique identifier of a Cache Group whose fallback configurations shall be retrieved</td>
</tr>
<tr>
<td>fallbackId</td>
<td>yes</td>
<td>The integral, unique identifier of a fallback Cache Group</td>
</tr>
</tbody>
</table>

# 123: Request Example

GET /api/1.3/cachegroup_fallbacks?cacheGroupId=11&fallbackId=7 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- cacheGroupId The integral, unique identifier of the Cache Group described by this entry
- cacheGroupName The name of the Cache Group described by this entry
- fallbackId The integral, unique identifier of the Cache Group on which the Cache Group described by this entry will fall back
- fallbackName The name of the Cache Group on which the Cache Group described by this entry will fall back

1 At least one of these must be provided, not necessarily both (though both is perfectly valid).
fallbackOrder  The order of the fallback described by “fallbackId” and “fallbackName” in the list of fallbacks for the Cache Group described by this entry

# 124: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 15:40:34 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 14 Nov 2018 19:40:34 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
9kau9r4Ea5ELZk0f4LpQr+Wcx6NHigWrn2JvyupRIOBQiUec3UW/9I9HdtE98xkrthz1daXKmdUkDhon8Q==
Content-Length: 125

{ "response": [
    {
        "cacheGroupId": 11,
        "fallbackOrder": 1,
        "fallbackName": "CDN_in_a_Box_Edge",
        "fallbackId": 7,
        "cacheGroupName": "test"
    }
]}
```

POST

Creates fallback configuration for a Cache Group.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Array

Request Structure

The request payload for this endpoint must be an array, even if only one fallback relationship is being created.

```
cacheGroupId  Integral, unique identifier of a Cache Group to which to assign a fallback
```
fallbackId Integral, unique identifier of a *Cache Group* on which the *Cache Group* identified by cacheGroupId will fall back

fallbackOrder The order of this fallback for the *Cache Group* identified by cacheGroupId

### # 125: Request Example

```
POST /api/1.3/cachegroup_fallbacks HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 59
Content-Type: application/x-www-form-urlencoded

[{"cacheGroupId": 11, "fallbackId": 7, "fallbackOrder": 1}]
```

### Response Structure

**cacheGroupId** The integral, unique identifier of the *Cache Group* to which this fallback was assigned

**cacheGroupName** The name of the *Cache Group* to which this fallback was assigned

**fallbackId** The integral, unique identifier of the *Cache Group* on which this entries *Cache Group* will fall back

**fallbackName** The name of the *Cache Group* on which this entries *Cache Group* will fall back

**fallbackOrder** The order of the fallback described by “fallbackId” and “fallbackName” in the list of fallbacks for the *Cache Group* described by this entry

### # 126: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-･･･...
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 08 Nov 2018 14:59:46 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 08 Nov 2018 18:59:46 GMT;
ｙpath=/;HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: ･･･;
```
Content-Length: 225

```
{
    "alerts": [
        {
            "level": "success",
            "text": "Backup configuration CREATE for cache group 11 successful."
        }
    ],
    "response": [
        {
            "cacheGroupId": 11,
            "fallbackName": "CDN_in_a_Box_Edge",
            "fallbackOrder": 1,
            "fallbackId": 7,
            "cacheGroupName": "test"
        }
    ]
}
```

**PUT**

Updates an existing fallback configuration for one or more Cache Groups.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Array

**Request Structure**

The request payload for this endpoint must be an array, even if only one fallback relationship is being updated. :cacheGroupId: Integral, unique identifier of a Cache Group to which to assign a fallback :fallbackId: Integral, unique identifier of a Cache Group on which the Cache Group identified by cacheGroupId will fall back :fallbackOrder: The order of this fallback for the Cache Group identified by cacheGroupId

**Note:** The request data should be an array of these objects (and any number can be submitted per request), see the example

**# 127: Request Example**

```
PUT /api/1.1/cachegroup_fallbacks HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```

(continues on next page)
Response Structure

**cacheGroupId**  The integral, unique identifier of the *Cache Group* to which this fallback was assigned

**cacheGroupName**  The name of the *Cache Group* to which this fallback was assigned

**fallbackId**  The integral, unique identifier of the *Cache Group* on which this entries *Cache Group* will fall back

**fallbackName**  The name of the *Cache Group* on which this entries *Cache Group* will fall back

**fallbackOrder**  The order of the fallback described by “fallbackId” and “fallbackName” in the list of fallbacks for the *Cache Group* described by this entry

# 128: Response Example

```json
{
   "alerts": [
      {
         "level": "success",
         "text": "Backup configuration UPDATE for cache group 11 successful."
      }
   ]
}
```

(continues on next page)
DELETE

Delete fallback list assigned to a Cache Group

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object (string)

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheGroupId</td>
<td>yes</td>
<td>The integral, unique identifier of a Cache Group whose fallback configurations shall be retrieved</td>
</tr>
<tr>
<td>fallbackId</td>
<td>yes²</td>
<td>The integral, unique identifier of a fallback Cache Group</td>
</tr>
</tbody>
</table>

# 129: Request Example

```
DELETE /api/1.2/cachegroup_fallbacks?cacheGroupId=11&fallbackId=7

Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

² At least one of “cacheGroupId” or “fallbackId” must be sent with the request. If both are sent, a single fallback relationship is deleted, whereas using only “cacheGroupId” will result in all fallbacks being removed from the Cache Group identified by that integral, unique identifier, and using only “fallbackId” will remove the Cache Group identified by that integral, unique identifier from all other Cache Groups’ fallback lists.
Response Structure

# 130: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 08 Nov 2018 15:48:56 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 08 Nov 2018 19:48:56 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: MG2FNZ18EAvy/IgdUPX4XRjXYYc1Xtp0e/kCMfi/mx9C427LNwvjLlseXkvu9crT2o68i0H2q1efshDJHO81IQ==
Content-Length: 76

{
    "response": "Backup Cachegroup 7 DELETED from cachegroup 11 fallback list"
}
```

cachegroup/\{{parameter ID}}}/parameter

Deprecated since version 1.1: Use cachegroupparameters instead

**Caution:** This endpoint does not appear to work, and thus its use is strongly discouraged!

**GET**

Extract identifying information about all cachegroups with a specific parameter

**Auth. Required** Yes

**Roles Required** None

**Response Type** Object
Request Structure

Table 79: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter_ID</td>
<td>yes</td>
<td>An integral, unique identifier for a parameter</td>
</tr>
</tbody>
</table>

Response Structure

- **cachegroups** An array of all Cache Groups with an associated parameter identifiable by the `parameter_id` request path parameter
- **id** The numeric ID of the Cache Group
- **name** The human-readable name of the Cache Group

# 131: Response Example

```json
{  
  "response": {  
    "cachegroups": [  
      {  
        "name": "CDN_in_a_Box_Edge",  
        "id": 7  
      },  
      {  
        "name": "CDN_in_a_Box_Mid",  
        "id": 6  
      },  
      {  
        "name": "TRAFFIC_ANALYTICS",  
        "id": 1  
      },  
      {  
        "name": "TRAFFIC_OPS",  
        "id": 2  
      },  
      {  
        "name": "TRAFFIC_OPS_DB",  
        "id": 3  
      },  
      {  
        "name": "TRAFFIC_PORTAL",  
        "id": 4  
      },  
      {  
        "name": "TRAFFIC_STATS",  
        "id": 5  
      }  
    ]  
  }  
}
```
cachegroupparameters

GET

Extract information about parameters associated with Cache Groups

Auth. Required  Yes
Roles Required  None
Response Type  Object

Response Structure

No available parameters

Response Structure

cachegroupParameters  An array of identifying information for parameters assigned to Cache Group profiles

  parameter  Numeric ID of the parameter

  last_updated  Date and time of last modification in ISO format

cachegroup  Name of the Cache Group

# 132: Response Example

```json
{
  "response": {
    "cachegroupParameters": [ {
      "parameter": 124,
    } ],
  }
}
```

(continues on next page)
POST

Assign parameter(s) to *Cache Group*(s).

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Array

**Request Structure**

The request data can take the form of either a single object or an array of one or more objects.

- `cacheGroupId`  Integral, unique identifier for the *Cache Group* to which a parameter is being assigned
- `parameterId`  Integral, unique identifier for the Parameter being assigned

# 133: Request Example

```json
POST /api/1.1/cachegroupparameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 39
Content-Type: application/json

{
    "cachegroupId": 8,
    "parameterId": 124
}
```

**Response Structure**

- `parameter`  Numeric ID of the parameter
- `last_updated`  Date and time of last modification in ISO format
- `cachegroup`  Name of the *Cache Group*
# 134: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 15:47:49 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 14 Nov 2018 19:47:49 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: wCv388wFaSjgFLCnI9dchlcyyGxaVr8IhBAG25F+rpI2/azCswrTcV8s1Y0g6NxTQrzGkluMvn67j16rV+vNsQ==
Content-Length: 136

{
  "alerts": [
    {
      "level": "success",
      "text": "Profile parameter associations were created."
    }
  ],
  "response": [
    {
      "cacheGroupId": 8,
      "parameterId": 124
    }
  ]
}
```

cachegroupparameters/{{ID}}/{{parameterID}}

DELETE

De-associate a parameter with a Cache Group

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined
Request Structure

Table 80: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| ID          | Unique identifier for the *Cache Group* which will have the parameter associa-
|             | tion deleted                                                               |
| parameterID | Unique identifier for the parameter which will be removed from a *Cache Group* |

# 135: Request Example

DELETE /api/1.1/cachegroupparameters/8/124 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

# 136: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 18:26:40 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 14 Nov 2018 22:26:40 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
   Cuj+ZPAKsDLp4FpbJDcwsWY0yVQAi1UmlCWrjaTQEMlyJSBEm17oKQWDjzTrvqV8Pthu3glcHoV
Content-Length: 84

```json
{
    "alerts": [
        {
            "level": "success",
            "text": "Profile parameter association was deleted."
        }
    ]
}
```
cachegroups

GET

Extract information about all Cache Groups.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 81: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>no</td>
<td>Return only Cache Groups that are of the type identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>

# 137: Request Example

GET /api/1.3/cachegroups?type=23  HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

fallbackToClosest  If true, Traffic Router will direct clients to peers of this Cache Group in the event that it becomes unavailable.

id  A numeric, unique identifier for the Cache Group
lastUpdated  The time and date at which this entry was last updated in ISO format
latitude  Latitude for the Cache Group
longitude  Longitude for the Cache Group
name  The name of the Cache Group entry
parentCachegroupId  ID of this Cache Group’s parent Cache Group (if any)
pARENT_CachegroupNAME  Name of this Cache Group’s parent Cache Group (if any)

secondaryParentCachegroupId  ID of this Cache Group’s secondary parent Cache Group (if any)
**secondaryParentCachegroupName**  Name of this *Cache Group*’s secondary parent *Cache Group* (if any)

**shortName**  Abbreviation of the *Cache Group* name

**typeId**  Unique identifier for the ‘Type’ of *Cache Group* entry

**typeName**  The name of the type of *Cache Group* entry

**Note:** The default value of `fallbackToClosest` is ‘true’, and if it is ‘null’ Traffic Control components will still interpret it as ‘true’.

---

# 138: Response Example

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: oV6ifEgoFy+v049tVjSsRdWQf4bxjrUvIYfDdgpUt1xiC7gzCv3lm5bXQ8EUBW4eg2hfYM+BsGvJpn
X-Server-Name: traffic_ops_golang/
Date: Wed, 07 Nov 2018 19:46:36 GMT
Content-Length: 379

{  
"response": [  
  {  
    "id": 7,  
    "name": "CDN_in_a_Box_Edge",  
    "shortName": "ciabEdge",  
    "latitude": 38.897663,  
    "longitude": -77.036574,  
    "parentCachegroupName": "CDN_in_a_Box_Mid",  
    "parentCachegroupId": 6,  
    "secondaryParentCachegroupName": null,  
    "secondaryParentCachegroupId": null,  
    "fallbackToClosest": [],  
    "localizationMethods": [],  
    "typeName": "EDGE_LOC",  
    "typeId": 23,  
    "lastUpdated": "2018-11-07 14:45:43+00"  
  }  
]  
}
```
POST

Creates a *Cache Group*

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object

**Request Structure**

- `fallbackToClosest` If `true`, the Traffic Router will fall back on the ‘closest’ *Cache Group* to this one, when this one fails

  __Note:__ The default value of `fallbackToClosest` is ‘true’, and if it is ‘null’ Traffic Control components will still interpret it as ‘true’.

- `latitude` An optional field which, if present, will define the latitude for the *Cache Group* to ISO-standard double specification

- `longitude` An optional field which, if present, will define the longitude for the *Cache Group* to ISO-standard double specification

- `localizationMethods` Array of enabled localization methods (as strings)

- `fallbacks` Array of fallback server hostnames.

- `name` The name of the *Cache Group*

- `parentCachegroupId` An optional field which, if present, should be an integral, unique identifier for this *Cache Group*’s primary parent

- `secondaryParentCachegroupId` An optional field which, if present, should be an integral, unique identifier for this *Cache Group*’s secondary parent

- `shortName` An abbreviation of the `name`

- `typeId` An integral, unique identifier for the type of *Cache Group*; one of:
  - `EDGE_LOC` Indicates a group of Edge-tier caches
  - `MID_LOC` Indicates a group of Mid-tier caches
  - `ORG_LOC` Indicates a group of origin servers (though only one server will typically be in any given ORG_LOC)

  __Note:__ The actual, integral, unique identifiers for these types must first be obtained, generally via `types`.

---

1 While these fields are technically optional, note that if they are not specified many things may break. For this reason, Traffic Portal requires them when creating or editing *Cache Groups*. 

---

278 Chapter 5. APIs
POST /api/1.1/cachegroups HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 252
Content-Type: application/x-www-form-urlencoded

{  
  "fallbackToClosest": false,
  "latitude": 0,
  "longitude": 0,
  "localizationMethods": [],
  "fallbacks": [],
  "name": "test",
  "parentCachegroupId": 7,
  "shortName": "test",
  "typeId": 23
}

Response Structure

- **fallbackToClosest**: If `true`, Traffic Router will direct clients to peers of this Cache Group in the event that it becomes unavailable.
- **id**: A numeric, unique identifier for the Cache Group
- **lastUpdated**: The time and date at which this entry was last updated in ISO format
- **latitude**: Latitude for the Cache Group
- **longitude**: Longitude for the Cache Group
- **localizationMethods**: Array of enabled localization methods (as strings)
- **fallbacks**: Array of fallback server hostnames
- **name**: The name of the Cache Group entry
- **parentCachegroupId**: ID of this Cache Group’s parent Cache Group (if any)
- **parentCachegroupName**: Name of this Cache Group’s parent Cache Group (if any)
- **secondaryParentCachegroupId**: ID of this Cache Group’s secondary parent Cache Group (if any)
- **secondaryParentCachegroupName**: Name of this Cache Group’s secondary parent Cache Group (if any)
- **shortName**: Abbreviation of the Cache Group name
**typeId**  Unique identifier for the ‘Type’ of *Cache Group* entry

**typeName**  The name of the type of *Cache Group* entry

---

**HTTP/1.1 200 OK**

Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: YvZlh3rpfl3nBq6SbNVhbkt3IvckbB9amqGW2JhLxWK9K3cxjBq5J2sIHBUhrLKhE9afpxtvaYrLR
X-Server-Name: traffic_ops_golang/
Date: Wed, 07 Nov 2018 22:11:50 GMT
Content-Length: 379

```json
{
    "alerts": [
        {
            "text": "cachegroup was created.",
            "level": "success"
        }
    ],
    "response": {
        "id": 10,
        "name": "test",
        "shortName": "test",
        "latitude": 0,
        "longitude": 0,
        "parentCachegroupName": "CDN_in_a_Box_Mid",
        "parentCachegroupId": 7,
        "secondaryParentCachegroupName": null,
        "secondaryParentCachegroupId": null,
        "fallbackToClosest": false,
        "localizationMethods": [],
        "fallbacks": [],
        "typeName": "EDGE_LOC",
        "typeId": 23,
        "lastUpdated": "2018-11-07 22:11:50+00"
    }
}
```

cachegroups/{{ID}}

Extracts information about a single *Cache Group*
GET

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 82: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a Cache Group</td>
</tr>
</tbody>
</table>

Response Structure

fallbackToClosest If true, Traffic Router will direct clients to peers of this Cache Group in the event that it becomes unavailable

id Integral, unique identifier for the Cache Group

lastUpdated The date and time at which this Cache Group was last updated, in an ISO-like format

latitude Latitude of the Cache Group

localizationMethods An array of strings that name the localization methods enabled for this Cache Group. Each of the three available localization methods may be present, with the following meanings:

CZ Lookup in the Traffic Router’s “Coverage Zone” file is enabled

DEEP_CZ Lookup in the Traffic Router’s “Deep Coverage Zone” file is enabled

GEO Use of a geographical location-to-IP mapping database is enabled

longitude Longitude of the Cache Group

name The name of the Cache Group

parentCachegroupId Integral, unique identifier of the Cache Group that is this Cache Group’s parent

parentCachegroupName The name of the Cache Group that is this Cache Group’s parent

secondaryParentCachegroupId Integral, unique identifier of the Cache Group that is this Cache Group’s secondary parent

secondaryParentCachegroupName The name of the Cache Group that is this Cache Group’s secondary parent

5.1. Traffic Ops API
shortName  Abbreviation of the Cache Group Name

typeId  The integral, unique identifier for the ‘Type’ of Cache Group

typeName  The name of the type of this Cache Group

Note: The default value of fallbackToClosest is ‘true’, and if it is ‘null’ Traffic Control components will still interpret it as ‘true’.

# 141: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: ...
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 18:35:53 GMT
Content-Length: 357

{ "response": [
  {
    "id": 8,
    "name": "test",
    "shortName": "test",
    "latitude": 0,
    "longitude": 0,
    "parentCachegroupName": "CDN_in_a_Box_Mid",
    "parentCachegroupId": 6,
    "secondaryParentCachegroupName": null,
    "secondaryParentCachegroupId": null,
    "fallbackToClosest": [],
    "localizationMethods": [
      "DEEP_CZ",
      "CZ"
    ],
    "typeName": "EDGE_LOC",
    "typeId": 23,
    "lastUpdated": "2018-11-14 18:23:33+00"
  }
]}
```
**PUT**

Update *Cache Group*

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

### Request Structure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a <em>Cache Group</em></td>
</tr>
</tbody>
</table>

**fallbackToClosest**  An optional field which, if present and true, will cause Traffic Router to direct clients to peers of this *Cache Group* in the event that it becomes unavailable.

**Note:**  The default value of *fallbackToClosest* is true, and if it is null or undefined Traffic Control components will still interpret it as true.

**latitude**  An optional field which, if specified, will set the latitude of the new *Cache Group*.

**localizationMethods**  An optional array of strings that name the localization methods enabled for this *Cache Group*. Each of the three available localization methods may be present, with the following meanings:

- **CZ**  Lookup in the Traffic Router’s “Coverage Zone” file will be enabled
- **DEEP_CZ**  Lookup in the Traffic Router’s “Deep Coverage Zone” file will be enabled
- **GEO**  Use of a geographical location-to-IP mapping database will be enabled

**longitude**  An optional field which, if specified, will set the longitude of the new *Cache Group*.

**name**  The desired name of the *Cache Group* entry

**parentCachegroup**  An optional field which, if specified, should be the integral, unique identifier of *Cache Group* to use as the new *Cache Group*’s parent

---

1 While these fields are technically optional, note that if they are not specified many things may break. For this reason, Traffic Portal requires them when creating or editing *Cache Groups*.  

---

5.1. Traffic Ops API
**secondaryParentCacheGroup** An optional field which, if specified, should be the integral, unique identifier of *Cache Group* to use as the new *Cache Group*’s parent.

**shortName** A more human-friendly abbreviation of the *Cache Group*’s name.

**typeId** The integral, unique identifier of the desired type of the new *Cache Group* - by default the valid options are: “EDGE_LOC”, “MID_LOC” or “ORG_LOC”

---

**Note:** Rather than the actual name of the type, be sure to use the “database ID” of the desired type. Typically this will require looking up the types via the API first, as the IDs of even these default types is not deterministic.

---

### # 142: Request Example

```plaintext
PUT /api/1.3/cachegroups/8 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 118
Content-Type: application/json

{"latitude": 0.0, "longitude": 0.0, "name": "test", "shortName": "test", "typeId": 23, "localizationMethods": ["GEO"]}
```

### Response Structure

**fallbackToClosest** If true, Traffic Router will direct clients to peers of this *Cache Group* in the event that it becomes unavailable.

**id** Integral, unique identifier for the *Cache Group*.

**lastUpdated** The date and time at which this *Cache Group* was last updated, in an ISO-like format.

**latitude** Latitude of the *Cache Group*.

**localizationMethods** An array of strings that name the localization methods enabled for this *Cache Group*. Each of the three available localization methods may be present, with the following meanings:

- **CZ** Lookup in the Traffic Router’s “Coverage Zone” file is enabled.
- **DEEP_CZ** Lookup in the Traffic Router’s “Deep Coverage Zone” file is enabled.
- **GEO** Use of a geographical location-to-IP mapping database is enabled.

**longitude** Longitude of the *Cache Group*. 

---
name  The name of the *Cache Group*

parentCachegroupId  Integral, unique identifier of the *Cache Group* that is this *Cache Group*’s parent

parentCachegroupName  The name of the *Cache Group* that is this *Cache Group*’s parent

secondaryParentCachegroupId  Integral, unique identifier of the *Cache Group* that is this *Cache Group*’s secondary parent

secondaryParentCachegroupName  The name of the *Cache Group* that is this *Cache Group*’s secondary parent

shortName  Abbreviation of the *Cache Group* Name

typeId  The integral, unique identifier for the ‘Type’ of *Cache Group*

typeName  The name of the type of this *Cache Group*

---

# 143: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: t1W65/2kj25QyHt0Ib0xpBaAR2sXu2kOsRZ49WjKZp/…AK5S1YWhX7VMWCUUGiN1VM4QRNqQDC/7ewhYDFUncA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 19:14:28 GMT
Content-Length: 385

{
    "alerts": [
        {
            "text": "cachegroup was updated.",
            "level": "success"
        }
    ],

    "response": {
        "id": 8,
        "name": "test",
        "shortName": "test",
        "latitude": 0,
        "longitude": 0,
        "parentCachegroupName": null,
        "parentCachegroupId": null,
        "secondaryParentCachegroupName": null,
        "secondaryParentCachegroupId": null,
        "fallbackToClosest": []
    }
}
```

(continues on next page)
"localizationMethods": [  "GEO"  ],  "typeName": "EDGE_LOC",  "typeId": 23,  "lastUpdated": "2018-11-14 19:14:28+00"
}

DELETE

Delete Cache Group. Cache Groups which have assigned servers or child Cache Groups cannot be deleted.

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type undefined

Request Structure

Table 84: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a Cache Group to be deleted</td>
</tr>
</tbody>
</table>

# 144: Request Example

DELETE /api/1.4/cachegroups/42 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

# 145: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
**cachegroups/{{ID}}/deliveryservices**

**POST**

Assigns a *Cache Group* to one or more *Delivery Services*

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

deliveryServices  The integral, unique identifiers of the *Delivery Services* to which the *Cache Group* is being assigned

# 146: Request Example

```bash
POST /api/1.3/cachegroups/8/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 25
Content-Type: application/x-www-form-urlencoded

{"deliveryServices": [2]}
```
Response Structure

**deliveryServices** An array of all Delivery Services to which the Cache Group is assigned (not just the one(s) to which it was assigned via the request)

**id** The Cache Group’s ID

**serverNames** An array of the (short) hostnames of all servers in the Cache Group

# 147: Response Example

```json
{
  "alerts": [
    {
      "text": "Delivery services successfully assigned to all the servers of cache group 8.",
      "level": "success"
    }
  ],
  "response": {
    "id": 8,
    "serverNames": [
      "foo"
    ],
    "deliveryServices": [
      2
    ]
  }
}
```

cachegroups/{{ID}}/parameters

Gets all the parameters associated with a Cache Group

See also:

Profile Parameters
GET

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 85: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a Cache Group</td>
</tr>
</tbody>
</table>

Response Structure

configFile  Configuration file associated with the parameter
id  A numeric, unique identifier for this parameter
lastUpdated  The date and time at which this parameter was last updated, in an ISO-like format
name  Name of the parameter
secure  If true, the parameter value is only visible to “admin”-role users
value  Value of the parameter

# 148: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 19:56:23 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 14 Nov 2018 23:56:23 GMT;
Vary: Accept-Encoding
Whole-Content-Sha512: DfqPtySzVMpnBYqVt/45sSRG/1pRT1QdCYuQ2OCQt79QSHLzU5e4TbDqht6ntvNP041LimKsJ5RzP1PX1n6tg==
Content-Length: 135

{ "response": [ (continues on next page)
cachegroups/{{ID}}/queue_update

POST

Queue or dequeue updates for all servers assigned to a Cache Group limited to a specific CDN.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

Request Structure

Table 86: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the Cache Group for which updates are being queued/dequeued</td>
</tr>
</tbody>
</table>

**action** The action to perform; one of “queue” or “dequeue”

**cdn** The full name of the CDN in need of update queue/dequeue

**cdnId** The integral, unique identifier for the CDN in need of update queue/dequeue

# 149: Request Example

```
POST /api/1.3/cachegroups/8(queue_update HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 42
Content-Type: application/json
```

---

1 Either ‘cdn’ or ‘cdnID’ must be in the request data (but not both).
Response Structure

**action**  The action processed, one of “queue” or “dequeue”

**cachegroupId**  The integral, unique identifier of the *Cache Group* for which updates were queued/dequeued

**cachegroupName**  The name of the *Cache Group* for which updates were queued/dequeued

**cdn**  The name of the CDN to which the queue/dequeue operation was restricted

**serverNames**  An array of the (short) hostnames of the servers within the *Cache Group* which are also assigned to the CDN specified in the "cdn" field

---

**# 150: Response Example**

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: UAcP7Lrf1UlRnfR4UqbQrJcz1k5rkrcL0tTXJTFvIUXxK1Ek12khkE4vewjDaVIhJ6YGq8jmPGQpr
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:19:46 GMT
Content-Length: 115

{
  "response": {
    "cachegroupName": "test",
    "action": "queue",
    "serverNames": [
      "foo"
    ],
    "cdn": "CDN-in-a-Box",
    "cachegroupID": 8
  }
}
```

cachegroups/{{id}}/unassigned_parameters

Gets all the parameters NOT associated with a specific *Cache Group*

See also:

---

5.1. Traffic Ops API 291
Profile Parameters

GET

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 87: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>Cache Group ID</td>
</tr>
</tbody>
</table>

Response Structure

configFile  Configuration file associated with the parameter
id  A numeric, unique identifier for this parameter
lastUpdated  The Time / Date this entry was last updated
name  Name of the parameter
secure  Is the parameter value only visible to admin users
value  Value of the parameter

# 151: Response Example

```
{ "response": [
    {
      "lastUpdated": "2018-10-09 11:14:33.862905+00",
      "value": "/opt/trafficserver/etc/trafficserver",
      "secure": false,
      "name": "location",
      "id": 6836,
      "configFile": "hdr_rw_bamtech-nhl-live.config"
    },
    {
      "lastUpdated": "2018-10-09 11:14:33.862905+00",
      "value": "/opt/trafficserver/etc/trafficserver",
      "secure": false,
      "name": "location",
      "id": 6837,
      "configFile": "hdr_rw_mid_bamtech-nhl-live.config"
    }
  ]
```

(continues on next page)
cachegroups/{{parameter ID}}/parameter/available

Deprecated since version 1.1: Use cachegroupparameters instead

**Caution:** This endpoint does not appear to work, and thus its use is strongly discouraged!

**GET**

Gets a list of Cache Groups which are available to have a specific parameter assigned to them

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

Table 88: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter ID</td>
<td>yes</td>
<td>The integral, unique identifier of the parameter of interest</td>
</tr>
</tbody>
</table>

5.1. Traffic Ops API
Response Structure

**id**  An integral, unique identifier for the *Cache Group*

**name**  The name of the *Cache Group*

---

# 152: Response Example

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

---

cachegroups/trimmed

Extract just the names of all *Cache Groups*.

**GET**

Auth. Required  Yes

Roles Required  None

Response Type  Array

---

Request Structure

No parameters available

---

Response Structure

**name**  The name of the *Cache Group*

---

# 153: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
```

(continues on next page)
caches/stats

An API endpoint that returns cache statistics using the Traffic Monitor APIs.

See also:

This gives a set of basic statistics for all caches at the current time. For statistics from time ranges and/or aggregated over a specific CDN, use cache_stats.

GET

Retrieves cache stats from Traffic Monitor. Also includes rows for aggregates.

## 5.1. Traffic Ops API
Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available.

Response Structure

cachegroup  The name of the Cache Group to which this cache belongs
connections  Current number of TCP connections maintained by the cache
healthy  true if Traffic Monitor has marked the cache as “healthy”, false otherwise

See also:

Health Protocol

hostname  The (short) hostname of the cache
ip  The IP address of the cache
kbps  Cache upload speed (to clients) in Kilobits per second
profile  The name of the profile in use by this cache
status  The status of the cache

# 154: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 20:25:01 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 00:25:01 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: DqbLgitanS8q81/qKC1i+ImMiEMF+SW4G9rb79FWdeWcgwFjL810t1TRplnNNfHV+tajqjyK+wMHobqVyaNEfA==
Content-Length: 133
```

(continues on next page)
capabilities

GET

Get all capabilities.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No available parameters

Response Structure

  name  Name of the capability
  description  Describes the APIs covered by the capability.
  lastUpdated  Date and time of the last update made to this capability, in ISO format

# 155: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

(continues on next page)
POST
Create a capability.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

name  The name of the capability being created
description  A description of what the capability allows

# 156: Request Example

POST /api/1.4/capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 109
Content-Type: application/json

{
    "name": "test",
    "description": "This is only a test. If this were a real capability, it might do something"
}

Response Structure

description  Describes the APIs covered by the capability.

name  Name of the capability

# 157: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 20:33:00 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 00:33:00 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: HhhQzw3JBLv901OeeSGj75uknADanz3fUnQt1E266HAKPTFuTju1Jpf8ni9fb9Chv9LN7mt16utcHM
Content-Length: 183

{
    "alerts": [
        {
            "level": "success",
            "text": "Capability was created."
        }
    ],
    "response": {
        "name": "test",
        "description": "This is only a test. If this were a real capability, it might do something"
    }
}

capabilities/{{name}}
GET

Get a capability by name.

Auth. Required Yes

Roles Required None

Response Type Array

Request Structure

Table 89: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the capability of interest</td>
</tr>
</tbody>
</table>

Response Structure

description Describes the APIs covered by the capability

lastUpdated Date and time of the last update made to this capability, in ISO format

name Name of the capability

# 158: Response Example

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 20:37:17 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 00:37:17 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 0YBTC5TEAOJ6B8gsaKgOD1ni2hn28Kh9u2JhmcmExoGIPaMEKpp40mr4Fg1kOQZuh/
IB90ejJBMNMeCeZ6CwRg==
Content-Length: 167

{ "response": [  
    {  
        "lastUpdated": "2018-11-14 20:33:00.275376+00",  
    }  
]  
}
```
"name": "test",
"description": "This is only a test. If this were a real capability, it might do something"
}

PUT

Edit a capability.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object

Request Structure

Table 90: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the capability to be modified</td>
</tr>
</tbody>
</table>

description Describes the APIs covered by the capability

# 159: Request Example

```
PUT /api/1.4/capabilities/test HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 45
Content-Type: application/json

{"description": "A much shorter description"}
```

Response Structure

description Describes the APIs covered by the capability.

lastUpdated Date and time of the last update made to this capability, in ISO format

name The name of the capability
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 20:40:33 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 00:40:33 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: +5mLZ/CJnDkJMbnFviXtVdjwt4bu7ykiMIs73zsnuKV/k4q/d025b2pjYDQKSgtfWPJ73FcusAuBM9TCVT3KsA==
Content-Length: 181

```json
{
  "alerts": [
    {
      "level": "success",
      "text": "Capability was updated."
    }
  ],
  "response": {
    "lastUpdated": "2018-11-14 20:33:00.275376+00",
    "name": "test",
    "description": "A much shorter description"
  }
}
```

**DELETE**

Delete a capability.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

**Request Structure**

Table 91: Request Path Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>Capability name.</td>
</tr>
</tbody>
</table>
Response Structure

# 161: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 20:45:37 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 00:45:37 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
IlAiV4ebwTpMIgeYlR5RuwhwmHs8Ekt7AaEDb3v+1XjvjkqU98xFsfNWvPbT/VJnotENhVq8Tv3vPlG==
Content-Length: 61

{ "alerts": [
    {
        "level": "success",
        "text": "Capability deleted."
    }
]}
```

cdns

Extract information about all CDNs

GET

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

No parameters available
Response Structure

- **dnssecEnabled**: true if DNSSEC is enabled on this CDN, otherwise false
- **domainName**: Top Level Domain name within which this CDN operates
- **id**: The integral, unique identifier for the CDN
- **lastUpdated**: Date and time when the CDN was last modified in ISO format
- **name**: The name of the CDN

### # 162: Response Example

```json
{
  "response": [
    {
      "dnssecEnabled": false,
      "domainName": "-",
      "id": 1,
      "lastUpdated": "2018-11-14 18:21:06+00",
      "name": "ALL"
    },
    {
      "dnssecEnabled": false,
      "domainName": "mycdn.ciab.test",
      "id": 2,
      "lastUpdated": "2018-11-14 18:21:14+00",
      "name": "CDN-in-a-Box"
    }
  ]
}
```

**POST**

Allows user to create a CDN

- **Auth. Required**: Yes
- **Roles Required**: “admin” or “operations”
Response Type  Object

Request Structure

dnssecEnabled  If true, this CDN will use DNSSEC, if false it will not

domainName  The top-level domain (TLD) belonging to the new CDN

name  Name of the new CDN

# 163: Request Structure

```
POST /api/1.4/cdns HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 63
Content-Type: application/json

{"name": "test", "domainName": "quest", "dnssecEnabled": false}
```

Response Structure

dnssecEnabled  true if the CDN uses DNSSEC, false otherwise

domainName  The top-level domain (TLD) assigned to the newly created CDN

id  An integral, unique identifier for the newly created CDN

name  The newly created CDN’s name

# 164: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 1rZRlOfQioGRrEb4nCfjGGx7y3Ub2h7BZ4z6Nbhcy4acPs1KSUNM8QLjWTvWLu4WpkfJNxc0yy8N1K...
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:49:28 GMT
Content-Length: 174

{ "alerts": [ ]
```
(continues on next page)


cdns/capacity

GET

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

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

No parameters available.

Response Structure

availablePercent  The percent of available (unused) bandwidth to 64 bits of precision

unavailablePercent  The percent of unavailable (used) bandwidth to 64 bits of precision

utilizedPercent  The percent of bandwidth currently in use to 64 bits of precision

maintenancePercent  The percent of bandwidth being used for administrative or analytical processes internal to the CDN to 64 bits of precision

# 165: Response Example

```json
{
    "response": {
        "availablePercent": 89.0939840205533,
        "unavailablePercent": 14.999999999999998,
        "utilizedPercent": 5.999999999999999,
        "maintenancePercent": 0.09999999999999999
    }
}
```

Following IEEE 754
cdns/{{cdn}}/configfiles/ats/{{filename}}

See also:
The servers/{{server}}/configfiles/ats endpoint

GET

Gets the configuration file filename from the CDN cdn (by either name or ID).

Auth. Required Yes

Roles Required “operations”

Response Type NOT PRESENT - endpoint returns custom text/plain response (represents the contents of the requested configuration file)

Request Structure

Table 92: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn</td>
<td>string or integer</td>
<td>yes</td>
<td>Either the name or integral, unique, identifier of a CDN</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>yes</td>
<td>The name of a configuration file used by cdn</td>
</tr>
</tbody>
</table>

Response Structure

Note: If the file identified by filename does exist, but is not used by the entire CDN, a JSON response will be returned and the alerts array will contain a "level": "error" node which identifies the correct scope of the configuration file.
cdns/configs

Deprecated since version 1.0: Use one of `cdns/{{name}}/configs/monitoring`, `cdns/{{name}}/configs/routing`, or `servers/{{server}}/configfiles/ats` instead.

**Caution:** This endpoint doesn’t appear to work as of Traffic Control version 3.0.0 - it is strongly advised that its used be avoided.

GET

Retrieves CDN configuration information.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

No parameters available.

**Response Properties**

- `config_file` Presumably the name of some configuration file
- `id` The integral, unique identifier for this CDN
- `name` The CDN’s name
- `value` Presumably the content of some configuration file

cdns/dnsseckeys/refresh

---

These values are currently missing from this endpoint’s output. **DO NOT count on this endpoint to provide this information.**
GET

Refresh the DNSSEC keys for all CDNs. This call initiates a background process to refresh outdated keys, and immediately returns a response that the process has started.

**Auth. Required**  Yes

**Roles Required**  “admin”

**Response Type**  Object (string)

**Request Structure**

No parameters available

**Response Structure**

# 167: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 21:37:30 GMT
X-Server-Name: traffic_ops_golang/
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 01:37:30 GMT;
Vary: Accept-Encoding
Whole-Content-Sha512: Uwl+924m6Ye3NraFP+RBpldhcNTTDyXHZbzRaYV95p9tP56Z61gckeKSr1oQIKNXjXcCsDN5Dum7
Content-Length: 69

{
   "response": "Checking DNSSEC keys for refresh in the background"
}
```

cdns/domains

GET

Gets a list of domains and their related Traffic Router profiles for all CDNs.

**Auth. Required**  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available.

Response Structure

domainName  The top-level domain (TLD) assigned to this CDN
parameterId  The integral, unique identifier for the parameter that sets this TLD on the Traffic Router
profileDescription  A short, human-readable description of the Traffic Router’s profile
profileId  The integral, unique identifier for the profile assigned to the Traffic Router responsible for serving domainName
profileName  The name of the profile assigned to the Traffic Router responsible for serving domainName

# 168: Response Example

```
{  "response": [  
    {  
      "profileId": 12,
      "parameterId": -1,
      "profileName": "CCR_CIAB",
      "profileDescription": "Traffic Router for CDN-In-A-Box",
      "domainName": "mycdn.ciab.test"
    }
  ]}
```

cdns/health

Extract health information from all Cache Groups across all CDNs

See also:

Health Protocol

GET

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

No parameters available

Response Structure

cachegroups  An array of objects describing the health of each Cache Group

name  The name of the Cache Group
offline  The number of OFFLINE caches in the Cache Group
online  The number of ONLINE caches in the Cache Group

totalOffline  Total number of OFFLINE caches across all Cache Groups which are assigned to any CDN

totalOnline  Total number of ONLINE caches across all Cache Groups which are assigned to any CDN

# 169: Response Example

```json
{
  "response": {
    "totalOffline": 0,
    "totalOnline": 1,
    "cachegroups": [
      {
        "offline": 0,
        "name": "CDN_in_a_Box_Edge",
        "online": 1
      }
    ]
  }
}
```

cdns/{{ID}}

GET

Extract information about a specific CDN.

Auth. Required  Yes
Roles Required  None
Response Type  Array
Request Structure

Table 93: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integral, unique identifier for the CDN to inspect</td>
</tr>
</tbody>
</table>

Response Structure

- **dnssecEnabled**: `true` if DNSSEC is enabled on this CDN, otherwise `false`
- **domainName**: Top Level Domain name within which this CDN operates
- **id**: The integral, unique identifier for the CDN
- **lastUpdated**: Date and time when the CDN was last modified in ISO format
- **name**: The name of the CDN

# 170: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: bTz86xdnGfbKhxneb4geXohaw31hG+h5wc2I/\nnchFATwp1h80h+txxySCIVFa0hgBrJHEdpG2QsH5w5IknrQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:59:34 GMT
Content-Length: 137

{
    "response": [
        {
            "dnssecEnabled": false,
            "domainName": "mycdn.ciab.test",
            "id": 2,
            "lastUpdated": "2018-11-14 18:21:14+00",
            "name": "CDN-in-a-Box"
        }
    ]
}
```

PUT

Allows a user to edit a specific CDN

**Auth. Required**: Yes
Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

Table 94: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integral, unique identifier for the CDN to update</td>
</tr>
</tbody>
</table>

dnssecEnabled  If true, this CDN will use DNSSEC, if false it will not

domainName  The top-level domain (TLD) belonging to the CDN

name  Name of the new CDN

# 171: Request Example

```plaintext
PUT /api/1.4/cdns/3 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 63
Content-Type: application/json

{"name": "quest", "domainName": "test", "dnssecEnabled": false}
```

Response Structure

dnssecEnabled  true if the CDN uses DNSSEC, false otherwise

domainName  The top-level domain (TLD) assigned to the newly created CDN

id  An integral, unique identifier for the newly created CDN

name  The newly created CDN’s name

# 172: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
  Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: sI1hzBwG+/
  VAzoFY20kqGFA2RqrU0ThtMeeJgk0ZxH3TRxTWuA8BetACct/
  XICC3n7hPDL1RVpwckEyBdyJkXg==
```

(continues on next page)
DELETE

Allows a user to delete a specific CDN

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type undefined

Request Structure

Table 95: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the CDN to delete</td>
</tr>
</tbody>
</table>

Response Structure

# 173: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
```
cdns/{ID}/queue_update

POST

Queue or dequeue updates for all servers assigned to a specific CDN.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the CDN on which to (de)queue updates</td>
</tr>
</tbody>
</table>

action One of “queue” or “dequeue” as appropriate

# 174: Request Example

POST /api/1.4/cdns/2/queue_update HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 19
Content-Type: application/json

{"action": "queue"}
Response Structure

**action**  The action processed, either "queue" or "dequeue"

**cdnId**  The integral, unique identifier for the CDN on which updates were (de)queued

# 175: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: rBpFfrrP+9IFkwsRloEM+v+I8MuBZXqFu+WUTGtRGypnAn2gHooPoNQRyVvJGjyIQR1XLvqjEtve+
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:02:07 GMT
Content-Length: 41

{ "response": {
    "action": "queue",
    "cdnId": 2
}}
```

cdns/{{ID}}/snapshot

Deprecated since version 1.1: Use snapshot/{{name}} instead.

**PUT**

Performs a CDN snapshot. Effectively, this propagates the new *configuration* of the CDN to its *operating state*, which replaces the output of the cdns/{{name}}/snapshot endpoint with the output of the cdns/{{name}}/snapshot/new endpoint.

**Note:**  Snapshotting the CDN also deletes all HTTPS certificates for every *Delivery Service* which has been deleted since the last CDN *Snapshot*.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  undefined
Request Structure

Table 97: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the CDN for which a snapshot shall be taken</td>
</tr>
</tbody>
</table>

# 176: Request Example

```
PUT /api/1.4/cdns/2/snapshot HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=... |
```

Response Structure

# 177: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXcg/
→Sp1dNs6c5H0NE8XYXysP+DGKNKfuwvY7kxvUdBeoGlODJ6+SfaPg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 12 Dec 2018 22:04:46 GMT
Content-Length: 0
Content-Type: text/plain; charset=utf-8

cdns/metric_types/{{metric}}/start_date/{{start}}/end_date/{{end}}
```

Caution: This API endpoint does not work. It isn’t implemented in Traffic Ops, and is not expected to be added at any point in the near future. See GitHub issue #2309 for more information.

Deprecated since version 1.2: Use Traffic Stats or `cache_stats` instead
GET

Retrieves edge metrics of one or all Cache Groups.

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

### Request Structure

Table 98: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric_type</td>
<td>yes</td>
<td>ooff, origin_tps</td>
</tr>
<tr>
<td>start</td>
<td>yes</td>
<td>UNIX time, yesterday, now</td>
</tr>
<tr>
<td>end</td>
<td>yes</td>
<td>UNIX time, yesterday, now</td>
</tr>
</tbody>
</table>

### Response Structure

**stats** object

- **count** string
- **98thPercentile** string
- **min** string
- **max** string
- **5thPercentile** string
- **95thPercentile** string
- **mean** string
- **sum** string

**data** array

- **time** int
- **value** number

**label** string

# 178: Response Example

```json
{ "response": [ { "stats": { "count": 1, "98thPercentile": 1668.03, ...
```
cdns/{{name}}/configs/monitoring

See also:

Health Protocol

GET

Retrieves information concerning the monitoring configuration for a specific CDN.

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which monitoring configuration will be fetched</td>
</tr>
</tbody>
</table>

Table 99: Request Path Parameters
Response Structure

**cacheGroups** An array of objects representing each of the Cache Groups being monitored within this CDN

  **coordinates** An object representing the physical location of this Cache Group
   - **latitude** The geographic latitude of this Cache Group
   - **longitude** The geographic longitude of this Cache Group

  **name** The name of this Cache Group

**config** A collection of parameters used to configure the monitoring behaviour of Traffic Monitor

  **hack.ttl** Unknown

  **health.event-count** The total number of health events to store

  **health.polling.interval** An interval in milliseconds on which to poll for health statistics

  **health.threadPool** The number of threads to be used for health polling

  **health.timepad** A ‘padding time’ to add to requests to spread them out for Traffic Control systems that use a large number of Traffic Monitors

  **tm.crConfig.polling.url** The URL from which a CRConfig can be obtained

  **tm.dataServer.polling.url** The URL from which a list of data servers can be obtained

  **tm.healthParams.polling.url** The URL from which a list of health-polling parameters can be obtained

  **tm.polling.interval** The interval at which to poll for configuration updates

**deliveryServices** An array of objects representing each Delivery Service provided by this CDN

  **status** The Delivery Service’s status

  **totalKbpsThreshold** A threshold rate of data transfer this Delivery Service is configured to handle, in Kilobits per second

  **totalTpsThreshold** A threshold amount of transactions per second that this Delivery Service is configured to handle
xmlId  An integral, unique identifier for this Deliver Service (named “xmlId” for legacy reasons)

profiles  An array of the profiles in use by the *cache servers* and *Delivery Services* belonging to this CDN

  name  The profile’s name

parameters  An array of the parameters in this profile that relate to monitoring configuration. This can be null if the servers using this profile cannot be monitored (e.g. Traffic Routers)

  health.connection.timeout  A timeout value, in milliseconds, to wait before giving up on a health check request

  health.polling.url  A URL to request for polling health. Substitutions can be made in a shell-like syntax using the properties of an object from the "trafficServers" array

  health.threshold.availableBandwidthInKbps  The total amount of bandwidth that servers using this profile are allowed, in Kilobits per second. This is a string and using comparison operators to specify ranges, e.g. “>10” means “more than 10 kbps”

  health.threshold.loadavg  The UNIX loadavg at which the server should be marked “unhealthy” - see man uptime

  health.threshold.queryTime  The highest allowed length of time for completing health queries (after connection has been established) in milliseconds

history.count  The number of past events to store; once this number is reached, the oldest event will be forgotten before a new one can be added

type  The type of the profile

trafficMonitors  An array of objects representing each Traffic Monitor that monitors this CDN (this is used by Traffic Monitor’s “peer polling” function)

  fqdn  AN FQDN that resolves to the IP (and/or IPv6) address of the server running this Traffic Monitor instance

  hostname  The hostname of the server running this Traffic Monitor instance

  ip6  The IPv6 address of this Traffic Monitor - when applicable
ip  The IP address of this Traffic Monitor
port  The port on which this Traffic Monitor listens for incoming connections
profile  The name of the profile assigned to this Traffic Monitor
status  The status of the server running this Traffic Monitor instance
trafficServers  An array of objects that represent the caches being monitored within this CDN

cacheGroup  The Cache Group to which this cache belongs
fqdn  A Fully Qualified Domain Name (FQDN) that resolves to the cache server’s IP (or IPv6) address
hashId  The short name for the cache server - named “hashId” for legacy reasons
hostName  The (short) hostname of the cache server
interfacename  The name of the network interface device being used by the cache’s HTTP proxy
ip6  The cache’s IPv6 address - when applicable
ip  The cache’s IP address
port  The port on which the cache listens for incoming connections
profile  The name of the profile assigned to this cache
status  The status of the Cache
type  The type of the cache - should be either EDGE or MID

# 179: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: uLR+troqR8Syo38j3DV9wQ+IkJ7Kf+MCfKcWZtsgbplJ+0S6f+IiI8laNVeDgrM/P23MAQ6BSepm+EJRlAXQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:09:31 GMT
Transfer-Encoding: chunked
```
(continues on next page)
{  "response": {  "trafficServers": [  {  "profile": "ATS_EDGE_TIER_CACHE",  "status": "REPORTED",  "ip": "172.16.239.100",  "ip6": "fc01:9400:1000:8::100",  "port": 80,  "cachegroup": "CDN_in_a_Box_Edge",  "hostname": "edge",  "fqdn": "edge.infra.ciab.test",  "interfacename": "eth0",  "type": "EDGE",  "hashid": "edge" },  {  "profile": "ATS_MID_TIER_CACHE",  "status": "REPORTED",  "ip": "172.16.239.120",  "ip6": "fc01:9400:1000:8::120",  "port": 80,  "cachegroup": "CDN_in_a_Box_Mid",  "hostname": "mid",  "fqdn": "mid.infra.ciab.test",  "interfacename": "eth0",  "type": "MID",  "hashid": "mid" }  ],  "trafficMonitors": [  {  "profile": "RASCAL-Traffic_Monitor",  "status": "ONLINE",  "ip": "172.16.239.40",  "ip6": "fc01:9400:1000:8::40",  "port": 80,  "cachegroup": "CDN_in_a_Box_Edge",  "hostname": "trafficmonitor",  "fqdn": "trafficmonitor.infra.ciab.test"  }  ],  "cacheGroups": [  {  "name": "CDN_in_a_Box_Mid",  "coordinates": {  "latitude": 38.897663,  "longitude": -77.036574  }  }  ]  }  (continues on next page)


```json
{
    "name": "CDN_in_a_Box_Edge",
    "coordinates": {
        "latitude": 38.897663,
        "longitude": -77.036574
    }
}
"profiles": [
    {
        "name": "CCR_CIAB",
        "type": "CCR",
        "parameters": null
    },
    {
        "name": "ATS_EDGE_TIER_CACHE",
        "type": "EDGE",
        "parameters": {
            "health.connection.timeout": 2000,
            "health.polling.url": "http://$\{hostname\}/_astats?application=&inf.name=${interface_name}"
        },
        "availableBandwidthInKbps": ">1750000",
        "health.threshold.loadavg": "25.0",
        "health.threshold.queryTime": 1000,
        "history.count": 30
    },
    {
        "name": "ATS_MID_TIER_CACHE",
        "type": "MID",
        "parameters": {
            "health.connection.timeout": 2000,
            "health.polling.url": "http://$\{hostname\}/_astats?application=&inf.name=${interface_name}"
        },
        "availableBandwidthInKbps": ">1750000",
        "health.threshold.loadavg": "25.0",
        "health.threshold.queryTime": 1000,
        "history.count": 30
    }
],
"deliveryServices": [],
"config": {
    "hack.ttl": 30,
    "health.event-count": 200,
    "health.polling.interval": 6000,
    "health.threadPool": 4,

(continues on next page)
```
"health.timepad": 0,
"heartbeat.polling.interval": 3000,
"location": "/opt/traffic_monitor/conf",
"peers.polling.interval": 3000,
"tm.crConfig.polling.url": "https://${tmHostname}/CRConfig-Snapshots/${cdnName}/CRConfig.xml",
"tm.dataServer.polling.url": "https://${tmHostname}/dataserver/orderby/id",
"tm.healthParams.polling.url": "https://${tmHostname}/health/${cdnName}",
"tm.polling.interval": 2000
}
}

```c
cdns/{{name}}/configs/routing
```

**Caution:** This API route is currently broken, see GitHub issue #2941 for more information.

**GET**

Retrieves CDN routing information.

**Auth. Required** Yes

**Roles Required** None

**Response Type**

**Request Structure**

Table 100: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>The name of the CDN to be inspected</td>
</tr>
</tbody>
</table>

**Response Structure**

- **cacheGroups** A collection of cache groups.
  - **coordinates** object
    - **latitude** number
    - **longitude** number
name  string

cfg  object

coveragezone.polling.url  string
domain_name  string
geolocation.polling.interval  integer
geolocation.polling.url  string
geolocation6.polling.interval  integer
geolocation6.polling.url  string
tcoveragezone.polling.interval  integer
tld.soa.admin  string
tld.soa.expire  integer
tld.soa.minimum  integer
tld.soa.refresh  integer
tld.soa.retry  integer
tld.ttls.A  integer
tld.ttls.AAAA  integer
tld.ttls.NS  integer
tld.ttls.SOA  integer
deliveryServices  An array of delivery services.
coverageZoneOnly  boolean
bypassDestination  object
    maxDnsIpsForLocation  integer
ttl  integer
type  string
geoEnabled  string
matchSets  array
    protocol  string
matchList  array
        matchType  string
        regex  string
missCoordinates  object
    latitude  number
longitude number

soa object
  admin string
  expire integer
  minimum integer
  refresh integer
  retry integer

ttl integer

ttls object
  A integer
  AAAA integer
  NS integer
  SOA integer

xmlId string

stats object
  cdnName string
  date integer
  trafficOpsHost string
  trafficOpsPath string
  trafficOpsUser string
  trafficOpsVersion string

trafficMonitors An array of Traffic Monitors
  fqdn string
  hostName string
  ip6 string
  ip string
  location string
  port integer
  profile string
  status string

trafficRouters object
  apiPort integer
fqdn  string
hostName  string
ip6  string
ip  string
location  string
port  integer
profile  integer
status  string

trafficServers  An array of Traffic Servers.

cacheGroup  string
deliveryServices  array

  xmlId  string
  remaps  array
  
hostName  string

fqdn  string
hashId  string
interfaceName  string
ip  string
ip6  string
port  integer
profile  string
status  string
type  string

cdns/{name}/dnsseckeys/ksk/generate

New in version 1.4.

POST

Generates a new Key-Signing Key (KSK) for a specific CDN.

  Auth. Required  Yes
  
  Roles Required  “admin”
  
  Response Type  Object (string)
Request Structure

Table 101: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>The name of the CDN for which the KSK will be generated</td>
</tr>
</tbody>
</table>

Table 102: Request Data Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expirationDays</td>
<td>yes</td>
<td>int</td>
<td>The number of days until the newly generated KSK expires</td>
</tr>
<tr>
<td>effectiveDate</td>
<td>no</td>
<td>str</td>
<td>The time at which the newly generated KSK becomes effective, in RFC3339 format - defaults to the current time</td>
</tr>
</tbody>
</table>

Response Structure

# 180: Response Example

```json
{ "response": "Successfully generated ksk dnssec keys for my-cdn-name" } 
```

`cdns/{{name}}/federations`  

**GET**

Retrieves a list of federations in use by a specific CDN.

**Auth. Required** Yes  
**Roles Required** None  
**Response Type** Array

Request Structure

Table 103: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which federations will be listed</td>
</tr>
</tbody>
</table>
# 181: Request Example

```plaintext
GET /api/1.4/cdns/CDN-in-a-Box/federations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

## Response Structure

**cname**  The Canonical Name (CNAME) used by the federation

**deliveryService**  An object with keys that provide identifying information for the *Delivery Service* using this federation

  **id**  The integral, unique identifier for the *Delivery Service*

  **xmlId**  The *Delivery Service*’s uniquely identifying ‘xml_id’

**description**  An optionally-present field containing a description of the field

---

*Note:*  This key will only be present if the description was provided when the federation was created. Refer to the POST method of this endpoint to see how federations can be created.

**lastUpdated**  The date and time at which this federation was last modified, in ISO format

**ttl**  Time to Live (TTL) for the *cname*, in hours

---

# 182: Response Example

```plaintext
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-
  Type, Accept, Set-Cookie, Cookie
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
content-type: application/json
set-cookie: mojolicious=...; Path=/; HttpOnly
whole-content-sha512:
  SJA7G+7G5KcOfCtnE3Dq5DCobWtGRUKSpipiDkfLZoG5+paq4ElazGqUb6vGVsd+TpPg75ML1hyqfdf
  g==
x-server-name: traffic_ops_golang/
content-length: 170
date: Wed, 05 Dec 2018 00:35:40 GMT

{ "response": [ }  
```
POST

Creates a new federation.

Auth. Required Yes
Roles Required “admin”
Response Type Object

Request Structure

Table 104: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which a new federation will be created</td>
</tr>
</tbody>
</table>

cname The Canonical Name (CNAME) used by the federation

Note: The CNAME must end with a “.”

description An optional description of the federation
ttl Time to Live (TTL) for the name record used for cname

# 183: Request Example

```
POST /api/1.4/cdns/CDN-in-a-Box/federations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 72
```
Content-Type: application/json
{
    "cname": "test.quest.",
    "ttl": 48,
    "description": "A test federation"
}

Response Structure

cname The Canonical Name (CNAME) used by the federation
description An optionally-present field containing a description of the field

Note: This key will only be present if the description was provided when the federation was created

lastUpdated The date and time at which this federation was last modified, in ISO format
ttl Time to Live (TTL) for the cname, in hours

# 184: Response Example

HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
content-type: application/json
set-cookie: mojolicious=...; Path=/; HttpOnly
whole-content-sha512: rRsWAihXzVlJ8Hy+8afjp4JolQGTK49m0N1AP5QDyyA2I7fNldgtciju7u7F1n1PWRFiv6D9CySF
x-server-name: traffic_ops_golang/
content-length: 192
date: Wed, 05 Dec 2018 00:05:16 GMT

{ "alerts": [
    {
        "text": "cdnfederation was created.",
        "level": "success"
    }
],
"response": {
    "id": 1,
    "cname": "test.quest."
}
cdns/{{name}}/federations/{{ID}}

GET

Retrieves a specific federation used within a specific CDN.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

### Request Structure

#### Table 105: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which the federation identified by ID will be inspected</td>
</tr>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the federation to be inspected</td>
</tr>
</tbody>
</table>

#### # 185: Request Example

```
GET /api/1.4/cdns/CDN-in-a-Box/federations/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

- **cname** The Canonical Name (CNAME) used by the federation
- **deliveryService** An object with keys that provide identifying information for the *Delivery Service* using this federation
  - **id** The integral, unique identifier for the *Delivery Service*
  - **xmlId** The *Delivery Service’s* uniquely identifying ‘xml_id’
- **description** An optionally-present field containing a description of the field
**Note:** This key will only be present if the description was provided when the federation was created. Refer to the POST method of the `cdns/{name}/federations` endpoint to see how federations can be created.

- **lastUpdated** The date and time at which this federation was last modified, in ISO format
- **ttl** Time to Live (TTL) for the `cname`, in hours

### # 186: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
content-type: application/json
set-cookie: mojolicious=...; Path=/; HttpOnly
whole-content-sha512:
  SJA7G+7G5Kc0fCtnE3Dq5DCobWtGRUKspipDkfLzoG5+paq4E1a2GqUb6vGVsd+TpPg75ML1hyqdfda...g==
x-server-name: traffic_ops_golang/
content-length: 170
date: Wed, 05 Dec 2018 00:36:57 GMT
{
  "response": [
    {
      "id": 1,
      "cname": "test.quest.",
      "ttl": 48,
      "description": "A test federation",
      "lastUpdated": "2018-12-05 00:05:16+00",
      "deliveryService": {
        "id": 1,
        "xmlId": "demo1"
      }
    }
  ]
}
```

**PUT**

Updates a federation.

- **Auth. Required** Yes
- **Roles Required** “admin”
- **Response Type** Object
Request Structure

Table 106: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which the federation identified by ID will be inspected</td>
</tr>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the federation to be inspected</td>
</tr>
</tbody>
</table>

- **cname**: The Canonical Name (CNAME) used by the federation

  **Note**: The CNAME must end with a “.”

- **description**: An optional description of the federation

- **ttl**: Time to Live (TTL) for the name record used for **cname**

# 187: Request Example

```
PUT /api/1.4/cdns/CDN-in-a-Box/federations/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 33
Content-Type: application/json

{
    "cname": "foo.bar."
    "ttl": 48
}
```

Response Structure

- **cname**: The Canonical Name (CNAME) used by the federation

- **description**: An optionally-present field containing a description of the field

  **Note**: This key will only be present if the description was provided when the federation was created

- **lastUpdated**: The date and time at which this federation was last modified, in ISO format

- **ttl**: Time to Live (TTL) for the **cname**, in hours
# 188: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
content-type: application/json
set-cookie: mojolicious=...; Path=/; HttpOnly
whole-content-sha512: qcjfQ+gDjNxYQ1aq+dlddgrkFWnkFYxsFF+SHDqqH0uVHBVksmU0aTFgltozek/
                        u6wbrGoR1LFf9Fr1C1SbigA==
x-server-name: traffic_ops_golang/
content-length: 174
date: Wed, 05 Dec 2018 01:03:40 GMT

{
    "alerts": [
        {
            "text": "cdnfederation was updated.",
            "level": "success"
        }
    ],
    "response": {
        "id": 1,
        "cname": "foo.bar.",
        "ttl": 48,
        "description": null,
        "lastUpdated": "2018-12-05 01:03:40+00"
    }
}
```

DELETE

Deletes a specific federation.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** undefined

## Request Structure

Table 107: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which the federation identified by ID will be inspected</td>
</tr>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the federation to be inspected</td>
</tr>
</tbody>
</table>
# 189: Request Example

```
DELETE /api/1.4/cdns/CDN-in-a-Box/federations/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

# 190: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
content-type: application/json
set-cookie: mojolicious=...; Path=/; HttpOnly
whole-content-sha512: Cnkfj6dmzTD3if9oiDq33tqf7CnAf1KK/\SPqyfu6HUFoJLQgKIZvkcs2wWY6EjLVdw5qsatsd4FPoCyjvcw==
x-server-name: traffic_ops_golang/
content-length: 68
date: Wed, 05 Dec 2018 01:17:24 GMT

{
  "alerts": [
    {
      "text": "cdnfederation was deleted.",
      "level": "success"
    }
  ]
}
```

cdns/\{name\}/health

**GET**

Retrieves the health of all *Cache Groups* for a given CDN.

**Auth. Required**  Yes  
**Roles Required**  None  
**Response Type**  Object
Request Structure

Table 108: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which health will be reported</td>
</tr>
</tbody>
</table>

Response Structure

cachegroups An array of objects describing the health of each Cache Group

name The name of the Cache Group

offline The number of OFFLINE caches in the Cache Group

online The number of ONLINE caches in the Cache Group

totalOffline Total number of OFFLINE caches across all Cache Groups which are assigned to the CDN defined by the name request path parameter

totalOnline Total number of ONLINE caches across all Cache Groups which are assigned to the CDN defined by the name request path parameter

# 191: Response Example

```json
{
  "response": {
    "totalOffline": 0,
    "totalOnline": 1,
    "cachegroups": [
      {
        "offline": 0,
        "name": "CDN_in_a_Box_Edge",
        "online": 1
      }
    ]
  }
}
```
cdns/name/{{name}}

GET

Extract information about a CDN, identified by name.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 109: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN to be inspected</td>
</tr>
</tbody>
</table>

Response Structure

dnssecEnabled true if DNSSEC is enabled on this CDN, otherwise false
domainName Top Level Domain name within which this CDN operates
id The integral, unique identifier for the CDN
lastUpdated Date and time when the CDN was last modified in ISO format
name The name of the CDN

# 192: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: bTz86xdnGfbKhxnneb4geXohaw31hG+h5wc21/
\-
\-ncHFATwp1h80h+txxySCIVfa0hgBrJHEdpGZQsH5w5IknrQ==

(continues on next page)
DELETE

Allows a user to delete a CDN by name

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

**Request Structure**

Table 110: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN to be deleted</td>
</tr>
</tbody>
</table>

**Response Structure**

# 193: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
```

(continues on next page)
cdns/name/{{name}}/dnsseckeys

GET

Gets a list of DNSSEC keys for CDN and all associated Delivery Services. Before returning response to user, this will make sure DNSSEC keys for all delivery services exist and are not expired. If they don’t exist or are expired, they will be (re-)generated.

Auth. Required Yes

Roles Required “admin”

Response Type Object

Request Structure

Table 111: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which keys will be fetched</td>
</tr>
</tbody>
</table>

Response Structure

- **name** The name of the CDN or Delivery Service to which the enclosed keys belong

  - **zsk** The short-term Zone-Signing Key (ZSK)

  - **expirationDate** A Unix epoch timestamp (in seconds) representing the date and time whereupon the key will expire

  - **inceptionDate** A Unix epoch timestamp (in seconds) representing the date and time when the key was created
name  The name of the domain for which this key will be used

private  Encoded private key

public  Encoded public key

ttl  The time for which the key should be trusted by the client

ksk  The long-term Key-Signing Key (KSK)

dsRecord  An optionally present object containing information about the algorithm used to generate the key

  algorithm  The name of the algorithm used to generate the key

  digest  A hash of the DNSKEY record

  digestType  The type of hash algorithm used to create the value of digest

expirationDate  A Unix epoch timestamp (in seconds) representing the date and time whereupon the key will expire

inceptionDate  A Unix epoch timestamp (in seconds) representing the date and time when the key was created

name  The name of the domain for which this key will be used

private  Encoded private key

public  Encoded public key

ttl  The time for which the key should be trusted by the client

Changed in version 1.2: Added the dsRecord field to KSK entries

# 194: Response Example

```json
{
  "response": {
    "cdn1": {
      "zsk": {
        "ttl": "60",
        "inceptionDate": "1426196750",
        "private": "zsk private key",
        "public": "zsk public key",
        "expirationDate": "1428788750",
        "name": "foo.kabletown.com."
      }
    }
  }
}
```

(continues on next page)
"ksk": {
    "name": "foo.kabletown.com.",
    "expirationDate": "1457732750",
    "public": "ksk public key",
    "private": "ksk private key",
    "inceptionDate": "1426196750",
    "ttl": "60",
    "dsRecord": {
        "algorithm": "5",
        "digestType": "2",
        "digest": "abc123def456"
    }
},
"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"
    }
}

```
"cdns/name/{{name}}/dnsseckeys/delete"
```

**GET**

Delete DNSSEC keys for a CDN and all associated Delivery Services.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** Object (string)
Request Structure

Table 112: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which DNSSEC keys will be deleted</td>
</tr>
</tbody>
</table>

Response Structure

# 195: Response Example

```
{
    "response": "Successfully deleted dnssec keys for test"
}
```

`cdns/name/{{name}}/sslkeys`

GET

Returns SSL certificates for all Delivery Services that are a part of the CDN.

- **Auth. Required**: Yes
- **Roles Required**: “admin”
- **Response Type**: Array

Request Structure

Table 113: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which keys will be fetched</td>
</tr>
</tbody>
</table>

Response Structure

- **certificate**: An object representing The SSL keys used for the Delivery Service identified by deliveryservice
  - **key**: Base 64-encoded private key for SSL certificate
  - **crt**: Base 64-encoded SSL certificate
- **deliveryservice**: The xml_id of the Delivery Service using the SSL key within certificate
# 196: Response Example

```json
{
  "response": [
    {
      "deliveryservice": "ds1",
      "certificate": {
        "crt": "base64encodedcrt1",
        "key": "base64encodedkey1"
      }
    },
    {
      "deliveryservice": "ds2",
      "certificate": {
        "crt": "base64encodedcrt2",
        "key": "base64encodedkey2"
      }
    }
  ]
}
```

cdns/{{name}}/snapshot

**Caution:** This page is a stub! Much of it may be missing or just downright wrong - it needs a lot of love from people with the domain knowledge required to update it.

### GET

Retrieves the *current* snapshot for a CDN, which represents the current *operating state* of the CDN, not the current *configuration* of the CDN. The contents of this snapshot are currently used by Traffic Monitor and Traffic Router.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

### Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which a snapshot shall be returned</td>
</tr>
</tbody>
</table>
# 197: Request Example

```
GET /api/1.4/cdns/CDN-in-a-Box/snapshot HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

- **config**: An object containing basic configurations on the actual CDN object
  - `api.cache-control.max-age`: A string containing an integer which specifies the value of `max-age` in the `Cache-Control` header of some HTTP responses, likely the Traffic Router API responses.
    - Deprecated since version 1.1: This field still exists for legacy compatibility reasons, but has no known use at the time of this writing.
  - `certificates.polling.interval`: A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for updated SSL certificates.
  - `consistent.dns.routing`: A string containing a boolean which indicates whether DNS routing will use a consistent hashing method or “round-robin”.
    - “false”: The “round-robin” method will be used to define DNS routing.
    - “true”: A consistent hashing method will be used to define DNS routing.
  - `coveragezone.polling.interval`: A string containing an integer which specifies the interval, in seconds, on which Traffic Routers should check for a new Coverage Zone file.
  - `coveragezone.polling.url`: The URL where a Coverage Zone file may be requested by Traffic Routers.
  - `dnssec.dynamic.response.expiration`: A string containing a number and unit suffix that specifies the length of time for which dynamic responses to DNSSEC lookup queries should remain valid.
  - `dnssec.enabled`: A string that tells whether or not the CDN uses DNSSEC; one of:
    - “false”: DNSSEC is not used within this CDN.
“true” DNSSEC is used within this CDN

**domain_name** The Top-Level Domain Name (TLD) served by the CDN

**edge.dns.limit** This field is of unknown use, and may be remnants of a legacy system

**edge.dns.routing** This field is of unknown use, and may be remnants of a legacy system

**edge.http.limit** This field is of unknown use, and may be remnants of a legacy system

**edge.http.routing** This field is of unknown use, and may be remnants of a legacy system

**federationmapping.polling.interval** A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for new federation mappings

**federationmapping.polling.url** The URL where Traffic Control components can request federation mappings

**geolocation.polling.interval** A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for new IP-to-geographic-location mapping databases

**geolocation.polling.url** The URL where Traffic Control components can request IP-to-geographic-location mapping database files

**keystore.maintenance.interval** A string containing an integer which specifies the interval, in seconds, on which Traffic Routers should refresh their zone caches

**neustar.polling.interval** A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for new “Neustar” databases

**neustar.polling.url** The URL where Traffic Control components can request “Neustar” databases

**soa** An object defining the Start of Authority (SOA) for the CDN’s TLD (defined in **domain_name**)

**admin** The name of the administrator for this zone - i.e. the RNAME

---

**Note:** This rarely represents a proper email
address, unfortunately.

**expire**  A string containing an integer that sets the number of seconds after which secondary name servers should stop answering requests for this zone if the master does not respond.

**minimum**  A string containing an integer that sets the Time To Live (TTL) - in seconds - of the record for the purpose of negative caching.

**refresh**  A string containing an integer that sets the number of seconds after which secondary name servers should query the master for the SOA record, to detect zone changes.

**retry**  A string containing an integer that sets the number of seconds after which secondary name servers should retry to request the serial number from the master if the master does not respond.

**Note:**  RFC 1035 dictates that this should always be less than refresh.

**See also:**

The Wikipedia page on Start of Authority records.

**steeringmapping.polling.interval**  A string containing an integer which specifies the interval, in seconds, on which Traffic Control components should check for new steering mappings.

**ttls**  An object that contains keys which are types of DNS records that have values which are strings containing integers that specify the time for which a response to the specific type of record request should remain valid.

**zonemanager.cache.maintenance.interval**  A configuration option for the ZoneManager Java class of Traffic Router.

**zonemanager.threadpool.scale**  A configuration option for the ZoneManager Java class of Traffic Router.

**contentRouters**  An object containing keys which are the (short) hostnames of the Traffic Routers that serve requests for Delivery Services in this CDN.

**api.port**  A string containing the port number on which the Traffic Router API is served by this Traffic Router.
fqdn This Traffic Router’s Fully Qualified Domain Name (FQDN)

httpsPort The port number on which this Traffic Router listens for incoming HTTPS requests

ip This Traffic Router’s IPv4 address

ip6 This Traffic Router’s IPv6 address

location The name of the Cache Group to which this Traffic Router belongs

port The port number on which this Traffic Router listens for incoming HTTP requests

profile The name of the profile used by this Traffic Router

status The health status of this Traffic Router

See also:

Health Protocol

collectionServers An object containing keys which are the (short) hostnames of the Edge-Tier cache server s in the CDN; the values corresponding to those keys are routing information for said servers

cacheGroup The name of the Cache Group to which the server belongs

deliveryServices An object containing keys which are the names of Delivery Services to which this cache server is assigned; the values corresponding to those keys are arrays of FQDNs that resolve to this cache server

Note: Only Edge-tier cache server s can be assigned to a Delivery Service, and therefore this field will only be present when type is "EDGE".

fqdn The server’s Fully Qualified Domain Name (FQDN)

hashCount The number of servers to be placed into a single “hash ring” in Traffic Router

hashId A unique string to be used as the key for hashing servers - as of version 3.0.0 of Traffic Control, this is always the same as the server’s (short) hostname and only still exists for legacy compatibility reasons

httpsPort The port on which the cache server listens for incoming HTTPS requests

interfaceName The name of the main network interface device used by this cache server
The server’s IPv6 address

The server’s IPv4 address

This field is exactly the same as cacheGroup and only exists for legacy compatibility reasons

The port on which this cache server listens for incoming HTTP requests

The name of the profile used by the cache server

An integer representing the boolean concept of whether or not Traffic Routers should route client traffic this cache server; one of:

0  Do not route traffic to this server
1  Route traffic to this server normally

This cache server’s status

See also:

Health Protocol

The type of this cache server; one of:

EDGE  This is an Edge-tier cache server
MID   This is a Mid-tier cache server

An object containing keys which are the ‘xml_id’s of all of the Delivery Services within the CDN

A string containing a boolean that tells whether or not Anonymized IP Addresses are blocked by this Delivery Service; one of:

“true”  Anonymized IP addresses are blocked by this Delivery Service
“false” Anonymized IP addresses are not blocked by this Delivery Service

See also:

Configure Anonymous Blocking

A string containing a boolean that tells whether or not this Delivery Service routes traffic based only on its Coverage Zone file

A string that tells when Deep Caching is used by this Delivery Service; one of:

“ALWAYS”  Deep Caching is always used by this Delivery Service
“NEVER” Deep Caching is never used by this Delivery Service

dispersion An object describing the “dispersion” - or number of caches within a single Cache Group across which the same content is spread - within the Delivery Service

limit The maximum number of caches in which the response to a single request URL will be stored

Note: If this is greater than the number of caches in the Cache Group chosen to service the request, then content will be spread across all of them. That is, it causes no problems.

shuffled A string containing a boolean that tells whether the caches chosen for content dispersion are chosen randomly or based on a consistent hash of the request URL; one of:

“false” Caches will be chosen consistently

“true” Caches will be chosen at random

domains An array of domains served by this Delivery Service

geolocationProvider The name of a provider for IP-to-geographic-location mapping services - currently the only valid value is "maxmindGeolocationService"

ip6RoutingEnabled A string containing a boolean that tells whether IPv6 traffic can be routed on this Delivery Service; one of:

“false” IPv6 traffic will not be routed by this Delivery Service

“true” IPv6 traffic will be routed by this Delivery Service

matchList An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service

pattern A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

setNumber An integral, unique identifier for the set of types to which the type field belongs

type The type of match performed using pattern to determine whether or not to use
this *Delivery Service*

**HOST_REGEXP**  Use the *Delivery Service* if pattern matches the Host: HTTP header of an HTTP request\(^1\)

**HEADER_REGEXP**  Use the *Delivery Service* if pattern matches an HTTP header (both the name and value) in an HTTP request\(^1\)

**PATH_REGEXP**  Use the *Delivery Service* if pattern matches the request path of this *Delivery Service*’s URL

**STEERING_REGEXP**  Use the *Delivery Service* if pattern matches the xml_id of one of this *Delivery Service*’s “Steering” target *Delivery Services*

**missLocation**  An object representing the default geographic coordinates to use for a client when lookup of their IP has failed in both the Coverage Zone file(s) and the IP-to-geographic-location database

  - **lat**  Geographic latitude
  - **long**  Geographic longitude

**protocol**  An object that describes how the *Delivery Service* ought to handle HTTP requests both with and without TLS encryption

  - **acceptHttps**  A string containing a boolean that tells whether HTTPS requests should be normally serviced by this *Delivery Service*; one of:
    - “false”  Refuse to service HTTPS requests
    - “true”  Service HTTPS requests normally

  - **redirectToHttps**  A string containing a boolean that tells whether HTTP requests ought to be re-directed to use HTTPS; one of:
    - “false”  Do not redirect unencrypted traffic; service it normally
    - “true”  Respond to HTTP requests with instructions to use HTTPS instead

**regionalGeoBlocking**  A string containing a boolean that tells whether Regional Geographic Blocking is enabled on this

\(^1\) These only apply to HTTP-routed *Delivery Services*
Delivery Service; one of:

“false” Regional Geographic Blocking is not used by this Delivery Service

“true” Regional Geographic Blocking is used by this Delivery Service

See also:

Configure Regional Geo-blocking (RGB)

routingName The highest-level part of the FQDNs serviced by this Delivery Service

soa An object defining the Start of Authority (SOA) record for the Delivery Service’s TLDs (defined in domains)

  admin The name of the administrator for this zone - i.e. the RNAME

  Note: This rarely represents a proper email address, unfortunately.

expire A string containing an integer that sets the number of seconds after which secondary name servers should stop answering requests for this zone if the master does not respond

minimum A string containing an integer that sets the Time To Live (TTL) - in seconds - of the record for the purpose of negative caching

refresh A string containing an integer that sets the number of seconds after which secondary name servers should query the master for the SOA record, to detect zone changes

retry A string containing an integer that sets the number of seconds after which secondary name servers should retry to request the serial number from the master if the master does not respond

  Note: RFC 1035 dictates that this should always be less than refresh.

See also:

The Wikipedia page on Start of Authority records.
sslEnabled  A string containing a boolean that tells whether this Delivery Service uses SSL; one of:

“false”  SSL is not used by this Delivery Service

“true”  SSL is used by this Delivery Service

ttls  An object that contains keys which are types of DNS records that have values which are strings containing integers that specify the time for which a response to the specific type of record request should remain valid

Note:  This overrides config.ttls.

dgeLocations  An object containing keys which are the names of Edge-Tier Cache Groups within the CDN

backupLocations  An object that describes fallbacks for when this Cache Group is unavailable

fallbackToClosest  A string containing a boolean which tells whether requests should fall back on the closest available Cache Group when this Cache Group is not available; one of:

“false”  Do not fall back on the closest available Cache Group

“true”  Fall back on the closest available Cache Group

list  If any fallback Cache Groups have been configured for this Cache Group, this key will appear and will be an array of the names of all of those fallback Cache Groups, in the prescribed order

latitude  The geographic latitude of this Cache Group

localizationMethods  An array of short names for localization methods available for this Cache Group

longitude  The geographic longitude of this Cache Group

monitors  An object containing keys which are the (short) hostnames of Traffic Monitors within this CDN

fqdn  The FQDN of this Traffic Monitor

httpsPort  The port number on which this Traffic Monitor listens for incoming HTTPS requests

ip6  This Traffic Monitor’s IPv6 address

ip  This Traffic Monitor’s IPv4 address
location  The name of the Cache Group to which this Traffic Monitor belongs

port  The port number on which this Traffic Monitor listens for incoming HTTP requests

profile  The name of the profile used by this Traffic Monitor

---

**Note:** For legacy reasons, this must always start with "RASCAL-".

---

status  The health status of this Traffic Monitor

**See also:**

*Health Protocol*

stats  An object containing metadata information regarding the CDN

  CDN_name  The name of this CDN

  date  The UNIX epoch timestamp date in the Traffic Ops server’s own timezone

  tm_host  The FQDN of the Traffic Ops server

  tm_path  A path relative to the root of the Traffic Ops server where a request may be replaced to have this snapshot overwritten by the current configured state of the CDN

  Deprecated since version 1.1: This field is still present for legacy compatibility reasons, but its contents should be ignored. Instead, make a **PUT** request to `snapshot/{{name}}`.

  tm_user  The username of the currently logged-in user

  tm_version  The full version number of the Traffic Ops server, including release number, git commit hash, and supported Enterprise Linux version

trafficRouterLocations  An object containing keys which are the names of Cache Groups within the CDN which contain Traffic Routers

backupLocations  An object that describes fallbacks for when this Cache Group is unavailable

  fallbackToClosest  A string containing a boolean which tells whether requests should fall back on the closest available Cache Group when this Cache Group is not available; one of:

  "false"  Do not fall back on the closest available Cache Group
“true” Fall back on the closest available Cache Group

**latitude** The geographic latitude of this Cache Group

**localizationMethods** An array of short names for localization methods available for this Cache Group

**longitude** The geographic longitude of this Cache Group

# 198: Response Example

```json
{
    "response": {
        "config": {
            "api.cache-control.max-age": "10",
            "certificates.polling.interval": "300000",
            "consistent.dns.routing": "true",
            "coveragezone.polling.interval": "3600000",
            "coveragezone.polling.url": "https://trafficops.infra.ciab.test:443/coverage-zone.json",
            "dnssec.dynamic.response.expiration": "300s",
            "dnssec.enabled": "false",
            "domain_name": "mycdn.ciab.test",
            "edge.dns.limit": "6",
            "edge.dns.routing": "true",
            "edge.http.limit": "6",
            "edge.http.routing": "true",
            "federationmapping.polling.interval": "60000",
            "federationmapping.polling.url": "https://$\{toHostname\}/internal/api/1.3/federations.json",
            "geolocation.polling.interval": "86400000",
            "geolocation.polling.url": "https://trafficops.infra.ciab.test:443/GeoLite2-City.mmdb.gz",
            "keystore.maintenance.interval": "300",
            "neustar.polling.interval": "86400000",
            "soa": {
```
"admin": "twelve_monkeys",
"expire": "604800",
"minimum": "30",
"refresh": "28800",
"retry": "7200"
},
"steeringmapping.polling.interval": "60000",
"ttls": {
  "A": "3600",
  "AAAA": "3600",
  "DNSKEY": "30",
  "DS": "30",
  "NS": "3600",
  "SOA": "86400"
},
"zonemanager.cache.maintenance.interval": "300",
"zonemanager.threadpool.scale": "0.50"
},
"contentServers": {
  "edge": {
    "cacheGroup": "CDN_in_a_Box_Edge",
    "fqdn": "edge.infra.ciab.test",
    "hashCount": 999,
    "hashId": "edge",
    "httpsPort": 443,
    "interfaceName": "eth0",
    "ip": "172.16.239.100",
    "ip6": "fc01:9400:1000:8::100",
    "locationId": "CDN_in_a_Box_Edge",
    "port": 80,
    "profile": "ATS_EDGE_TIER_CACHE",
    "status": "REPORTED",
    "type": "EDGE",
    "deliveryServices": {
      "demo1": [
        "edge.demo1.mycdn.ciab.test"
      ]
    },
    "routingDisabled": 0
  },
  "mid": {
    "cacheGroup": "CDN_in_a_Box_Mid",
    "fqdn": "mid.infra.ciab.test",
    "hashCount": 999,
    "hashId": "mid",
    "httpsPort": 443,
    "interfaceName": "eth0",
    "ip": "172.16.239.120",
    "ip6": "fc01:9400:1000:8::120",
    "routingDisabled": 0
  }
}
"locationId": "CDN_in_a_Box_Mid",
"port": 80,
"profile": "ATS_MID_TIER_CACHE",
"status": "REPORTED",
"type": "MID",
"routingDisabled": 0
}
},
"contentRouters": {
  "trafficrouter": {
    "api.port": "3333",
    "fqdn": "trafficrouter.infra.ciab.test",
    "httpsPort": 443,
    "ip": "172.16.239.60",
    "ip6": "fc01:9400:1000:8::60",
    "location": "CDN_in_a_Box_Edge",
    "port": 80,
    "profile": "CCR_CIAB",
    "status": "ONLINE"
  }
},
"deliveryServices": {
  "demol": {
    "anonymousBlockingEnabled": "false",
    "coverageZoneOnly": "false",
    "dispersion": {
      "limit": 1,
      "shuffled": "true"
    },
    "domains": [
      "demol.mycdn.ciab.test"
    ],
    "geolocationProvider": {
      "maxmindGeolocationService",
      "matchsets": [
        {
          "protocol": "HTTP",
          "matchlist": [
            {
              "regex": ".*\..*",
              "match-type": "HOST"
            }
          ]
        }
      ],
      "missLocation": {
        "lat": 42,
        "lon": 120
      }
    }
  }
}
"long": -88
},
"protocol": {
    "acceptHttps": "false",
    "redirectToHttps": "false"
},
"regionalGeoBlocking": "false",
"soa": {
    "admin": "traffic_ops",
    "expire": "604800",
    "minimum": "30",
    "refresh": "28800",
    "retry": "7200"
},
"sslEnabled": "false",
"ttls": {
    "A": "",
    "AAAA": "",
    "NS": "3600",
    "SOA": "86400"
},
"ip6RoutingEnabled": "true",
"routingName": "video",
"deepCachingType": "NEVER"
}
"edgeLocations": {
    "CDN_in_a_Box_Edge": {
        "latitude": 38.897663,
        "longitude": -77.036574,
        "backupLocations": {
            "fallbackToClosest": "true"
        },
        "localizationMethods": [
            "GEO",
            "CZ",
            "DEEP_CZ"
        ]
    }
}
"trafficRouterLocations": {
    "CDN_in_a_Box_Edge": {
        "latitude": 38.897663,
        "longitude": -77.036574,
        "backupLocations": {
            "fallbackToClosest": "false"
        },
        "localizationMethods": [
            "GEO"
        ]
    }
}
cdns/{{name}}/snapshot/new

**GET**

Retrieves the *pending* snapshot for a CDN, which represents the current *configuration* of the CDN, not the current *operating state* of the CDN. The contents of this snapshot are currently used by Traffic Monitor and Traffic Router.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

**Request Structure**

Table 115: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which a snapshot shall be returned</td>
</tr>
</tbody>
</table>
# 199: Request Example

```text
GET /api/1.4/cdns/CDN-in-a-Box/snapshot/new HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

## Response Structure

- **config** An object containing basic configurations on the actual CDN object
  
  - **api.cache-control.max-age** A string containing an integer which specifies the value of `max-age` in the `Cache-Control` header of some HTTP responses, likely the Traffic Router API responses.

    Deprecated since version 1.1: This field still exists for legacy compatibility reasons, but has no known use at the time of this writing.

  - **certificates.polling.interval** A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for updated SSL certificates.

  - **consistent.dns.routing** A string containing a boolean which indicates whether DNS routing will use a consistent hashing method or “round-robin”.

    - “false” The “round-robin” method will be used to define DNS routing.
    - “true” A consistent hashing method will be used to define DNS routing.

  - **coveragezone.polling.interval** A string containing an integer which specifies the interval, in seconds, on which Traffic Routers should check for a new Coverage Zone file.

  - **coveragezone.polling.url** The URL where a Coverage Zone file may be requested by Traffic Routers.

  - **dnssec.dynamic.response.expiration** A string containing a number and unit suffix that specifies the length of time for which dynamic responses to DNSSEC lookup queries should remain valid.

  - **dnssec.enabled** A string that tells whether or not the CDN uses DNSSEC; one of:

    - “false” DNSSEC is not used within this CDN.
"true" DNSSEC is used within this CDN

domain_name The Top-Level Domain Name (TLD) served by the CDN

dns.limit This field is of unknown use, and may be remnants of a legacy system

dns.routing This field is of unknown use, and may be remnants of a legacy system

http.limit This field is of unknown use, and may be remnants of a legacy system

http.routing This field is of unknown use, and may be remnants of a legacy system

federationmapping.polling.interval A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for new federation mappings

federationmapping.polling.url The URL where Traffic Control components can request federation mappings

golocation.polling.interval A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for new IP-to-geographic-location mapping databases

golocation.polling.url The URL where Traffic Control components can request IP-to-geographic-location mapping database files

keystore.maintenance.interval A string containing an integer which specifies the interval, in seconds, on which Traffic Routers should refresh their zone caches

neustar.polling.interval A string containing an integer which specifies the interval, in seconds, on which other Traffic Control components should check for new “Neustar” databases

neustar.polling.url The URL where Traffic Control components can request “Neustar” databases

soa An object defining the Start of Authority (SOA) for the CDN’s TLD (defined in domain_name)

admin The name of the administrator for this zone - i.e. the RNAME

Note: This rarely represents a proper email
address, unfortunately.

**expire** A string containing an integer that sets the number of seconds after which secondary name servers should stop answering requests for this zone if the master does not respond.

**minimum** A string containing an integer that sets the Time To Live (TTL) - in seconds - of the record for the purpose of negative caching.

**refresh** A string containing an integer that sets the number of seconds after which secondary name servers should query the master for the SOA record, to detect zone changes.

**retry** A string containing an integer that sets the number of seconds after which secondary name servers should retry to request the serial number from the master if the master does not respond.

**Note:** RFC 1035 dictates that this should always be less than **refresh**.

**See also:**

The Wikipedia page on Start of Authority records.

**steeringmapping.polling.interval** A string containing an integer which specifies the interval, in seconds, on which Traffic Control components should check for new steering mappings.

**ttls** An object that contains keys which are types of DNS records that have values which are strings containing integers that specify the time for which a response to the specific type of record request should remain valid.

**zonemanager.cache.maintenance.interval** A configuration option for the ZoneManager Java class of Traffic Router.

**zonemanager.threadpool.scale** A configuration option for the ZoneManager Java class of Traffic Router.

**contentRouters** An object containing keys which are the (short) hostnames of the Traffic Routers that serve requests for Delivery Services in this CDN.

**api.port** A string containing the port number on which the Traffic Router API is served by this Traffic Router.
**fqdn** This Traffic Router’s Fully Qualified Domain Name (FQDN)

**httpsPort** The port number on which this Traffic Router listens for incoming HTTPS requests

**ip** This Traffic Router’s IPv4 address

**ip6** This Traffic Router’s IPv6 address

**location** The name of the Cache Group to which this Traffic Router belongs

**port** The port number on which this Traffic Router listens for incoming HTTP requests

**profile** The name of the profile used by this Traffic Router

**status** The health status of this Traffic Router

See also:

*Health Protocol*

**contentServers** An object containing keys which are the (short) hostnames of the Edge-Tier *cache server* s in the CDN; the values corresponding to those keys are routing information for said servers

**cacheGroup** The name of the Cache Group to which the server belongs

**deliveryServices** An object containing keys which are the names of *Delivery Services* to which this *cache server* is assigned; the values corresponding to those keys are arrays of FQDNs that resolve to this *cache server*

---

**Note:** Only Edge-tier *cache server* s can be assigned to a Delivery SErvice, and therefore this field will only be present when type is "EDGE".

---

**fqdn** The server’s Fully Qualified Domain Name (FQDN)

**hashCount** The number of servers to be placed into a single “hash ring” in Traffic Router

**hashId** A unique string to be used as the key for hashing servers - as of version 3.0.0 of Traffic Control, this is always the same as the server’s (short) hostname and only still exists for legacy compatibility reasons

**httpsPort** The port on which the *cache server* listens for incoming HTTPS requests

**interfaceName** The name of the main network interface device used by this *cache server*
ip6  The server’s IPv6 address

ip  The server’s IPv4 address

locationId  This field is exactly the same as cacheGroup and only exists for legacy compatibility reasons

port  The port on which this cache server listens for incoming HTTP requests

profile  The name of the profile used by the cache server

routingDisabled  An integer representing the boolean concept of whether or not Traffic Routers should route client traffic this cache server; one of:

0  Do not route traffic to this server

1  Route traffic to this server normally

status  This cache server’s status

See also:

Health Protocol

type  The type of this cache server; one of:

EDGE  This is an Edge-tier cache server

MID  This is a Mid-tier cache server

deliveryServices  An object containing keys which are the ‘xml_id’s of all of the Delivery Services within the CDN

anonymousBlockingEnabled  A string containing a boolean that tells whether or not Anonymized IP Addresses are blocked by this Delivery Service; one of:

“true”  Anonymized IP addresses are blocked by this Delivery Service

“false”  Anonymized IP addresses are not blocked by this Delivery Service

See also:

Configure Anonymous Blocking

coverageZoneOnly  A string containing a boolean that tells whether or not this Delivery Service routes traffic based only on its Coverage Zone file

deepCachingType  A string that tells when Deep Caching is used by this Delivery Service; one of:

“ALWAYS”  Deep Caching is always used by this Delivery Service
Deep Caching is never used by this **Delivery Service**

**dispersion** An object describing the “dispersion” - or number of caches within a single Cache Group across which the same content is spread - within the **Delivery Service**

**limit** The maximum number of caches in which the response to a single request URL will be stored

---

**Note:** If this is greater than the number of caches in the Cache Group chosen to service the request, then content will be spread across all of them. That is, it causes no problems.

---

**shuffled** A string containing a boolean that tells whether the caches chosen for content dispersion are chosen randomly or based on a consistent hash of the request URL; one of:

- **“false”** Caches will be chosen consistently
- **“true”** Caches will be chosen at random

**domains** An array of domains served by this **Delivery Service**

**geolocationProvider** The name of a provider for IP-to-geographic-location mapping services - currently the only valid value is "maxmindGeolocationService"

**ip6RoutingEnabled** A string containing a boolean that tells whether IPv6 traffic can be routed on this **Delivery Service**; one of:

- **“false”** IPv6 traffic will not be routed by this **Delivery Service**
- **“true”** IPv6 traffic will be routed by this **Delivery Service**

**matchList** An array of methods used by Traffic Router to determine whether or not a request can be serviced by this **Delivery Service**

- **pattern** A regular expression - the use of this pattern is dependent on the **type** field (backslashes are escaped)

- **setNumber** An integral, unique identifier for the set of types to which the **type** field belongs

- **type** The type of match performed using **pattern** to determine whether or not to use
this *Delivery Service*

**HOST_REGEXP** Use the *Delivery Service* if *pattern* matches the Host: HTTP header of an HTTP request\(^1\)

**HEADER_REGEXP** Use the *Delivery Service* if *pattern* matches an HTTP header (both the name and value) in an HTTP request\(^1\)

**PATH_REGEXP** Use the *Delivery Service* if *pattern* matches the request path of this *Delivery Service*’s URL

**STEERING_REGEXP** Use the *Delivery Service* if *pattern* matches the xml_id of one of this *Delivery Service*’s “Steering” target *Delivery Services*

**missLocation** An object representing the default geographic coordinates to use for a client when lookup of their IP has failed in both the Coverage Zone file(s) and the IP-to-geographic-location database

- **lat** Geographic latitude
- **long** Geographic longitude

**protocol** An object that describes how the *Delivery Service* ought to handle HTTP requests both with and without TLS encryption

- **acceptHttps** A string containing a boolean that tells whether HTTPS requests should be normally serviced by this *Delivery Service*; one of:
  - “false” Refuse to service HTTPS requests
  - “true” Service HTTPS requests normally

- **redirectToHttps** A string containing a boolean that tells whether HTTP requests ought to be re-directed to use HTTPS; one of:
  - “false” Do not redirect unencrypted traffic; service it normally
  - “true” Respond to HTTP requests with instructions to use HTTPS instead

**regionalGeoBlocking** A string containing a boolean that tells whether Regional Geographic Blocking is enabled on this

---

\(^1\) These only apply to HTTP-routed *Delivery Services*
Delivery Service; one of:

“false” Regional Geographic Blocking is not used by this Delivery Service

“true” Regional Geographic Blocking is used by this Delivery Service

See also:

Configure Regional Geo-blocking (RGB)

routingName The highest-level part of the FQDNs serviced by this Delivery Service

soa An object defining the Start of Authority (SOA) record for the Delivery Service’s TLDs (defined in domains)

  admin The name of the administrator for this zone - i.e. the RNAME

  Note: This rarely represents a proper email address, unfortunately.

expire A string containing an integer that sets the number of seconds after which secondary name servers should stop answering requests for this zone if the master does not respond

minimum A string containing an integer that sets the Time To Live (TTL) - in seconds - of the record for the purpose of negative caching

refresh A string containing an integer that sets the number of seconds after which secondary name servers should query the master for the SOA record, to detect zone changes

retry A string containing an integer that sets the number of seconds after which secondary name servers should retry to request the serial number from the master if the master does not respond

  Note: RFC 1035 dictates that this should always be less than refresh.

See also:

The Wikipedia page on Start of Authority records.
sslEnabled  A string containing a boolean that tells whether this 'Delivery Service' uses SSL; one of:

“false”  SSL is not used by this 'Delivery Service'

“true”  SSL is used by this 'Delivery Service'

ttls  An object that contains keys which are types of DNS records that have values which are strings containing integers that specify the time for which a response to the specific type of record request should remain valid

Note:  This overrides config.ttls.

deliveryLocations  An object containing keys which are the names of Edge-Tier Cache Groups within the CDN

backupLocations  An object that describes fallbacks for when this Cache Group is unavailable

fallbackToClosest  A string containing a boolean which tells whether requests should fall back on the closest available Cache Group when this Cache Group is not available; one of:

“false”  Do not fall back on the closest available Cache Group

“true”  Fall back on the closest available Cache Group

list  If any fallback Cache Groups have been configured for this Cache Group, this key will appear and will be an array of the names of all of those fallback Cache Groups, in the prescribed order

latitude  The geographic latitude of this Cache Group

localizationMethods  An array of short names for localization methods available for this Cache Group

longitude  The geographic longitude of this Cache Group

monitors  An object containing keys which are the (short) hostnames of Traffic Monitors within this CDN

fqdn  The FQDN of this Traffic Monitor

httpsPort  The port number on which this Traffic Monitor listens for incoming HTTPS requests

ip6  This Traffic Monitor’s IPv6 address

ip  This Traffic Monitor’s IPv4 address
location The name of the Cache Group to which this Traffic Monitor belongs

port The port number on which this Traffic Monitor listens for incoming HTTP requests

profile The name of the profile used by this Traffic Monitor

---

Note: For legacy reasons, this must always start with “RASCAL-“.

status The health status of this Traffic Monitor

See also:

Health Protocol

stats An object containing metadata information regarding the CDN

CDN_name The name of this CDN

date The UNIX epoch timestamp date in the Traffic Ops server’s own timezone

tm_host The FQDN of the Traffic Ops server

tm_path A path relative to the root of the Traffic Ops server where a request may be replaced to have this snapshot overwritten by the current configured state of the CDN

Deprecated since version 1.1: This field is still present for legacy compatibility reasons, but its contents should be ignored. Instead, make a PUT request to snapshot/\{\text{name}\}.

tm_user The username of the currently logged-in user

tm_version The full version number of the Traffic Ops server, including release number, git commit hash, and supported Enterprise Linux version

trafficRouterLocations An object containing keys which are the names of Cache Groups within the CDN which contain Traffic Routers

backupLocations An object that describes fallbacks for when this Cache Group is unavailable

fallbackToClosest A string containing a boolean which tells whether requests should fall back on the closest available Cache Group when this Cache Group is not available; one of:

“false” Do not fall back on the closest available Cache Group
“true” Fall back on the closest available Cache Group

**latitude** The geographic latitude of this Cache Group

**localizationMethods** An array of short names for localization methods available for this Cache Group

**longitude** The geographic longitude of this Cache Group

# 200: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512:
X-Server-Name: traffic_ops_golang/
Date: Wed, 12 Dec 2018 21:41:48 GMT
Transfer-Encoding: chunked

{
  "response": {
    "config": {
      "api.cache-control.max-age": "10",
      "certificates.polling.interval": "300000",
      "consistent.dns.routing": "true",
      "coveragezone.polling.interval": "3600000",
      "coveragezone.polling.url": "https://trafficops.
          infra.ciab.test:443/coverage-zone.json",
      "dnssec.dynamic.response.expiration": "300s",
      "dnssec.enabled": "false",
      "domain_name": "mycdn.ciab.test",
      "edge.dns.limit": "6",
      "edge.dns.routing": "true",
      "edge.http.limit": "6",
      "edge.http.routing": "true",
      "federationmapping.polling.interval": "60000",
      "federationmapping.polling.url": "https://$..."
    }
  }
}
```
"admin": "twelve_monkeys",
"expire": "604800",
"minimum": "30",
"refresh": "28800",
"retry": "7200"
},
"steeringmapping.polling.interval": "60000",
"tls": {
  "A": "3600",
  "AAAA": "3600",
  "DNSKEY": "30",
  "DS": "30",
  "NS": "3600",
  "SOA": "86400"
},
"zonemanager.cache.maintenance.interval": "300",
"zonemanager.threadpool.scale": "0.50"
},
"contentServers": {
  "edge": {
    "cacheGroup": "CDN_in_a_Box_Edge",
    "fqdn": "edge.infra.ciab.test",
    "hashCount": 999,
    "hashId": "edge",
    "httpsPort": 443,
    "interfaceName": "eth0",
    "ip": "172.16.239.100",
    "ip6": "fc01:9400:1000:8::100",
    "locationId": "CDN_in_a_Box_Edge",
    "port": 80,
    "profile": "ATS_EDGE_TIER_CACHE",
    "status": "REPORTED",
    "type": "EDGE",
    "deliveryServices": {
      "demo1": [
        "edge.demo1.mycdn.ciab.test"
      ]
    },
    "routingDisabled": 0
  },
  "mid": {
    "cacheGroup": "CDN_in_a_Box_Mid",
    "fqdn": "mid.infra.ciab.test",
    "hashCount": 999,
    "hashId": "mid",
    "httpsPort": 443,
    "interfaceName": "eth0",
    "ip": "172.16.239.120",
    "ip6": "fc01:9400:1000:8::120",
    "routingDisabled": 0
  }
}
"locationId": "CDN_in_a_Box_Mid",
"port": 80,
"profile": "ATS_MID_TIER_CACHE",
"status": "REPORTED",
"type": "MID",
"routingDisabled": 0
}
],
"contentRouters": {
  "trafficrouter": {
    "api.port": "3333",
    "fqdn": "trafficrouter.infra.ciab.test",
    "httpsPort": 443,
    "ip": "172.16.239.60",
    "ip6": "fc01:9400:1000:8::60",
    "location": "CDN_in_a_Box_Edge",
    "port": 80,
    "profile": "CCR_CIAB",
    "status": "ONLINE"
  }
},
"deliveryServices": {
  "demol": {
    "anonymousBlockingEnabled": "false",
    "coverageZoneOnly": "false",
    "dispersion": {
      "limit": 1,
      "shuffled": "true"
    },
    "domains": [
      "demol.mycdn.ciab.test"
    ],
    "geolocationProvider":
      "maxmindGeolocationService",
    "matchsets": [
      {
        "protocol": "HTTP",
        "matchlist": [
          {
            "regex": ".*\..*",
            "match-type": "HOST"
          }
        ]
      }
    ],
    "missLocation": {
      "lat": 42,
      "lon": 111
    }
  }
}
"long": -88
},
"protocol": {
  "acceptHttps": "false",
  "redirectToHttps": "false"
},
"regionalGeoBlocking": "false",
"soa": {
  "admin": "traffic_ops",
  "expire": "604800",
  "minimum": "30",
  "refresh": "28800",
  "retry": "7200"
},
"sslEnabled": "false",
"ttls": {
  "A": ",
  "AAAA": ",
  "NS": "3600",
  "SOA": "86400"
},
"ip6RoutingEnabled": "true",
"routingName": "video",
"deepCachingType": "NEVER"
}
},
"edgeLocations": {
  "CDN_in_a_Box_Edge": {
    "latitude": 38.897663,
    "longitude": -77.036574,
    "backupLocations": {
      "fallbackToClosest": "true",
      "list": ["test"
    ]
  },
  "localizationMethods": ["GEO", "CZ", "DEEP_CZ"
  ]
}
},
"trafficRouterLocations": {
  "CDN_in_a_Box_Edge": {
    "latitude": 38.897663,
    "longitude": -77.036574,
    "backupLocations": {
      "fallbackToClosest": "false"
})

"localizationMethods": [
    "GEO",
    "CZ",
    "DEEP_CZ"
]
}

"monitors": {
    "trafficmonitor": {
        "fqdn": "trafficmonitor.infra.ciab.test",
        "httpsPort": 443,
        "ip": "172.16.239.40",
        "ip6": "fc01:9400:1000:8::40",
        "location": "CDN_in_a_Box_Edge",
        "port": 80,
        "profile": "RASCAL-Traffic_Monitor",
        "status": "ONLINE"
    }
},

"stats": {
    "CDN_name": "CDN-in-a-Box",
    "date": 1544650908,
    "tm_host": "ipcdn-cache-51.cdnlab.comcast.net:6443",
    "tm_path": "/tools/write_crconfig/CDN-in-a-Box",
    "tm_user": "admin",
    "tm_version": "traffic_ops-3.0.0-9813.8ad7bd8e.e17"
}
}

**cdns/routing**

**GET**

Retrieves the aggregate routing percentages of Cache Groups assigned to any CDN.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Object

**Request Structure**

No parameters available
Response Structure

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

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type,
Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 21:29:32 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 01:29:32 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 7LjytwkYrzSKM4cRIOl4OMlxApFpTWJaxSdSjI64XxLVzP00GRU7XeJ22YKUyQ30qb
Content-Length: 130

{
  "response": {
    "staticRoute": 0,
    "geo": 20.6251834458468,
    "err": 0,
    "fed": 0.287643087760493,
    "cz": 79.0607572644555,
    "regionalAlternate": 0,
    "dsr": 0,
    "miss": 0.0264162019371881,
    "regionalDenied": 0
  }
}
```

cdns/usage/overview

New in version 1.2.
GET

Retrieves the high-level CDN usage metrics from Traffic Stats

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

No parameters available.

Response Structure

- **currentGbps**  The current throughput of all CDNs, in Gigabits per second
- **maxGbps**  The all-time maximum throughput of all CDNs, in Gigabits per second
- **source**  The name of the service providing the statistics. This will almost always be “TrafficStats”
- **tps**  The number of transactions being performed per second
- **version**  The version of the service providing the statistics (named in "source")

**Warning:** The "tps" field is currently broken, and will return 0 every time. See GitHub issue #1020 for more information.

# 202: Response Example

```
{ 
    "response": { 
        "currentGbps": "975.920621333333", 
        "source": "TrafficStats", 
        "tps": 0, 
        "version": "1.2", 
        "maxGbps": 12085 
    } 
}
```

consistenthash

Test Pattern-Based Consistent Hashing for a Delivery Service using a regular expression and a request path
POST

Queries database for an active Traffic Router on a given CDN and sends GET request to get the resulting path to consistent hash with a given regex and request path.

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

regex  The regex to apply to the request path to get a resulting path that will be used for consistent hashing
requestPath  The request path to use to test the regular expression against
cdnId  The unique identifier of a CDN that will be used to query for an active Traffic Router

# 203: Request Example

```
POST /api/1.4/consistenthash HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.54.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 80
Content-Type: application/x-www-form-urlencoded

{"regex": "/.*?(/.*?/).*?(m3u8)\","requestPath": "/test/path/asset.m3u8 \="","cdnId":2}
```

Response Structure

resultingPathToConsistentHash  The resulting path that Traffic Router will use for consistent hashing
consistentHashRegex  The regex used by Traffic Router derived from POST ‘regex’ parameter
requestPath  The request path used by Traffic Router to test regex against

# 204: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
```
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: QMDFO7ufgH4TcZ4YnUqyxnXDier0YiUAlfwBGDC7ySjw9uASBGsLQW351pnKp4as0vY1HuSSGpe4X-Server-Name: traffic_ops_golang/
Date: Tue, 12 Feb 2019 21:32:05 GMT
Content-Length: 142

{ "response": {
    "resultingPathToConsistentHash": "/path/m3u8",
    "consistentHashRegex": "/.*?(/.*?)/.*?(m3u8)",
    "requestPath": "/test/path/asset.m3u8"
}}

**coordinates**

New in version 1.3.

**GET**

Gets a list of all coordinates in the Traffic Ops database

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>no</td>
<td>Return only coordinates that have this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Return only coordinates with this name</td>
</tr>
</tbody>
</table>

**Response Structure**

- **id** Integral, unique, identifier for this coordinate pair
- **lastUpdated** The time and date at which this entry was last updated, in a ctime-like format
- **latitude** Latitude of the coordinate
longitude Longitude of the coordinate

name The name of the coordinate - typically this just reflects the name of the Cache Group for which the coordinate was created

# 205: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Y2vxC3hpxIg6aNBBT7i2hbAViIJP+dJoqHlzu3acFM+vGay/
I5E+e2YOC9RY8hcJPrKKNysZOD8Dob9KsFgaw==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:32:28 GMT
Content-Length: 942

{
    "response": [
        {
            "id": 1,
            "name": "from_cachegroup_TRAFFIC_ANALYTICS",
            "latitude": 38.897663,
            "longitude": -77.036574,
            "lastUpdated": "2018-10-24 16:07:04+00"
        },
        {
            "id": 2,
            "name": "from_cachegroup_TRAFFIC_OPS",
            "latitude": 38.897663,
            "longitude": -77.036574,
            "lastUpdated": "2018-10-24 16:07:04+00"
        },
        {
            "id": 3,
            "name": "from_cachegroup_TRAFFIC_OPS_DB",
            "latitude": 38.897663,
            "longitude": -77.036574,
            "lastUpdated": "2018-10-24 16:07:04+00"
        },
        {
            "id": 4,
            "name": "from_cachegroup_TRAFFIC_PORTAL",
            "latitude": 38.897663,
            "longitude": -77.036574,
            "lastUpdated": "2018-10-24 16:07:04+00"
        }
    ]
}
```
POST

Creates a new coordinate pair

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object

Request Structure

name The name of the new coordinate

latitude The desired latitude of the new coordinate (must be on the interval [-180, 180])

longitude The desired longitude of the new coordinate (must be on the interval [-90, 90])

# 206: Request Example

```
POST /api/1.4/coordinates HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```
Response Structure

id  Integral, unique, identifier for the newly created coordinate pair

lastUpdated  The time and date at which this entry was last updated, in a ctime-like format

latitude  Latitude of the newly created coordinate

longitude  Longitude of the newly created coordinate

name  The name of the coordinate

# 207: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 7pWdeZyIIXE1P7o/JVon+5eSCbDw+FGamAzdXzWHXJ81hF+Vh+/tWFckzHYw3rP2kBvW2u+gqlffjQpBCMjt7A==
X-Server-Name: traffic_ops_golang/
Date: Thu, 15 Nov 2018 17:48:55 GMT
Content-Length: 165

{ "alerts": [ 
    {
      "text": "coordinate was created.",
      "level": "success"
    }
  ],
  "response": {
    "id": 9,
    "name": "test",
    "latitude": 0,
    "longitude": 0,
    "lastUpdated": "2018-11-15 17:48:55+00"
  }
}
```
PUT

Updates a coordinate

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

### Request Structure

Table 117: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the coordinate to edit</td>
</tr>
</tbody>
</table>

**name** The name of the new coordinate

**latitude** The desired new latitude of the coordinate (must be on the interval [-180, 180])

**longitude** The desired new longitude of the coordinate (must be on the interval [-90, 90])

### # 208: Request Example

```http
PUT /api/1.4/coordinates?id=9 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 48
Content-Type: application/json

{"name": "quest", "latitude": 0, "longitude": 0}
```

### Response Structure

**id** Integral, unique, identifier for the coordinate pair

**lastUpdated** The time and date at which this entry was last updated, in a `ctime`-like format

**latitude** Latitude of the coordinate
longitude Longitude of the coordinate
name The name of the coordinate

# 209: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512:
zd03Uvbnv8EbSZ275Xp5tnnYStZsZTdyPxXnoqK4Q25WhELLPL8iH1Rf0aiLTbrUWUeJ8ue2HRz6aB+iXCCGA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 15 Nov 2018 17:54:30 GMT
Content-Length: 166

{
  "alerts": [
    {
      "text": "coordinate was updated.",
      "level": "success"
    }
  ],
  "response": {
    "id": 9,
    "name": "quest",
    "latitude": 0,
    "longitude": 0,
    "lastUpdated": "2018-11-15 17:54:30+00"
  }
}
```

DELETE

Deletes a coordinate

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type undefined
Request Structure

Table 118: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the coordinate to delete</td>
</tr>
</tbody>
</table>

Response Structure

# 210: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 82x/Wdckqgk4LN5LI1ZfBJ26xxDrUVUGDjs5QFa/\nLzap7dU3OZkjv8XW41xeFYj8PdmxHlpb7hiVObvLaxnEDA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 15 Nov 2018 17:57:42 GMT
Content-Length: 65

{ "alerts": [
    {
        "text": "coordinate was deleted.",
        "level": "success"
    }
]
```

deliveryservice_server/{{DSID}}/{{serverID}}

DELETE

Removes a cache server from a Delivery Service.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”\(^1\)

**Response Type** undefined

\(^1\) Users with the “admin” or “operations” roles will be able to delete any Delivery Service, whereas other users will only be able to delete Delivery Services that their tenant has permissions to delete.
Request Structure

Table 119: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsId</td>
<td>yes</td>
<td>An integral, unique identifier for a Delivery Service</td>
</tr>
<tr>
<td>serverID</td>
<td>yes</td>
<td>An integral, unique identifier for a server</td>
</tr>
</tbody>
</table>

Note: The server identified by serverID must be a cache server, or the assignment will fail.

Response Structure

# 211: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: rGD2sOMHYF0sga1zuTytyLHCUkkc3ZwQRKvZ/\n\nHuPzObOP4WztKTOVXB4uhs3iJqB9g9zRB2TucMxONHN+3/yShQ==
X-Server-Name: traffic_ops_golang/
Date: Thu, 01 Nov 2018 14:24:34 GMT
Content-Length: 80

{ "alerts": [ 
    { 
      "text": "Server unlinked from delivery service.",
      "level": "success"
    }
]}
```

deliveryservice_stats

Caution: This page is a stub! Much of it may be missing or just downright wrong - it needs a lot of love from people with the domain knowledge required to update it.

New in version 1.2.
**Warning:** This endpoint does NOT respect tenancy permissions! The bug is tracked by GitHub Issue #3187.

**GET**

Retrieves time-aggregated statistics on a specific *Delivery Service*.

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Object
### Request Structure

Table 120: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deliveryServiceName</td>
<td>yes</td>
<td>The name of the Delivery Service for which statistics will be aggregated</td>
</tr>
<tr>
<td>metricType</td>
<td>yes</td>
<td>The metric type being reported - one of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>kbps</strong> The total traffic rate in kilobytes per second served by the Delivery Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>out_bytes</strong> The total number of bytes sent out to clients through the Delivery Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>status_4xx</strong> The amount of requests that were serviced with 400-499 HTTP status codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>status_5xx</strong> The amount of requests that were serviced with 500-599 HTTP status codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>tps_total</strong> The total traffic rate in transactions per second served by the Delivery Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>tps_2xx</strong> The total traffic rate in transactions per second served with 200-299 HTTP status codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>tps_3xx</strong> The total traffic rate in transactions per second served with 300-399 HTTP status codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>tps_4xx</strong> The total traffic rate in transactions per second served with 400-499 HTTP status codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>tps_5xx</strong> The total traffic rate in transactions per second served with 500-599 HTTP status codes</td>
</tr>
<tr>
<td>startDate</td>
<td>yes</td>
<td>The date and time from which statistics shall be aggregated in ISO8601 format, e.g. 2018-08-11T12:30:00-07:00</td>
</tr>
<tr>
<td>endDate</td>
<td>yes</td>
<td>The date and time until which statistics shall be aggregated in ISO8601 format, e.g. 2018-08-12T12:30:00-07:00</td>
</tr>
</tbody>
</table>
### Response Structure

#### Table 121: Response Keys

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>string</td>
<td>The source of the data</td>
</tr>
<tr>
<td>summary</td>
<td>hash</td>
<td>Summary data</td>
</tr>
<tr>
<td>&gt;totalBytes</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;count</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>&gt;min</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;max</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;fifthPercentile</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;ninetyEighthPercentile</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;ninetyFifthPercentile</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;average</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>&gt;totalTransactions</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>series</td>
<td>hash</td>
<td>Series data</td>
</tr>
<tr>
<td>&gt;count</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>&gt;columns</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td>&gt;name</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>&gt;values</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;time</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;value</td>
<td>float</td>
<td></td>
</tr>
</tbody>
</table>

#### # 212: Response Example

```json
{  
  "response": {  
    "source": "TrafficStats",  
    "summary": {  
      "average": 1081172.785,  
      "count": 28,  
      "fifthPercentile": 888827.26,  
      "max": 1326680.31,  
      "min": 888827.26,  
      "ninetyEighthPercentile": 1324785.47,  
      "ninetyFifthPercentile": 1324785.47,  
      "totalBytes": 37841047.475,  
      "totalTransactions": 1020202030101  
    },  
    "series": {  
      "columns": [  
        "time",  
        ""  
      ],  
      "count": 60,  
      "name": "kbps",  
      "tags": {  
        "cachegroup": "total"  
      }  
    }  
  }
```

(continues on next page)
deliveryservice_user

POST

Assigns one or more Delivery Services to a user.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object
Request Structure

**userId** An integral, unique identifier for the user to whom the *Delivery Service*(s) identified in `deliveryServices` will be assigned

**deliveryServices** An array of integral, unique identifiers for the *Delivery Service*(s) being assigned to the user identified by `userId`

**replace** An optional field which, when present and `true` will replace existing user/ds assignments? (true|false)

# 213: Request Example

```
POST /api/1.4/deliveryservice_user HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 38
Content-Type: application/json

{"userId": 5, "deliveryServices": [1]}
```

Response Structure

**userId** The integral, unique identifier of the user to whom the *Delivery Service*(s) identified in `deliveryServices` are assigned

**deliveryServices** An array of integral, unique identifiers of *Delivery Service*s assigned to the user identified by `userId`

**replace** If `true`, any and all existing, conflicting *Delivery Service* assignments were overwritten by this assignment operation

# 214: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 14 Nov 2018 21:37:30 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 01:37:30 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: ...
```

(continues on next page)
Content-Length: 127

```json
{
  "alerts": [
    {
      "level": "success",
      "text": "Delivery service assignments complete."
    }
  ],
  "response": {
    "userId": 5,
    "deliveryServices": [
      1
    ]
  }
}
```

deliveryservice_user/{{dsID}}/{{userID}}

DELETE

Removes a Delivery Service from a user.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type undefined

Request Structure

Table 122: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsId</td>
<td>An integral, unique identifier for the Delivery Service which should no longer be assigned to the user identified by userId</td>
</tr>
<tr>
<td>userId</td>
<td>An integral, unique identifier for the user to whom the Delivery Service identified by dsId should no longer be assigned</td>
</tr>
</tbody>
</table>

Response Structure

# 215: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
```
deliveryservices

GET

Retrieves all Delivery Services

Auth. Required  Yes

Roles Required  None¹

Response Type  Array

¹ Users with the roles “admin” and/or “operations” will be able to see all Delivery Services, whereas any other user will only see the Delivery Services their Tenant is allowed to see.
Request Structure

Table 123: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn</td>
<td>no</td>
<td>Show only the Delivery Services belonging to the CDN identified by this integral, unique identifier</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Show only the Delivery Service that has this integral, unique identifier</td>
</tr>
<tr>
<td>logsEnabled</td>
<td>no</td>
<td>If true, return only Delivery Services with logging enabled, otherwise return only Delivery Services with logging disabled</td>
</tr>
<tr>
<td>profile</td>
<td>no</td>
<td>Return only Delivery Services using the Profile identified by this integral, unique identifier</td>
</tr>
<tr>
<td>tenant</td>
<td>no</td>
<td>Show only the Delivery Services belonging to the Tenant identified by this integral, unique identifier</td>
</tr>
<tr>
<td>type</td>
<td>no</td>
<td>Return only Delivery Services of the Delivery Service type identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>

Response Structure

- **active** true if the Delivery Service is active, false otherwise
- **anonymousBlockingEnabled** true if Anonymous Blocking has been configured for the Delivery Service, false otherwise
- **cacheurl** A setting for a deprecated feature of now-unsupported Trafficserver versions
  Deprecation since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later
- **ccrDnsTtl** The Time To Live (TTL) of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons
- **cdnId** The integral, unique identifier of the CDN to which the Delivery Service belongs
- **cdnName** Name of the CDN to which the Delivery Service belongs
- **checkPath** The path portion of the URL to check connections to this Delivery Service’s origin server
- **consistentHashRegex** If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services
- **displayName** The display name of the Delivery Service
- **dnsBypassCname** Domain name to overflow requests for HTTP Delivery Services - bypass starts when the traffic on this Delivery Service exceeds
globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp The IPv4 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp6 The IPv6 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassTtl The time for which a DNS bypass of this Delivery Services shall remain active

dscp The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

dnsBypassIp The IPv4 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dscp The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

dscp The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

Warning: This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

dscp The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

dscp The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

geoLimit The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

0 None - no limitations
1 Only route when the client’s IP is found in the Coverage Zone File (CZF)
2 Only route when the client’s IP is found in the CZF, or when the client can be determined to be from the United States of America

geoLimitCountries A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this Delivery Service

geoLimitRedirectUrl A URL to which clients blocked by Regional Geographic Blocking or the geoLimit settings will be re-directed

gEO PROVIDER An integer that represents the provider of a database for mapping IPs to geographic locations; currently only the following values are supported:

0 The “Maxmind” GeoIP2 database (default)

This only applies to DNS-routed Delivery Services
Neustar

globalMaxMbps The maximum global bandwidth allowed on this Delivery Service. If exceeded, traffic will be routed to dnsBypassIp (or dnsBypassIp6 for IPv6 traffic) for DNS Delivery Services and to httpBypassFqdn for HTTP Delivery Services.

globalMaxTps The maximum global transactions per second allowed on this Delivery Service. When this is exceeded traffic will be sent to the dnsBypassIp (and/or dnsBypassIp6) for DNS Delivery Services and to the httpBypassFqdn for HTTP Delivery Services.

httpBypassFqdn The HTTP destination to use for bypass on an HTTP Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service.

id An integral, unique identifier for this Delivery Service.

infoUrl This is a string which is expected to contain at least one URL pointing to more information about the Delivery Service. Historically, this has been used to link relevant JIRA tickets.

initialDispersion The number of caches between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1.

ipv6RoutingEnabled If true, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache; if false all addresses will be IPv4, regardless of the client connection.

lastUpdated The date and time at which this Delivery Service was last updated, in a ctime-like format.

logsEnabled If true, logging is enabled for this Delivery Service, otherwise it is disabled.

longDesc A description of the Delivery Service.

longDesc1 A field used when more detailed information that that provided by longDesc is desired.

longDesc2 A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired.

matchList An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service.

    pattern A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped).

    setNumber An integral, unique identifier for the set of types to which the type field belongs.

---

2 This only applies to HTTP-routed Delivery Services.
**type** The type of match performed using `pattern` to determine whether or not to use this `Delivery Service`

**HOST_REGEXP** Use the `Delivery Service` if `pattern` matches the `Host` HTTP header of an HTTP request

**HEADER_REGEXP** Use the `Delivery Service` if `pattern` matches an HTTP header (both the name and value) in an HTTP request

**PATH_REGEXP** Use the `Delivery Service` if `pattern` matches the request path of this `Delivery Service`'s URL

**STEERING_REGEXP** Use the `Delivery Service` if `pattern` matches the `xml_id` of one of this `Delivery Service`'s “Steering” target `Delivery Services`

**maxDnsAnswers** The maximum number of IPs to put in responses to A/AAAA DNS record requests (0 means all available)

**maxOriginConnections** The maximum number of connections allowed to the origin (0 means no maximum).

New in version 1.4.

**midHeaderRewrite** Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

**missLat** The latitude to use when the client cannot be found in the CZF or a geographic IP lookup

**missLong** The longitude to use when the client cannot be found in the CZF or a geographic IP lookup

**multiSiteOrigin** `true` if the Multi Site Origin feature is enabled for this `Delivery Service`, `false` otherwise

**orgServerFqdn** The URL of the `Delivery Service`’s origin server for use in retrieving content from the origin server

**Note:** Despite the field name, this must truly be a full URL - including the protocol (e.g. `http://` or `https://`) - NOT merely the server’s Fully Qualified Domain Name (FQDN)

**originShield** An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This field is a string of FQDNs to use as origin shields, delimited by |
profileDescription The description of the Traffic Router Profile with which this Delivery Service is associated

profileId The integral, unique identifier for the Traffic Router profile with which this Delivery Service is associated

profileName The name of the Traffic Router Profile with which this Delivery Service is associated

protocol The protocol which clients will use to communicate with Edge-tier cache server s2 - this is an integer on the interval [0-2] where the values have these meanings:

0 HTTP
1 HTTPS
2 Both HTTP and HTTPS

qstringIgnore Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where the values have these meanings:

0 URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin
1 URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin
2 Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin

rangeRequestHandling Tells caches how to handle range requests7 - this is an integer on the interval [0,2] where the values have these meanings:

0 Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache
1 Use the background_fetch plugin to service the range request while caching the whole object
2 Use the experimental cache_range_requests plugin to treat unique ranges as unique objects

regexRemap A regular expression remap rule to apply to this Delivery Service at the Edge tier

See also:

The Apache Trafficserver documentation for the Regex Remap plugin

---

These fields are required for HTTP-routed and DNS-routed Delivery Services, but are optional for (and in fact may have no effect on) STEERING and ANY_MAP Delivery Services
regionalGeoBlocking  true if Regional Geo Blocking is in use within this Delivery Service, false otherwise - see Configure Regional Geo-blocking (RGB) for more information

remapText  Additional, raw text to add to the remap line for caches

See also:
The Apache Trafficserver documentation for the Regex Remap plugin

signed  true if token-based authentication is enabled for this Delivery Service, false otherwise

signingAlgorithm  Type of URL signing method to sign the URLs, basically comes down to one of two plugins or null:

null  Token-based authentication is not enabled for this Delivery Service

url_sig:  URL Signing token-based authentication is enabled for this Delivery Service

uri_signing  URI Signing token-based authentication is enabled for this Delivery Service

See also:
The Apache Trafficserver documentation for the url_sig plugin and the draft RFC for uri_signing - note, however that the current implementation of uri_signing uses Draft 12 of that RFC document, NOT the latest

sslKeyVersion  This integer indicates the generation of keys in use by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated

Warning:  This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

tenantId  The integral, unique identifier of the tenant who owns this Delivery Service

trRequestHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)^2

trResponseHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)^2

type  The name of the routing type of this Delivery Service e.g. “HTTP”
typeId  The integral, unique identifier of the routing type of this Delivery Service

xmlId  A unique string that describes this Delivery Service - exists for legacy reasons

5.1. Traffic Ops API

399
# 216: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: mCLMjvACRKHNgP/\nX-Server-Name: traffic_ops_golang/
Date: Thu, 15 Nov 2018 19:04:29 GMT
Transfer-Encoding: chunked

{
    "response": [
    {
        "active": true,
        "anonymousBlockingEnabled": false,
        "cacheurl": null,
        "ccrDnsTtl": null,
        "cdnId": 2,
        "cdnName": "CDN-in-a-Box",
        "checkPath": null,
        "displayName": "Demo 1",
        "dnsBypassCname": null,
        "dnsBypassIp": null,
        "dnsBypassIp6": null,
        "dnsBypassTtl": null,
        "dscp": 0,
        "edgeHeaderRewrite": null,
        "geoLimit": 0,
        "geoLimitCountries": null,
        "geoLimitRedirectURL": null,
        "geoProvider": 0,
        "globalMaxMbps": null,
        "globalMaxTps": null,
        "httpBypassFqdn": null,
        "id": 1,
        "infoUrl": null,
        "initialDispersion": 1,
        "ipv6RoutingEnabled": true,
        "lastUpdated": "2018-11-14 18:21:17+00",
        "logsEnabled": true,
        "longDesc": "Apachecon North America 2018",
        "longDesc1": null,
        "longDesc2": null,
        "matchList": [ ]
    }
]
POST

Allows users to create Delivery Service.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Array
Request Structure

- active: If true, the Delivery Service will immediately become active and serves traffic.
- anonymousBlockingEnabled: An optional field which, if defined and true, will cause Anonymous Blocking to be used with the new Delivery Service.
- cacheurl: An optional setting for a deprecated feature of now-unsupported Traffic-server versions. (read: “Don’t use this”)
  
  Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later.
- ccrDnsTtl: The Time To Live (TTL) in seconds of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router. Named “ccrDnsTtl” for legacy reasons.
- cdnId: The integral, unique identifier for the CDN to which this Delivery Service shall be assigned.
- checkPath: The path portion of the URL which will be used to check connections to this Delivery Service’s origin server.
- consistentHashRegex: If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services.
- deepCachingType: A string describing when to do Deep Caching for this Delivery Service:
  - NEVER: Deep Caching will never be used by this Delivery Service (default).
  - ALWAYS: Deep Caching will always be used by this Delivery Service.
- displayName: The human-friendly name for this Delivery Service.
- dnsBypassCname: Domain name to overflow requests for HTTP Delivery Services - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service.
- dnsBypassIp: The IPv4 IP to use for bypass on a DNS Delivery Service. - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service.
- dnsBypassIp6: The IPv6 IP to use for bypass on a DNS Delivery Service. - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service.
- dnsBypassTtl: The time for which a DNS bypass of this Delivery Service shall remain active.
**dscp**  The Differentiated Services Code Point (DSCP) with which to mark downstream (EDGE -> customer) traffic. This should be zero in most cases.

**edgeHeaderRewrite**  An optional string which, if present, defines rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin.

**fqPacingRate**  An optional integer which, if present, sets the Fair-Queuing Pacing Rate in bytes per second set on the all TCP connection sockets in the *Delivery Service* (see man `tc-vc_codel` for more information) - Linux only, defaults to 0 meaning “disabled”.

**geoLimit**  The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

- **0**  None - no limitations
- **1**  Only route when the client’s IP is found in the Coverage Zone File (CZF)
- **2**  Only route when the client’s IP is found in the CZF, or when the client can be determined to be from the United States of America

**Warning:**  This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

**geoLimitCountries**  A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this *Delivery Service*.

**geoLimitRedirectUrl**  A URL to which clients blocked by *Regional Geographic Blocking* or the *geoLimit* settings will be re-directed.

**geoProvider**  An integer that represents the provider of a database for mapping IPs to geographic locations; currently only the following values are supported:

- **0**  The “Maxmind” GeoIP2 database (default)
- **1**  Neustar

**globalMaxMbps**  An optional integer that will set the maximum global bandwidth allowed on this *Delivery Service*. If exceeded, traffic will be routed to `dnsBypassIp` (or `dnsBypassIp6` for IPv6 traffic) for DNS *Delivery Services* and to `httpBypassFqdn` for HTTP *Delivery Services*.

**globalMaxTps**  An optional integer that will set the maximum global transactions per second allowed on this *Delivery Service*. When this is exceeded traffic will be sent to the `dnsBpassIp` (and/or `dnsBypassIp6`) for DNS *Delivery Services* and to the `httpBypassFqdn` for HTTP *Delivery Services*.

**httpBypassFqdn**  An optional Fully Qualified Domain Name (FQDN) to use for bypass on an HTTP *Delivery Service* - bypass starts when the traffic

---

5 These fields must be defined if and only if `geoLimit` is non-zero.
on this Delivery Service exceeds globalMaxMbps, or when more than
globalMaxTps is being exceeded within the Delivery Service.\(^2\)

infoUrl An optional string which, if present, is expected to contain at least one
URL pointing to more information about the Delivery Service. Historically,
this has been used to link relevant JIRA tickets

initialDispersion The number of caches between which traffic requesting the
same object will be randomly split - meaning that if 4 clients all request the
same object (one after another), then if this is above 4 there is a possibility
that all 4 are cache misses. For most use-cases, this should be 1\(^2\)

ipv6RoutingEnabled If true, clients that connect to Traffic Router using IPv6
will be given the IPv6 address of a suitable Edge-tier cache; if false all
addresses will be IPv4, regardless of the client connection - optional for
ANY_MAP Delivery Services

logsEnabled If true, logging is enabled for this Delivery Service, otherwise it is
disabled

longDesc An optional description of the Delivery Service

longDesc1 An optional field used when more detailed information that that pro-
vided by longDesc is desired

longDesc2 An optional field used when even more detailed information that that
provided by either longDesc or longDesc1 is desired

maxDnsAnswers An optional field which, when present, specifies the maximum
number of IPs to put in responses to A/AAAA DNS record requests - defaults
to 0, meaning “no limit”\(^4\)

maxOriginConnections The maximum number of connections allowed to the ori-
gin (0 means no maximum).

New in version 1.4.

midHeaderRewrite An optional string containing rewrite operations to be per-
formed on TCP headers at the Edge-tier cache level - used by the Header
Rewrite Apache Trafficserver plugin

missLat The latitude to use when the client cannot be found in the CZF or a geo-
graphic IP lookup\(^7\)

missLong The longitude to use when the client cannot be found in the CZF or a geo-
graphic IP lookup\(^7\)

multiSiteOrigin true if the Multi Site Origin feature is enabled for this Delivery
Service, false otherwise\(^3\)

orgServerFqdn The URL of the Delivery Service’s origin server for use in re-
trieving content from the origin server\(^7\)

\(^6\) These fields are required for HTTP-routed Delivery Services, and optional for all others
Note: Despite the field name, this must truly be a full URL - including the protocol (e.g. http:// or https://) - NOT merely the server’s Fully Qualified Domain Name (FQDN)

originShield An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This optional field is a string of FQDNs to use as origin shields, delimited by |

profileId An optional, integral, unique identifier for the Traffic Router profile with which this Delivery Service shall be associated

protocol The protocol which clients will use to communicate with Edge-tier cache server s - this is an (optional for ANY_MAP Delivery Services) integer on the interval [0,2] where the values have these meanings:

  0 HTTP
  1 HTTPS
  2 Both HTTP and HTTPS

qstringIgnore Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where the values have these meanings:

  0 URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin
  1 URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin
  2 Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin

rangeRequestHandling Tells caches how to handle range requests - this is an integer on the interval [0,2] where the values have these meanings:

  0 Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache
  1 Use the background_fetch plugin to service the range request while caching the whole object
  2 Use the experimental cache_range_requests plugin to treat unique ranges as unique objects

regexRemap An optional, regular expression remap rule to apply to this Delivery Service at the Edge tier

See also:
The Apache Trafficserver documentation for the Regex Remap plugin

**regionalGeoBlocking**  
*true* if Regional Geo Blocking is in use within this *Delivery Service*, *false* otherwise - see **Configure Regional Geo-blocking (RGB)** for more information

**remapText**  
Optional, raw text to add to the remap line for caches

See also:

The Apache Trafficserver documentation for the Regex Remap plugin

**routingName**  
The routing name of this *Delivery Service*, used as the top-level part of the FQDN used by clients to request content from the *Delivery Service* e.g. `routingName.xml_id.CDNName.com`

**signed**  
An optional field which should be *true* if token-based authentication will be enabled for this *Delivery Service*, *false* (default) otherwise

**signingAlgorithm**  
Type of URL signing method to sign the URLs, basically comes down to one of two plugins or **null**:  

- **null**  
  Token-based authentication is not enabled for this *Delivery Service*

- **url_sig**  
  URL Signing token-based authentication is enabled for this *Delivery Service*

- **uri_signing**  
  URI Signing token-based authentication is enabled for this *Delivery Service*

See also:

The Apache Trafficserver documentation for the **url_sig** plugin and the draft RFC for **uri_signing** - note, however that the current implementation of uri_signing uses Draft 12 of that RFC document, **NOT** the latest

**sslKeyVersion**  
This optional integer indicates the generation of keys to be used by the *Delivery Service* - if any - and is incremented by the Traffic Portal client whenever new keys are generated

**tenantId**  
An optional, integral, unique identifier of the tenant who will own this *Delivery Service*

**trRequestHeaders**  
If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where `__RETURN__` translates to a carriage return and line feed (`\r\n`)

**trResponseHeaders**  
If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it’s a template where `__RETURN__` translates to a carriage return and line feed (`\r\n`)

**typeId**  
The integral, unique identifier for the routing type of this *Delivery Service*
xmlId A unique string that describes this Delivery Service - exists for legacy reasons

Note: This should almost never be different from the Delivery Service's displayName

# 217: Request Example

```json
POST /api/1.4/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 761
Content-Type: application/json

{
    "active": false,
    "anonymousBlockingEnabled": false,
    "cdnId": 2,
    "cdnName": "CDN-in-a-Box",
    "deepCachingType": "NEVER",
    "displayName": "test",
    "exampleURLs": [
        "http://test.test.mycdn.ciab.test"
    ],
    "dscp": 0,
    "geoLimit": 0,
    "geoProvider": 0,
    "initialDispersion": 1,
    "ipv6RoutingEnabled": false,
    "lastUpdated": "2018-11-14 18:21:17+00",
    "logsEnabled": true,
    "longDesc": "A :term:`Delivery Service` created expressly for API documentation examples",
    "missLat": -1,
    "missLong": -1,
    "maxOriginConnections": 0,
    "multiSiteOrigin": false,
    "orgServerFqdn": "http://origin.infra.ciab.test",
    "protocol": 0,
    "queryStringIgnore": 0,
    "rangeRequestHandling": 0,
    "regionalGeoBlocking": false,
    "routingName": "test",
    "signed": false,
    "tenant": "root",
    "tenantId": 1,
    "typeId": 1,
}
```

(continues on next page)
Response Structure

**active**  true if the *Delivery Service* is active, false otherwise

**anonymousBlockingEnabled**  true if *Anonymous Blocking* has been configured for the *Delivery Service*, false otherwise

**cacheurl**  A setting for a deprecated feature of now-unsupported Trafficserver versions

  Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later

**ccrDnsTtl**  The Time To Live (TTL) of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons

**cdnId**  The integral, unique identifier of the CDN to which the *Delivery Service* belongs

**cdnName**  Name of the CDN to which the *Delivery Service* belongs

**checkPath**  The path portion of the URL to check connections to this *Delivery Service*’s origin server

**consistentHashRegex**  If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services

**displayName**  The display name of the *Delivery Service*

**dnsBypassCname**  Domain name to overflow requests for HTTP *Delivery Services* - bypass starts when the traffic on this *Delivery Service* exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the *Delivery Service*.

**dnsBypassIp**  The IPv4 IP to use for bypass on a DNS *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the *Delivery Service*.

**dnsBypassIp6**  The IPv6 IP to use for bypass on a DNS *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the *Delivery Service*.

**dnsBypassTtl**  The time for which a DNS bypass of this *Delivery Service* shall remain active.
**dscp** The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

**edgeHeaderRewrite** Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

**fqPacingRate** The Fair-Queuing Pacing Rate in Bytes per second set on the all TCP connection sockets in the *Delivery Service* (see `man tc-fc_codel` for more information) - Linux only

**geoLimit** The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

- **0** None - no limitations
- **1** Only route when the client’s IP is found in the Coverage Zone File (CZF)
- **2** Only route when the client’s IP is found in the CZF, or when the client can be determined to be from the United States of America

**Warning:** This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

**geoLimitCountries** A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this *Delivery Service*

**geoLimitRedirectUrl** A URL to which clients blocked by Regional Geographic Blocking or the geoLimit settings will be re-directed

**geoProvider** An integer that represents the provider of a database for mapping IPs to geographic locations; currently only the following values are supported:

- **0** The “Maxmind” GeoIP2 database (default)
- **1** Neustar

**globalMaxMbps** The maximum global bandwidth allowed on this *Delivery Service*. If exceeded, traffic will be routed to `dnsBypassIp` (or `dnsBypassIp6` for IPv6 traffic) for DNS *Delivery Services* and to `httpBypassFqdn` for HTTP *Delivery Services*

**globalMaxTps** The maximum global transactions per second allowed on this *Delivery Service*. When this is exceeded traffic will be sent to the `dnsBypassIp` (and/or `dnsBypassIp6`) for DNS *Delivery Services* and to the `httpBypassFqdn` for HTTP *Delivery Services*

**httpBypassFqdn** The HTTP destination to use for bypass on an HTTP *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds `globalMaxMbps`, or when more than `globalMaxTps` is being exceeded within the *Delivery Service*

**id** An integral, unique identifier for this *Delivery Service*
infoUrl This is a string which is expected to contain at least one URL pointing to more information about the Delivery Service. Historically, this has been used to link relevant JIRA tickets.

initialDispersion The number of caches between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1.

ipv6RoutingEnabled If true, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache; if false all addresses will be IPv4, regardless of the client connection.

lastUpdated The date and time at which this Delivery Service was last updated, in a ctime-like format.

logsEnabled If true, logging is enabled for this Delivery Service, otherwise it is disabled.

longDesc A description of the Delivery Service.

longDesc1 A field used when more detailed information that that provided by longDesc is desired.

longDesc2 A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired.

matchList An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service.

- **pattern** A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped).

- **setNumber** An integral, unique identifier for the set of types to which the type field belongs.

- **type** The type of match performed using pattern to determine whether or not to use this Delivery Service.

  - **HOST_REGEXP** Use the Delivery Service if pattern matches the Host: HTTP header of an HTTP request.

  - **HEADER_REGEXP** Use the Delivery Service if pattern matches an HTTP header (both the name and value) in an HTTP request.

  - **PATH_REGEXP** Use the Delivery Service if pattern matches the request path of this Delivery Service's URL.

  - **STEERING_REGEXP** Use the Delivery Service if pattern matches the xml_id of one of this Delivery Service's “Steering” target Delivery Services.
**maxDnsAnswers**  The maximum number of IPs to put in responses to A/AAAA DNS record requests (0 means all available)\(^4\)

**maxOriginConnections**  The maximum number of connections allowed to the origin (0 means no maximum).

New in version 1.4.

**midHeaderRewrite**  Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

**missLat**  The latitude to use when the client cannot be found in the CZF or a geographic IP lookup

**missLong**  The longitude to use when the client cannot be found in the CZF or a geographic IP lookup

**multiSiteOrigin**  True if the Multi Site Origin feature is enabled for this Delivery Service, false otherwise\(^3\)

**orgServerFqdn**  The URL of the Delivery Service’s origin server for use in retrieving content from the origin server

---

**Note:** Despite the field name, this must truly be a full URL - including the protocol (e.g. http:// or https://) - **NOT** merely the server’s Fully Qualified Domain Name (FQDN)

---

**originShield**  An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This field is a string of FQDNs to use as origin shields, delimited by |

**profileDescription**  The description of the Traffic Router Profile with which this Delivery Service is associated

**profileId**  The integral, unique identifier for the Traffic Router profile with which this Delivery Service is associated

**profileName**  The name of the Traffic Router Profile with which this Delivery Service is associated

**protocol**  The protocol which clients will use to communicate with Edge-tier cache server s\(^2\) - this is an integer on the interval [0-2] where the values have these meanings:

0  HTTP
1  HTTPS
2  Both HTTP and HTTPS

**qstringIgnore**  Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where
the values have these meanings:

0 URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin.

1 URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin.

2 Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin.

**rangeRequestHandling** Tells caches how to handle range requests - this is an integer on the interval [0,2] where the values have these meanings:

0 Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache.

1 Use the `background_fetch` plugin to service the range request while caching the whole object.

2 Use the `experimental cache_range_requests` plugin to treat unique ranges as unique objects.

**regexRemap** A regular expression remap rule to apply to this Delivery Service at the Edge tier.

See also:

The Apache Trafficserver documentation for the Regex Remap plugin.

**regionalGeoBlocking** `true` if Regional Geo Blocking is in use within this Delivery Service, `false` otherwise - see Configure Regional Geo-blocking (RGB) for more information.

**remapText** Additional, raw text to add to the remap line for caches.

See also:

The Apache Trafficserver documentation for the Regex Remap plugin.

**signed** `true` if token-based authentication is enabled for this Delivery Service, `false` otherwise.

**signingAlgorithm** Type of URL signing method to sign the URLs, basically comes down to one of two plugins or `null`:

`null` Token-based authentication is not enabled for this Delivery Service.

**url_sig**: URL Signing token-based authentication is enabled for this Delivery Service.

**uri_signing** URI Signing token-based authentication is enabled for this Delivery Service.
See also:

The Apache Trafficserver documentation for the url_sig plugin and the draft RFC for uri Signing - note, however that the current implementation of uri Signing uses Draft 12 of that RFC document, NOT the latest

**sslKeyVersion** This integer indicates the generation of keys in use by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated

```
Warning: This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!
```

tenantId The integral, unique identifier of the tenant who owns this Delivery Service

**trRequestHeaders** If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where **RETURN** translates to a carriage return and line feed (\r\n)

**trResponseHeaders** If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it’s a template where **RETURN** translates to a carriage return and line feed (\r\n)

**type** The name of the routing type of this Delivery Service e.g. “HTTP”

**typeId** The integral, unique identifier of the routing type of this Delivery Service

**xmlId** A unique string that describes this Delivery Service - exists for legacy reasons

# 218: Response Example

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: SVveQShGwfpVv8N5APUuskwLoZwrTUYA+z8wuFLsSLCrI/vVnFJJ0QGmUctg1NbgAhUq795MJuucAwrR8dSOQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 19 Nov 2018 19:45:49 GMT
Content-Length: 1404

{ "alerts": [
    { "text": "Deliveryservice creation was successful.", "level": "success"
```
"active": false,
"anonymousBlockingEnabled": false,
"cacheurl": null,
"ccrDnsTtl": null,
"cdnId": 2,
"cdnName": "CDN-in-a-Box",
"checkPath": null,
"displayName": "test",
"dnsBypassCname": null,
"dnsBypassIp": null,
"dnsBypassIp6": null,
"dnsBypassTtl": null,
"dscp": 0,
"edgeHeaderRewrite": null,
"geoLimit": 0,
"geoLimitCountries": null,
"geoLimitRedirectURL": null,
"geoProvider": 0,
"globalMaxMbps": null,
"globalMaxTps": null,
"httpBypassFqdn": null,
"id": 2,
"infoUrl": null,
"initialDispersion": 1,
"ipv6RoutingEnabled": false,
"lastUpdated": "2018-11-19 19:45:49+00",
"logsEnabled": true,
"longDesc": "A :term:`Delivery Service` created expressly for API documentation examples",
"longDesc1": null,
"longDesc2": null,
"matchList": [
  [
    "type": "HOST_REGEXP",
    "setNumber": 0,
    "pattern": ".*\.test\.*"
  ]
],
"maxDnsAnswers": null,
"maxOriginConnections": 0,
"midHeaderRewrite": null,
"missLat": -1,
"missLong": -1,
"multiSiteOrigin": false,
"originShield": null,
"orgServerFqdn": "http://origin.infra.ciab.test",
"profileDescription": null,
"profileId": null,
"profileName": null,
"protocol": 0,
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regexRemap": null,
"regionalGeoBlocking": false,
"remapText": null,
"routingName": "test",
"signed": false,
"sslKeyVersion": null,
"tenantId": 1,
"type": "HTTP",
"typeId": 1,
"xmlId": "test",
"exampleURLs": [
   "http://test.test.mycdn.ciab.test"
],
"deepCachingType": "NEVER",
"signingAlgorithm": null,
"tenant": "root"
}}

**deliveryservices/dnsseckeys/generate**

**POST**

Generates Zone-Signing Key (ZSK) and Key-Signing Key (KSK) keypairs for a CDN and all associated Delivery Services.

- **Auth. Required**: Yes
- **Roles Required**: “admin”
- **Response Type**: Object (string)

**Request Structure**

- **key**: Name of the CDN
- **name**: Domain name used by the CDN
- **ttl**: Time for which the keypairs shall remain valid
- **kskExpirationDays**: Expiration (in days) for the KSKs
zskExpirationDays  Expiration (in days) for the ZSKs

effectiveDate  UNIX epoch start date for the signing keys

Changed in version 1.2: Added required ‘effectiveDate’ field to request

Response Structure

# 219: Response Example

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

deliveryservices/hostname/{{name}}/sslkeys

GET

Auth. Required  Yes

Roles Required  “admin”

Response Type  Object

Request Structure

Table 124: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The hostname of the desired Delivery Service - has the form {{routing name}}.{{xml_id}}.{{CDN domain}}</td>
</tr>
</tbody>
</table>

Table 125: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>no</td>
<td>The version number of SSL keys which shall be retrieved</td>
</tr>
</tbody>
</table>

# 220: Request Example

GET /api/1.4/deliveryservices/hostname/video.demo1.mycdn.ciab.test/sslkeys HTTP/1.1 Host: trafficops.infra.ciab.test User-Agent: curl/7.47.0 Accept: / Cookie: mojolicious=...
Response Structure

**businessUnit**  An optional field which, if present, contains the business unit entered by the user when generating the SSL certificate

**certificate**  An object containing the actual generated key, certificate, and signature of the SSL keys

- **crt**  Base 64-encoded (or not if the `decode` query parameter was given and `true`) certificate for the *Delivery Service* identified by `deliveryservice`

- **csr**  Base 64-encoded (or not if the `decode` query parameter was given and `true`) csr file for the *Delivery Service* identified by `deliveryservice`

- **key**  Base 64-encoded (or not if the `decode` query parameter was given and `true`) private key for the *Delivery Service* identified by `deliveryservice`

**Caution:** There’s almost certainly no good reason to request the private key! Even when “base 64-encoded” do not let ANYONE see this who would be unable to request it themselves!

- **cdn**  The CDN of the *Delivery Service* for which the certs were generated

- **city**  An optional field which, if present, contains the city entered by the user when generating the SSL certificate

- **country**  An optional field which, if present, contains the country entered by the user when generating the SSL certificate

- **deliveryservice**  The ‘xml_id’ of the *Delivery Service* for which the certificate was generated

- **hostname**  The hostname generated by Traffic Ops that is used as the common name when generating the certificate - this will be an FQDN for DNS *Delivery Services* and a wildcard URL for HTTP *Delivery Services*

- **organization**  An optional field which, if present, contains the organization entered by the user when generating certificate

- **state**  An optional field which, if present, contains the state entered by the user when generating certificate

- **version**  The version of the certificate record in Traffic Vault

**Note:** The response example uses abbreviated values for the `crt`, `key`, and `csr`, as these will generally be very large, base64-encoded SSL keys and certificates. Note that in general

---

1 These optional fields will be present in the response if and only if they were specified during key generation; they are optional during key generation and thus cannot be guaranteed to exist or not exist.
the output of this request should **not** be made available, as the **key** field contains the **private** SSL key corresponding to the certificate.

deliveryservices/{{ID}}

Deprecated since version 1.1: Use the **id** query parameter of deliveryservices instead

**GET**

Retrieves a specific *Delivery Service*

**Auth. Required**  Yes

**Roles Required**  None\(^1\)

**Response Type**  Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn</td>
<td>no</td>
<td>Show only the Delivery Services belonging to the CDN identified by this integral, unique identifier</td>
</tr>
<tr>
<td>logsEnabled</td>
<td>no</td>
<td>If true, return only Delivery Services with logging enabled, otherwise return only Delivery Services with logging disabled</td>
</tr>
<tr>
<td>profile</td>
<td>no</td>
<td>Return only Delivery Services using the profile identified by this integral, unique identifier</td>
</tr>
<tr>
<td>tenant</td>
<td>no</td>
<td>Show only the Delivery Services belonging to the tenant identified by this integral, unique identifier</td>
</tr>
<tr>
<td>type</td>
<td>no</td>
<td>Return only Delivery Services of the Delivery Service type identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>

**Response Structure**

Changed in version 1.3: Removed fqPacingRate field, added fields: deepCachingType, signingAlgorithm, and tenant.

\(^1\) Users with the roles “admin” and/or “operation” will be able to see all Delivery Services, whereas any other user will only see the Delivery Services their Tenant is allowed to see.
active  true if the Delivery Service is active, false otherwise

anonymousBlockingEnabled  true if Anonymous Blocking has been configured for the Delivery Service, false otherwise

cacheurl  A setting for a deprecated feature of now-unsupported Trafficserver versions

  Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later

ccrDnsTtl  The Time To Live (TTL) of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons

cdnId  The integral, unique identifier of the CDN to which the Delivery Service belongs

cdnName  Name of the CDN to which the Delivery Service belongs

checkPath  The path portion of the URL to check connections to this Delivery Service’s origin server

consistentHashRegex  If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services

  New in version 1.5.

deepCachingType  A string that describes when “Deep Caching” will be used by this Delivery Service - one of:

  ALWAYS  “Deep Caching” will always be used with this Delivery Service

  NEVER  “Deep Caching” will never be used with this Delivery Service

  New in version 1.3.

displayName  The display name of the Delivery Service

dnsBypassCname  Domain name to overflow requests for HTTP Delivery Services - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp  The IPv4 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp6  The IPv6 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassTtl  The time for which a DNS bypass of this Delivery Services shall remain active
dscp The FQDN with which to mark traffic as it leaves the CDN and reaches clients

edgeHeaderRewrite Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite ATS plugin

fqPacingRate The Fair-Queuing Pacing Rate in Bytes per second set on the all TCP connection sockets in the Delivery Service (see man tc-fc_codel for more information) - Linux only

    Deprecated since version 1.3: This field is only present/available in API versions 1.2 and lower - it has been removed in API version 1.3

geoLimit The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

    0 None - no limitations

    1 Only route when the client’s IP is found in the Coverage Zone File

    2 Only route when the client’s IP is found in the Coverage Zone File, or when the client can be determined to be from the United States of America

    Warning: This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

geoProvider An integer that represents the provider of a database for mapping IPs to geographic locations; currently only 0 - which represents MaxMind - is supported

globalMaxMbps The maximum global bandwidth allowed on this Delivery Service. If exceeded, traffic will be routed to dnsBypassIp (or dnsBypassIp6 for IPv6 traffic) for DNS Delivery Services and to httpBypassFqdn for HTTP Delivery Services

globalMaxTps The maximum global transactions per second allowed on this Delivery Service. When this is exceeded traffic will be sent to the dnsByPassIp* for DNS Delivery Services and to the httpBypassFqdn for HTTP Delivery Services

httpBypassFqdn The HTTP destination to use for bypass on an HTTP Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service
id  An integral, unique identifier for this Delivery Service

infoUrl  This is a string which is expected to contain at least one URL pointing to more information about the Delivery Service. Historically, this has been used to link relevant JIRA tickets

initialDispersion  The number of caches between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1

ipv6RoutingEnabled  If true, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache; if false all addresses will be IPv4, regardless of the client connection

lastUpdated  The date and time at which this Delivery Service was last updated, in a ctime-like format

logsEnabled  If true, logging is enabled for this Delivery Service, otherwise it is disabled

longDesc  A description of the Delivery Service

longDesc1  A field used when more detailed information that that provided by longDesc is desired

longDesc2  A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired

matchList  An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service

   pattern  A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

   setNumber  An integral, unique identifier for the set of types to which the type field belongs

   type  The type of match performed using pattern to determine whether or not to use this Delivery Service

      HOST_REGEXP  Use the Delivery Service if pattern matches the Host: HTTP header of an HTTP request

      HEADER_REGEXP  Use the Delivery Service if pattern matches an HTTP header (both the name and value) in an HTTP request

      PATH_REGEXP  Use the Delivery Service if pattern matches the request path of this Delivery Service's URL

---

2  This only applies to HTTP Delivery Services
STEERING_REGEXP Use the Delivery Service if the pattern matches the xml_id of one of this Delivery Service’s “Steering” target Delivery Services.

**maxDnsAnswers** The maximum number of IPs to put in a A/AAAA response for a DNS Delivery Service (0 means all available)

**midHeaderRewrite** Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

**missLat** The latitude to use when the client cannot be found in the CZF or a geographic IP lookup

**missLong** The longitude to use when the client cannot be found in the CZF or a geographic IP lookup

**multiSiteOrigin** true if the Multi Site Origin feature is enabled for this Delivery Service, false otherwise

**originShield** An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This field is a string of FQDNs to use as origin shields, delimited by |

**orgServerFqdn** The origin server’s Fully Qualified Domain Name (FQDN) - including the protocol (e.g. http:// or https://) - for use in retrieving content from the origin server

**profileDescription** The description of the Traffic Router Profile with which this Delivery Service is associated

**profileId** The integral, unique identifier for the Traffic Router profile with which this Delivery Service is associated

**profileName** The name of the Traffic Router Profile with which this Delivery Service is associated

**protocol** The protocol which clients will use to communicate with Edge-tier cache server - this is an integer on the interval [0-2] where the values have these meanings:

0 HTTP

1 HTTPS

2 Both HTTP and HTTPS

**qstringIgnore** Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where the values have these meanings:

0 URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin

---

3 See Use Multi-Site Origin Feature
1. URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin.

2. Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin.

**rangeRequestHandling**  
 Tells caches how to handle range requests\(^2\) - this is an integer on the interval \([0-2]\) where the values have these meanings:

- **0**: Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache.
- **1**: Use the **background_fetch** plugin to service the range request while caching the whole object.
- **2**: Use the experimental **cache_range_requests** plugin to treat unique ranges as unique objects.

**regexRemap**  
 A regular expression remap rule to apply to this *Delivery Service* at the Edge tier.

See also:

The Apache Trafficserver documentation for the Regex Remap plugin.

**regionalGeoBlocking**  
**true** if Regional Geo Blocking is in use within this *Delivery Service*, **false** otherwise - see **Configure Regional Geo-blocking (RGB)** for more information.

**remapText**  
 Additional, raw text to add to the remap line for caches.

See also:

The Apache Trafficserver documentation for the Regex Remap plugin.

**signed**  
**true** if token-based authentication is enabled for this *Delivery Service*, **false** otherwise.

**signingAlgorithm**  
Type of URL signing method to sign the URLs, basically comes down to one of two plugins or **null**:

- **null**: Token-based authentication is not enabled for this *Delivery Service*.
- **url_sig**: URL Signing token-based authentication is enabled for this *Delivery Service*.
- **uri_signing**: URI Signing token-based authentication is enabled for this *Delivery Service*.

See also:

The Apache Trafficserver documentation for the url_sig plugin and the draft RFC for uri_signing - note, however that the current implementation of uri_signing uses Draft 12 of that RFC document, NOT the latest.

New in version 1.3.
sslKeyVersion  This integer indicates the generation of keys in use by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated

**Warning:** This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

tenant  The name of the tenant who owns this Delivery Service

New in version 1.3.

tenantId  The integral, unique identifier of the tenant who owns this Delivery Service

tRequestHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)

tResponseHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)

type  The name of the routing type of this Delivery Service e.g. “HTTP”

typeld  The integral, unique identifier of the routing type of this Delivery Service

xmlId  A unique string that describes this Delivery Service - exists for legacy reasons

# 221: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Mw4ZsiNKfnx2vN+LsfAzxIZjgGTzcBLc2K24mMdhnN1XRMRtwej9VI3ExNvWkV3dp0f3RRCUTx6C+Si
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:43:36 GMT
Content-Length: 1290

{ "response": [ 
    {
        "active": true,
        "anonymousBlockingEnabled": false,
        "cacheurl": null,
        "ccrDnsTtl": null,
    }
]
"cdnId": 2,
"cdnName": "CDN-in-a-Box",
"checkPath": null,
"displayName": "Demo 1",
"dnsBypassCname": null,
"dnsBypassIp": null,
"dnsBypassIp6": null,
"dnsBypassTtl": null,
"dscp": 0,
"edgeHeaderRewrite": null,
"geoLimit": 0,
"geoLimitCountries": null,
"geoLimitRedirectURL": null,
"geoProvider": 0,
"globalMaxMbps": null,
"globalMaxTps": null,
"httpBypassFqdn": null,
"id": 1,
"infoUrl": null,
"initialDispersion": 1,
"ipv6RoutingEnabled": true,
"lastUpdated": "2018-11-14 18:21:17+00",
"logsEnabled": true,
"longDesc": "Apachecon North America 2018",
"longDesc1": null,
"longDesc2": null,
"matchList": [
    {
        "type": "HOST_REGEXP",
        "setNumber": 0,
        "pattern": ".*\.demo1\..*",
    }
],
"maxDnsAnswers": null,
"midHeaderRewrite": null,
"missLat": 42,
"missLong": -88,
"multiSiteOrigin": false,
"originShield": null,
"orgServerFqdn": "http://origin.infra.ciab.test",
"profileDescription": null,
"profileId": null,
"profileName": null,
"protocol": 0,
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regexRemap": null,
"regionalGeoBlocking": false,
"remapText": null,
"routingName": "video",
"signed": false,
"sslKeyVersion": null,
"tenantId": 1,
"type": "HTTP",
"typeId": 1,
"xmlId": "demo1",
"exampleURLs": [
  "http://video.demo1.mycdn.ciab.test"
],
"deepCachingType": "NEVER",
"signingAlgorithm": null,
"tenant": "root"
]
}

PUT

Allows users to edit an existing Delivery Service.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  NOT PRESENT - Despite returning a 200 OK response (rather than e.g. a 204 NO CONTENT response), this endpoint does not return a representation of the modified resource in its payload, and instead returns nothing - not even a success message.

Request Structure

active  If true, the Delivery Service will immediately become active and serves traffic

anonymousBlockingEnabled  An optional field which, if defined and true will cause Anonymous Blocking to be used with the new Delivery Service

cacheurl  An optional setting for a deprecated feature of now-unsupported Traffic-server versions (read: “Don’t use this”)

Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later

ccrDnsTtl  The Time To Live (TTL) in seconds of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons

Users with the roles “admin” and/or “operation” will be able to edit all Delivery Services, whereas any other user will only be able to edit the Delivery Services their Tenant is allowed to edit.
cdnId  The integral, unique identifier for the CDN to which this Delivery Service shall be assigned

checkPath  The path portion of the URL which will be used to check connections to this Delivery Service’s origin server

consistentHashRegex  If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services

New in version 1.5.

depthCachingType  A string describing when to do Deep Caching for this Delivery Service:

NEVER  Deep Caching will never be used by this Delivery Service (default)

ALWAYS  Deep Caching will always be used by this Delivery Service

displayName  The human-friendly name for this Delivery Service

dnsBypassCname  Domain name to overflow requests for HTTP Delivery Services - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp  The IPv4 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp6  The IPv6 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassTtl  The time for which a DNS bypass of this Delivery Service shall remain active

dscp  The Differentiated Services Code Point (DSCP) with which to mark down-stream (EDGE -> customer) traffic. This should be zero in most cases

edgeHeaderRewrite  An optional string which, if present, defines rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

fqPacingRate  An optional integer which, if present, sets the Fair-Queuing Pacing Rate in bytes per second set on the all TCP connection sockets in the Delivery Service (see man tc-fc_codel for more information) - Linux only, defaults to 0 meaning “disabled”

geoLimit  The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

0  None - no limitations
Only route when the client’s IP is found in the Coverage Zone File (CZF)

Only route when the client’s IP is found in the CZF, or when the client can be determined to be from the United States of America

**Warning:** This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

**geoLimitCountries** A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this *Delivery Service*.

**geoLimitRedirectUrl** A URL to which clients blocked by *Regional Geographic Blocking* or the *geoLimit* settings will be re-directed.

**geoProvider** An integer that represents the provider of a database for mapping IPs to geographic locations; currently only the following values are supported:

0 The “Maxmind” GeoIP2 database (default)

1 Neustar

**globalMaxMbps** An optional integer that will set the maximum global bandwidth allowed on this *Delivery Service*. If exceeded, traffic will be routed to *dnsBypassIp* (or *dnsBypassIp6* for IPv6 traffic) for DNS *Delivery Services* and to *httpBypassFqdn* for HTTP *Delivery Services*.

**globalMaxTps** An optional integer that will set the maximum global transactions per second allowed on this *Delivery Service*. When this is exceeded traffic will be sent to the *dnsBypassIp* (and/or *dnsBypassIp6*) for DNS *Delivery Services* and to the *httpBypassFqdn* for HTTP *Delivery Services*.

**httpBypassFqdn** An optional Fully Qualified Domain Name (FQDN) to use for bypass on an HTTP *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds *globalMaxMbps*, or when more than *globalMaxTps* is being exceeded within the *Delivery Service*.

**infoUrl** An optional string which, if present, is expected to contain at least one URL pointing to more information about the *Delivery Service*. Historically, this has been used to link relevant JIRA tickets.

**initialDispersion** The number of caches between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1.

**ipv6RoutingEnabled** If true, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache; if false all addresses will be IPv4, regardless of the client connection - optional for ANY_MAP *Delivery Services*.

---

5 These fields must be defined if and only if *geoLimit* is non-zero

6 These fields are required for HTTP-routed *Delivery Services*, and optional for all others
logsEnabled If true, logging is enabled for this Delivery Service, otherwise it is disabled

longDesc An optional description of the Delivery Service

longDesc1 An optional field used when more detailed information that that provided by longDesc is desired

longDesc2 An optional field used when even more detailed information that that provided by either longDesc or longDesc1 is desired

maxDnsAnswers An optional field which, when present, specifies the maximum number of IPs to put in responses to A/AAAA DNS record requests - defaults to 0, meaning “no limit”\(^4\)

midHeaderRewrite An optional string containing rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

missLat The latitude to use when the client cannot be found in the CZF or a geographic IP lookup\(^7\)

missLong The longitude to use when the client cannot be found in the CZF or a geographic IP lookup\(^7\)

multiSiteOrigin true if the Multi Site Origin feature is enabled for this Delivery Service, false otherwise\(^3\)

orgServerFqdn The URL of the Delivery Service’s origin server for use in retrieving content from the origin server\(^7\)

---

**Note:** Despite the field name, this must truly be a full URL - including the protocol (e.g. http:// or https://) - NOT merely the server’s Fully Qualified Domain Name (FQDN)

originShield An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This optional field is a string of FQDNs to use as origin shields, delimited by |.

profileId An optional, integral, unique identifier for the Traffic Router profile with which this Delivery Service shall be associated

protocol The protocol which clients will use to communicate with Edge-tier cache servers - this is an (optional for ANY_MAP Delivery Services) integer on the interval \([0, 2]\) where the values have these meanings:

- **0** HTTP
- **1** HTTPS

---

\(^4\) This only applies to DNS-routed Delivery Services

\(^7\) These fields are required for HTTP-routed and DNS-routed Delivery Services, but are optional for (and in fact may have no effect on) STEERING and ANY_MAP Delivery Services
Both HTTP and HTTPS

**qstringIgnore** Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where the values have these meanings:

0 URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin

1 URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin

2 Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin

**rangeRequestHandling** Tells caches how to handle range requests - this is an integer on the interval [0,2] where the values have these meanings:

0 Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache

1 Use the background_fetch plugin to service the range request while caching the whole object

2 Use the experimental cache_range_requests plugin to treat unique ranges as unique objects

**regexRemap** An optional, regular expression remap rule to apply to this Delivery Service at the Edge tier

See also:

The Apache Trafficserver documentation for the Regex Remap plugin

**regionalGeoBlocking** true if Regional Geo Blocking is in use within this Delivery Service, false otherwise - see Configure Regional Geo-blocking (RGB) for more information

**remapText** Optional, raw text to add to the remap line for caches

See also:

The Apache Trafficserver documentation for the Regex Remap plugin

**routingName** The routing name of this Delivery Service, used as the top-level part of the FQDN used by clients to request content from the Delivery Service e.g. routingName.xml_id.CDNName.com

**signed** An optional field which should be true if token-based authentication will be enabled for this Delivery Service, false (default) otherwise

**signingAlgorithm** Type of URL signing method to sign the URLs, basically comes down to one of two plugins or null:

---

8 See “token-based-auth” TODO — wat for more information
null Token-based authentication is not enabled for this Delivery Service

url_sig: URL Signing token-based authentication is enabled for this Delivery Service

uri_signing URI Signing token-based authentication is enabled for this Delivery Service

See also:

The Apache Trafficserver documentation for the url_sig plugin and the draft RFC for uri_signing - note, however that the current implementation of uri_signing uses Draft 12 of that RFC document, NOT the latest

sslKeyVersion This optional integer indicates the generation of keys to be used by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated

Warning: This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

tenantId An optional, integral, unique identifier of the tenant who will own this Delivery Service

trRequestHeaders If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)

trResponseHeaders If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)

typeId The integral, unique identifier for the routing type of this Delivery Service

xmlId A unique string that describes this Delivery Service - exists for legacy reasons

Note: While this field must be present, it is not allowed to change; this must be the same as the xml_id the Delivery Service already has. This should almost never be different from the Delivery Service's displayName.

# 222: Request Example

```bash
PUT /api/1.4/deliveryservices/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 761
Content-Type: application/json
```

(continues on next page)
{  
  "active": true,  
  "anonymousBlockingEnabled": false,  
  "cdnId": 2,  
  "cdnName": "CDN-in-a-Box",  
  "deepCachingType": "NEVER",  
  "displayName": "demo",  
  "exampleURLs": [  
    "http://video.demo.mycdn.ciab.test"  
  ],  
  "dscp": 0,  
  "geoLimit": 0,  
  "geoProvider": 0,  
  "initialDispersion": 1,  
  "ipv6RoutingEnabled": false,  
  "lastUpdated": "2018-11-14 18:21:17+00",  
  "logsEnabled": true,  
  "longDesc": "A :term:`Delivery Service` created expressly for API documentation examples",  
  "missLat": -1,  
  "missLong": -1,  
  "multiSiteOrigin": false,  
  "orgServerFqdn": "http://origin.infra.ciab.test",  
  "protocol": 0,  
  "qstringIgnore": 0,  
  "rangeRequestHandling": 0,  
  "regionalGeoBlocking": false,  
  "routingName": "video",  
  "signed": false,  
  "tenant": "root",  
  "tenantId": 1,  
  "typeId": 1,  
  "xmlId": "demo1"  
}

Response Structure

# 223: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Set-Cookie: mojolicious=...; Path=/; HttpOnly

(continues on next page)
DELETE

Deletes the target Delivery Service

**Auth. Required** Yes

**Roles Required** “admin” or “operations”¹¹

**Response Type** undefined

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery Service to be retrieved</td>
</tr>
</tbody>
</table>

# 224: Request Example

```
DELETE /api/1.4/deliveryservices/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

# 225: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
```

¹¹ Users with the roles “admin” and/or “operation” will be able to delete all Delivery Services, whereas any other user will only be able to delete the Delivery Services their Tenant is allowed to delete.
deliveryservices/{{ID}}/capacity

See also:

*Health Protocol*

**GET**

Retrieves the usage percentages of a server associated with a Delivery Service

**Auth. Required** Yes

**Roles Required** “admin” or “operations”¹

**Response Type** Object

**Request Structure**

Table 129: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the Delivery Service of interest</td>
</tr>
</tbody>
</table>

**Response Structure**

**availablePercent** The percent of servers assigned to this Delivery Service that is available - the allowed traffic level in terms of data per time period for all cache servers that remains unused

¹ Users with the roles “admin” and/or “operations” will be able to see details for all Delivery Services, whereas any other user will only see details for the Delivery Services their Tenant is allowed to see.
unavailablePercent The percent of servers assigned to this Delivery Service that is unavailable - the allowed traffic level in terms of data per time period for all cache servers that can’t be used because the servers are deemed unhealthy

utilizedPercent The percent of servers assigned to this Delivery Service that is currently in use - the allowed traffic level in terms of data per time period that is currently devoted to servicing requests

maintenancePercent The percent of servers assigned to this Delivery Service that is unavailable due to server maintenance - the allowed traffic level in terms of data per time period that is unavailable because servers have intentionally been marked offline by administrators

# 226: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 15 Nov 2018 14:41:27 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 18:41:27 GMT;
\path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:++dFR9V1c60CHGNwMjX6JSFEjHreXcL4QnhT03hiv04ByY379aLpL4OrOzX2bPgLjpR94+f6jZ0+ID
Content-Length: 134

{ "response": {
    "availablePercent": 99.9993696969697,
    "unavailablePercent": 0,
    "utilizedPercent": 0.00063030303030303,
    "maintenancePercent": 0
}}
```

deliveryservices/{{ID}}/health

See also:

Health Protocol

GET

Retrieves the health of all Cache Groups assigned to a particular Delivery Service

Auth. Required Yes

5.1. Traffic Ops API
Roles Required “admin” or “operations”\(^1\)

Response Type Object

Request Structure

Table 130: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery service for which Cache Groups will be displayed</td>
</tr>
</tbody>
</table>

Response Structure

**cachegroups** An array of objects that represent the health of each Cache Group assigned to this Delivery Service

- **name** The name of the Cache Group represented by this object
- **offline** The number of offline cache servers within this Cache Group
- **online** The number of online cache servers within this Cache Group
- **totalOffline** Total number of offline cache servers assigned to this Delivery Service
- **totalOnline** Total number of online cache servers assigned to this Delivery Service

# 227: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 15 Nov 2018 14:43:43 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 18:43:43 GMT; path=/; HttpOnly
Vary: Accept-Encoding
```

\(^1\) Users with the roles “admin” and/or “operations” will be able to see Cache Groups associated with any Delivery Services, whereas any other user will only be able to see the Cache Groups associated with Delivery Services their Tenant is allowed to see.
deliveryservices/{{ID}}/regexes

GET

Retrieves routing regular expressions for a specific Delivery Service.

Auth. Required  Yes
Roles Required  None¹
Response Type  Array

Request Structure

Table 131: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery Service being inspected</td>
</tr>
</tbody>
</table>

# 228: Request Example

```
GET /api/1.4/deliveryservices/1/regexes HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

¹ If tenancy is used, then users (regardless of role) will only be able to see the routing regular expressions used by Delivery Services their tenant has permissions to see.
Response Structure

- **id**: The integral, unique identifier of this regular expression
- **pattern**: The actual regular expression - \s are escaped
- **setNumber**: The order in which the regular expression is evaluated against requests
- **type**: The integral, unique identifier of the type of this regular expression
- **typeName**: The Type of regular expression - determines that against which it will be evaluated

# 229: Response Example

```json
{
  "response": [
    {
      "id": 1,
      "type": 31,
      "typeName": "HOST_REGEXP",
      "setNumber": 0,
      "pattern": ".*\\.demo1\..*"
    }
  ]
}
```

**POST**

Creates a routing regular expression for a **Delivery Service**.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”\(^2\)

**Response Type** Object

---

\(^2\) If tenancy is used, then users (regardless of role) will only be able to edit the routing regular expressions used by **Delivery Services** their tenant has permissions to edit. Assuming tenancy is satisfied, a routing regular expression can only be created by a user with the “admin” or “operations” role.
Request Structure

Table 132: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the <em>Delivery Service</em> being inspected</td>
</tr>
</tbody>
</table>

**pattern**  The actual regular expression

**Warning:** Be sure that \s are escaped, or the expression may not work as intended!

**setNumber**  The order in which this regular expression should be checked

**type**  The integral, unique identifier of a routing regular expression type

# 230: Request Example

```plaintext
POST /api/1.4/deliveryservices/1/regexes HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 55
Content-Type: application/json

{
    "pattern": ".*\\.foo-bar\\..*",
    "type": 31,
    "setNumber": 1
}
```

Response Structure

**id**  The integral, unique identifier of this regular expression

**pattern**  The actual regular expression - \s are escaped

**setNumber**  The order in which the regular expression is evaluated against requests

**type**  The integral, unique identifier of the type of this regular expression

**typeName**  The type of regular expression - determines that against which it will be evaluated
# 231: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: kS5dRzAhFKE7vfzHK7XVIwpMOjztksk9MU+qtj5YU/1oxVHmqNbJ12Fe0OIJsZJCXbY1nBS04sCI95Sz5wed1Q==
X-Server-Name: traffic_ops_golang/
Date: Wed, 28 Nov 2018 17:00:42 GMT
Content-Length: 188

{
  "alerts": [
    {
      "text": "Delivery service regex creation was successful.",
      "level": "success"
    }
  ],
  "response": {
    "id": 2,
    "type": 31,
    "typeName": "HOST_REGEXP",
    "setNumber": 1,
    "pattern": ".*\./foo-bar\..*"
  }
}
```

deliveryservices/{{ID}}/regexes/{{rID}}

**GET**

Retrieves a specific routing regular expression for a specific *Delivery Service*.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

---

1 If tenancy is used, then users (regardless of role) will only be able to see the routing regular expressions used by *Delivery Services* their tenant has permissions to see.
Request Structure

Table 133: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the <em>Delivery Service</em> being inspected</td>
</tr>
<tr>
<td>rID</td>
<td>The integral, unique identifier of the routing regular expression being inspected</td>
</tr>
</tbody>
</table>

# 232: Request Example

```bash
GET /api/1.4/deliveryservices/1/regexes/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **id** The integral, unique identifier of this regular expression
- **pattern** The actual regular expression - \s are escaped
- **setNumber** The order in which the regular expression is evaluated against requests
- **type** The integral, unique identifier of the type of this regular expression
- **typeName** The type of regular expression - determines that against which it will be evaluated

# 233: Response Example

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: fW9Fde4WRpp2ShRAc41P9s/PhU71LI/SEzHqYjGqLzhk45w0kpaWy76JvPfLp0y8eDTPs8Y8TL5rNGEc+bM+A==
X-Server-Name: traffic_ops_golang/
Date: Tue, 27 Nov 2018 21:08:34 GMT
Content-Length: 100

{ "response": [
    {
      "id": 1,
      ...
    }
  ]
```

(continues on next page)
"type": 31,
"typeName": "HOST_REGEXP",
"setNumber": 0,
"pattern": ".*\\.demo1\..*"
}
]

PUT

Updates a routing regular expression.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”²

**Response Type** Object

Request Structure

Table 134: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the <em>Delivery Service</em> being inspected</td>
</tr>
<tr>
<td>rID</td>
<td>The integral, unique identifier of the routing regular expression being inspected</td>
</tr>
</tbody>
</table>

**pattern** The actual regular expression

**Warning:** Be sure that \s are escaped, or the expression may not work as intended!

**setNumber** The order in which this regular expression should be checked

**type** The integral, unique identifier of a routing regular expression type

# 234: Request Example

```
PUT /api/1.4/deliveryservices/1/regexes/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 55
Content-Type: application/json
```

² If tenancy is used, then users (regardless of role) will only be able to edit the routing regular expressions used by *Delivery Services* their tenant has permissions to edit. Assuming tenancy is satisfied, a routing regular expression can only be edited by a user with the “admin” or “operations” role.
Response Structure

**id**  The integral, unique identifier of this regular expression

**pattern**  The actual regular expression - \s are escaped

**setNumber**  The order in which the regular expression is evaluated against requests

**type**  The integral, unique identifier of the type of this regular expression

**typeName**  The type of regular expression - determines that against which it will be evaluated

# 235: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: kS5dRzAhFKE7vfzHK7XVIwpMOjztksk9MU+qtj5YU/1oxVHmqNbJ12Fe0OIJ8ZJCXbYlnBS04sCI95Sz5wed1Q==
X-Server-Name: traffic_ops_golang/
Date: Thu, 29 Nov 2018 17:54:58 GMT
Content-Length: 188

{
    "alerts": [
        {
            "text": "Delivery service regex creation was successful."
            "level": "success"
        }
    ],
    "response": {
        "id": 2,
        "type": 33,
        "typeName": "PATH_REGEXP",
        "setNumber": 1
    }
}
```
"pattern": ".*\\ foo-bar\ \.*"
}}

DELETE

Deletes a routing regular expression.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”\(^3\)

**Response Type** undefined

**Request Structure**

**Table 135: Request Path Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery Service being inspected</td>
</tr>
<tr>
<td>rID</td>
<td>The integral, unique identifier of the routing regular expression being inspected</td>
</tr>
</tbody>
</table>

**# 236: Request Example**

DELETE /api/1.4/deliveryservices/1/regexes/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

**Response Structure**

**# 237: Response Example**

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly

\(^3\) If tenancy is used, then users (regardless of role) will only be able to delete the routing regular expressions used by Delivery Services their tenant has permissions to delete. Assuming tenancy is satisfied, a routing regular expression can only be deleted by a user with the “admin” or “operations” role.
deliveryservices/{{ID}}/routing

GET

Retrieves routing method statistics for a particular Delivery Service

Auth. Required Yes
Roles Required “admin” or “operations”¹
Response Type Object

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the Delivery Service of interest</td>
</tr>
</tbody>
</table>

# 238: Request Example

```
GET /api/1.4/deliveryservices/1/routing HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

¹ Users with the roles “admin” and/or “operations” will be able to see details for all Delivery Services, whereas any other user will only see details for the Delivery Services their Tenant is allowed to see.
Response Structure

**cz** The percent of requests to the Traffic Router for this *Delivery Service* that were satisfied by a coverage zone file (CZF)

**dsr** The percent of requests to the Traffic Router for this *Delivery Service* that were satisfied by sending the client to an overflow *Delivery Service*

**err** The percent of requests to the Traffic Router for this *Delivery Service* that resulted in an error

**fed** The percent of requests to the Traffic Router for this *Delivery Service* that were satisfied by sending the client to a federated CDN

**geo** The percent of requests to the Traffic Router for this *Delivery Service* that were satisfied using 3rd party geographic IP mapping

**miss** The percent of requests to the Traffic Router for this *Delivery Service* that could not be satisfied

**regionalAlternate** The percent of requests to the Traffic Router for this *Delivery Service* that were satisfied by sending the client to the alternate, Regional Geo-blocking URL

**regionalDenied** The percent of Traffic Router requests for this *Delivery Service* that were denied due to geographic location policy

**staticRoute** The percent of requests to the Traffic Router for this *Delivery Service* that were satisfied with pre-configured DNS entries

---

**HTTP/1.1 200 OK**

Access-Control-Allow-Credentials: true

Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept

Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE

Access-Control-Allow-Origin: *

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

Content-Type: application/json

Date: Fri, 30 Nov 2018 15:08:07 GMT

Server: Mojolicious (Perl)

Set-Cookie: mojolicious=...; expires=Fri, 30 Nov 2018 19:08:07 GMT; path=/; HttpOnly

Vary: Accept-Encoding

Whole-Content-Sha512: UgPziRC/5u4+CfkZ9xm0EkEzjjJVu6cwBrFd/n3xH/ZmlkaXkQaaly4+B7DyE46vxFLYE0ODoQchyn7JkoQOg==

Content-Length: 132

```json
{
  "response": {
    "staticRoute": 0,
    "geo": 100,
    "err": 0,
  }
}
```

(continues on next page)
"fed": 0,
"cz": 0,
"dsr": 0,
"regionalAlternate": 0,
"deepCz": 0,
"regionalDenied": 0,
"miss": 0
}
}

deliveryservices/{{ID}}/safe

PUT

Allows a user to edit metadata fields of a Delivery Service.

Auth. Required Yes

Roles Required “admin” or “operations”\(^1\)

Response Type Array

Request Structure

Table 137: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery Service being modified</td>
</tr>
</tbody>
</table>

displayName The human-friendly name for this Delivery Service

infoUrl A string which is expected to contain at least one URL pointing to more information about the Delivery Service. Historically, this has been used to link relevant JIRA tickets

longDesc A description of the Delivery Service

longDesc1 A field used when more detailed information that that provided by longDesc is desired

longDesc2 A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired

**Note:** All of these fields are optional; this PUT behaves more like a PATCH

\(^1\) Users with the “admin” or “operations” roles will be able to edit any Delivery Service, whereas other users will only be able to edit Delivery Services that their tenant has permissions to edit.
# 240: Request Example

```plaintext
PUT /api/1.4/deliveryservices/1/safe HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 165
Content-Type: application/x-www-form-urlencoded

{
    "displayName": "demo",
    "infoUrl": "www.info.com",
    "longDesc": "A :term:`Delivery Service` created for the CDN-in-a-Box project",
    "longDesc1": null,
    "longDesc2": null
}
```

Response Structure

Changed in version 1.3: Removed fqPacingRate field, added fields: deepCachingType, signingAlgorithm, and tenant.

- **active** true if the Delivery Service is active, false otherwise
- **anonymousBlockingEnabled** true if Anonymous Blocking has been configured for the Delivery Service, false otherwise
- **cacheurl** A setting for a deprecated feature of now-unsupported Trafficserver versions
  - Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later
- **ccrDnsTtl** The Time To Live (TTL) of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons
- **cdnId** The integral, unique identifier of the CDN to which the Delivery Service belongs
- **cdnName** Name of the CDN to which the Delivery Service belongs
- **checkPath** The path portion of the URL to check connections to this Delivery Service’s origin server
- **consistentHashRegex** If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services
  - New in version 1.5.
deepCachingType A string that describes when “Deep Caching” will be used by this Delivery Service - one of:

  ALWAYS “Deep Caching” will always be used with this Delivery Service
  NEVER “Deep Caching” will never be used with this Delivery Service

New in version 1.3.

displayName The display name of the Delivery Service

dnsBypassCname Domain name to overflow requests for HTTP Delivery Services - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp The IPv4 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp6 The IPv6 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassTtl The time for which a DNS bypass of this Delivery Service shall remain active

dscp The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

edgeHeaderRewrite Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

fqPacingRate The Fair-Queuing Pacing Rate in Bytes per second set on the all TCP connection sockets in the Delivery Service (see man tc-fc_codel for more information) - Linux only

Deprecated since version 1.3: This field is only present/available in API versions 1.2 and lower - it has been removed in API version 1.3

geoLimit The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

  0 None - no limitations

  1 Only clients who request content from the same IP as the CDN's geographic location are allowed to request content through this Delivery Service

  2 Only clients who request content from the same geographic location as the CDN as specified in the geoLimitCountries setting are allowed to request content through this Delivery Service

geoLimitCountries A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this Delivery Service

geoLimitRedirectUrl A URL to which clients blocked by Regional Geographic Blocking or the geoLimit settings will be re-directed

4 This only applies to DNS-routed Delivery Services
Only route when the client’s IP is found in the Coverage Zone File (CZF)

Only route when the client’s IP is found in the CZF, or when the client can be determined to be from the United States of America

**Warning:** This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

**geoProvider** An integer that represents the provider of a database for mapping IPs to geographic locations; currently only 0 - which represents MaxMind - is supported

**globalMaxMbps** The maximum global bandwidth allowed on this *Delivery Service*. If exceeded, traffic will be routed to `dnsBypassIp` (or `dnsBypassIp6` for IPv6 traffic) for DNS *Delivery Services* and to `httpBypassFqdn` for HTTP *Delivery Services*

**globalMaxTps** The maximum global transactions per second allowed on this *Delivery Service*. When this is exceeded traffic will be sent to the `dnsByPassIp*` for DNS *Delivery Services* and to the `httpBypassFqdn` for HTTP *Delivery Services*

**httpBypassFqdn** The HTTP destination to use for bypass on an HTTP *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds `globalMaxMbps`, or when more than `globalMaxTps` is being exceeded within the *Delivery Service*

**id** An integral, unique identifier for this *Delivery Service*

**infoUrl** This is a string which is expected to contain at least one URL pointing to more information about the *Delivery Service*. Historically, this has been used to link relevant JIRA tickets

**initialDispersion** The number of caches between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1

**ipv6RoutingEnabled** If `true`, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache; if `false` all addresses will be IPv4, regardless of the client connection

**lastUpdated** The date and time at which this *Delivery Service* was last updated, in a *ctime*-like format

**logsEnabled** If `true`, logging is enabled for this *Delivery Service*, otherwise it is disabled

**longDesc** A description of the *Delivery Service*

---

2 This only applies to HTTP-routed *Delivery Services*
longDesc1 A field used when more detailed information that that provided by longDesc is desired

longDesc2 A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired

matchList An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service

pattern A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

setNumber An integral, unique identifier for the set of types to which the type field belongs

type The type of match performed using pattern to determine whether or not to use this Delivery Service

HOST_REGEXP Use the Delivery Service if pattern matches the Host: HTTP header of an HTTP request²

HEADER_REGEXP Use the Delivery Service if pattern matches an HTTP header (both the name and value) in an HTTP request²

PATH_REGEXP Use the Delivery Service if pattern matches the request path of this Delivery Service’s URL

STEERING_REGEXP Use the Delivery Service if pattern matches the xml_id of one of this Delivery Service’s “Steering” target Delivery Services

maxDnsAnswers The maximum number of IPs to put in a A/AAAA response for a DNS Delivery Service (0 means all available)⁴

midHeaderRewrite Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

missLat The latitude to use when the client cannot be found in the CZF or a geographic IP lookup

missLong The longitude to use when the client cannot be found in the CZF or a geographic IP lookup

multiSiteOrigin true if the Multi Site Origin feature is enabled for this Delivery Service, false otherwise³

originShield An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This field is a string of FQDNs to use as origin shields, delimited by | ³

³ See Use Multi-Site Origin Feature

5.1. Traffic Ops API 451
orgServerFqdn The origin server’s Fully Qualified Domain Name (FQDN) - including the protocol (e.g. http:// or https://) - for use in retrieving content from the origin server

profileDescription The description of the Traffic Router Profile with which this Delivery Service is associated

profileId The integral, unique identifier for the Traffic Router profile with which this Delivery Service is associated

profileName The name of the Traffic Router Profile with which this Delivery Service is associated

protocol The protocol which clients will use to communicate with Edge-tier cache servers\(^2\) - this is an integer on the interval [0-2] where the values have these meanings:

0 HTTP

1 HTTPS

2 Both HTTP and HTTPS

qstringIgnore Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where the values have these meanings:

0 URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin

1 URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin

2 Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin

rangeRequestHandling Tells caches how to handle range requests\(^5\) - this is an integer on the interval [0-2] where the values have these meanings:

0 Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache

1 Use the background_fetch plugin to service the range request while caching the whole object

2 Use the experimental cache_range_requests plugin to treat unique ranges as unique objects

regexRemap A regular expression remap rule to apply to this Delivery Service at the Edge tier

---

\(^5\) These fields are required for HTTP-routed and DNS-routed Delivery Services, but are optional for (and in fact may have no effect on) STEERING and ANY_MAP Delivery Services
regionalGeoBlocking  true if Regional Geo Blocking is in use within this Delivery Service, false otherwise - see Configure Regional Geo-blocking (RGB) for more information

remapText  Additional, raw text to add to the remap line for caches

signed  true if token-based authentication is enabled for this Delivery Service, false otherwise

signingAlgorithm  Type of URL signing method to sign the URLs, basically comes down to one of two plugins or null:

null  Token-based authentication is not enabled for this Delivery Service

url_sig:  URL Signing token-based authentication is enabled for this Delivery Service

uri_signing  URI Signing token-based authentication is enabled for this Delivery Service

signed  true if token-based authentication is enabled for this Delivery Service, false otherwise

sslKeyVersion  This integer indicates the generation of keys in use by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated

Warning:  This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

tenant  The name of the tenant who owns this Delivery Service

New in version 1.3.

tenantId  The integral, unique identifier of the tenant who owns this Delivery Service

trRequestHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)²

5.1. Traffic Ops API
TRResponseHeaders If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it's a template where \_\_RETURN\_\_ translates to a carriage return and line feed (\r\n)²

type The name of the routing type of this Delivery Service e.g. “HTTP”
typeId The integral, unique identifier of the routing type of this Delivery Service
xmlId A unique string that describes this Delivery Service - exists for legacy reasons

# 241: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Mon, 19 Nov 2018 19:29:40 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 19 Nov 2018 23:29:40 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: wSCP0NQbFTN0FonjXYHl3jwTvOwo0ltSD0ACRQ4d/ewIfzNyAFAD/RafplUP2PIqttb6NlnHkKve0j6ETJ+gw==
Content-Length: 1439

{   "alerts": [       {           "level": "success",           "text": "Deliveryservice safe update was successful."
       },
       "response": [       {           "profileId": null,
           "protocol": 0,
           "deepCachingType": "NEVER",
           "regionalGeoBlocking": 0,
           "routingName": "video",
           "orgServerFqdn": "http://origin.infra.ciab.test",
           "cdnId": 2,
           "geoProvider": 0,
           "longDesc2": null,
           "globalMaxMbps": null,
           "dnsBypassIp6": null,
           "geoLimit": 0,
           "maxDnsAnswers": null,
       }
    ]
  }

(continues on next page)
"id": 1,
"sslKeyVersion": null,
"midHeaderRewrite": null,
"geoLimitRedirectURL": null,
"active": 1,
"logsEnabled": 1,
"initialDispersion": 1,
"regexRemap": null,
"geoLimitCountries": null,
"missLat": 42,
"anonymousBlockingEnabled": 0,
"longDesc": "A :term:`Delivery Service` created for the CDN-in-a-Box project,
"matchList": [
    {
        "pattern": ".*\demol\..*",
        "setNumber": 0,
        "type": "HOST_REGEXP"
    }
],
"rangeRequestHandling": 0,
"profileName": null,
"dnsBypassCname": null,
"globalMaxTps": null,
"type": "HTTP",
"httpBypassFqdn": null,
"infoUrl": "www.info.com",
"signingAlgorithm": null,
"missLong": -88,
"trRequestHeaders": null,
"trResponseHeaders": null,
"exampleURLs": [
    "http://video.demol.mycdn.ciab.test"
],
"remapText": null,
"longDesc1": null,
"displayName": "demo",
"qstringIgnore": 0,
"multiSiteOrigin": 0,
"xmlId": "demol",
"lastUpdated": "2018-11-19 16:26:57.310527+00",
"ipv6RoutingEnabled": 1,
"ccrDnsTtl": null,
"dscp": 0,
"dnsBypassIp": null,
"dnsBypassTtl": null,
"originShield": null,
"cacheurl": null,
"edgeHeaderRewrite": null,
deliveryservices/{{id}}/server_types/{{type}}/metric_types/start_date/{{start}}/end_date/{{end}}

Danger: This endpoint doesn’t appear to work, and so its use is strongly discouraged! The below documentation cannot be verified, therefore it may be inaccurate and/or incomplete!

GET

Retrieves detailed and summary metrics for Mid-tier and Edge-tier caches assigned to a Delivery Service.

Auth. Required  Yes

Roles Required  None

Response Type  Array

Request Structure

Table 138: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The delivery service id.</td>
</tr>
<tr>
<td>server_types</td>
<td>yes</td>
<td>EDGE or MID.</td>
</tr>
<tr>
<td>metric_types</td>
<td>yes</td>
<td>One of the following: “kbps”, “tps”, “tps_2xx”, “tps_3xx”, “tps_4xx”, “tps_5xx”.</td>
</tr>
<tr>
<td>start_date</td>
<td>yes</td>
<td>UNIX time</td>
</tr>
<tr>
<td>end_date</td>
<td>yes</td>
<td>UNIX time</td>
</tr>
</tbody>
</table>

Table 139: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stats</td>
<td>no</td>
<td>Flag used to return only summary metrics</td>
</tr>
</tbody>
</table>
Response Structure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stats</td>
<td>hash</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;count</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;98thPercentile</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;min</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;max</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;5thPercentile</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;95thPercentile</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;median</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;mean</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;stddev</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;sum</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;item</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;time</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;value</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

# 242: 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"
    }
  ]
```

(continues on next page)
deliveryservices/{{ID}}/servers

Deprecated since version 1.1: Use deliveryservice instead

GET

Retrieves properties of Edge-Tier servers assigned to a Delivery Service.

Auth. Required Yes

Roles Required “admin” or “operations”¹

Response Type Array

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery service for which servers will be displayed</td>
</tr>
</tbody>
</table>

Response Structure

- cachegroup The name of the Cache Group to which the server belongs
- cachegroupId An integral, unique identifier for the Cache Group to which the server belongs
- cdnId An integral, unique identifier the CDN to which the server belongs
- cdnName The name of the CDN to which the server belongs
- domainName The domain name part of the Fully Qualified Domain Name (FQDN) of the server
- guid Optionally represents an identifier used to uniquely identify the server
- hostName The (short) hostname of the server
- httpsPort The port on which the server listens for incoming HTTPS requests - 443 in most cases

¹ Users with the roles “admin” and/or “operations” will be able to see servers associated with any Delivery Services, whereas any other user will only be able to see the servers associated with Delivery Services their Tenant is allowed to see.
id  An integral, unique identifier for the server

iloIpAddress  The IPv4 address of the lights-out-management port

iloIpGateway  The IPv4 gateway address of the lights-out-management port

iloIpNetmask  The IPv4 subnet mask of the lights-out-management port

iloPassword  The password of the lights-out-management user - displays as ****** unless the requesting user has the ‘admin’ role

iloUsername  The user name for lights-out-management

interfaceMtu  The Maximum Transmission Unit (MTU) to configure for interfaceName

See also:
The Wikipedia article on Maximum Transmission Unit

interfaceName  The network interface name used by the server

ip6Address  The IPv6 address and subnet mask of the server - applicable for the interfaceName

ip6Gateway  The IPv6 gateway address of the server - applicable for the interfaceName

ipAddress  The IPv4 address of the server - applicable for the interfaceName

ipGateway  The IPv4 gateway of the server - applicable for the interfaceName

ipNetmask  The IPv4 subnet mask of the server - applicable for the interfaceName

lastUpdated  The time and date at which this server was last updated, in an ISO-like format

mgmtIpAddress  The IPv4 address of the server’s management port

mgmtIpGateway  The IPv4 gateway of the server’s management port

mgmtIpNetmask  The IPv4 subnet mask of the server’s management port

offlineReason  A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status (will be empty if not offline)

physLocation  The name of the physical location at which the server resides

physLocationId  An integral, unique identifier for the physical location at which the server resides

profile  The name of the profile assigned to this server

profileDesc  A description of the profile assigned to this server

\[2\]  See the Wikipedia article on Out-of-Band Management for more information.

---

### 5.1. Traffic Ops API

459
**profileId**  An integral, unique identifier for the profile assigned to this server

**rack** A string indicating “rack” location

**routerHostName** The human-readable name of the router

**routerPortName** The human-readable name of the router port

**status** The Status of the server

  See also:

  * Health Protocol*

**statusId** An integral, unique identifier for the status of the server

  See also:

  * Health Protocol*

**tcpPort** The default port on which the main application listens for incoming TCP connections - 80 in most cases

**type** The name of the type of this server

**typeId** An integral, unique identifier for the type of this server

**updPending** true if the server has updates pending, false otherwise

---

# 243: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: MaIvaO8OSjysr4bCkuXFEMf3o6mOqgalaM4IHN/tcP2aaliXEmA5IrHB7DaqNX/2vGHtLvN+0IFEAR/1RNqr1w==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:28:23 GMT
Content-Length: 891

{
    "response": [
        {
            "cachegroup": "CDN_in_a_Box_Edge",
            "cachegroupId": 7,
            "cdnId": 2,
            "cdnName": "CDN-in-a-Box",
            "domainName": "infra.ciab.test",
            "guid": null,
            "hostName": "edge",
            "httpsPort": 443,
            "id": 10,
```

(continues on next page)
deliveryservices/{{ID}}/servers/eligible

Caution: This endpoint may not work as advertised, and its use is therefore discouraged!

GET

Retrieves properties of Edge-Tier servers eligible for assignment to a particular Delivery Service.

Auth. Required Yes
Roles Required  “admin” or “operations”\(^1\)

Response Type  Array

Request Structure

Table 141: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery service for which servers will be displayed</td>
</tr>
</tbody>
</table>

Response Structure

- **cachegroup** The name of the Cache Group to which the server belongs
- **cachegroupId** An integral, unique identifier for the Cache Group to which the server belongs
- **cdnId** An integral, unique identifier the CDN to which the server belongs
- **cdnName** The name of the CDN to which the server belongs
- **domainName** The domain name part of the Fully Qualified Domain Name (FQDN) of the server
- **guid** Optionally represents an identifier used to uniquely identify the server
- **hostName** The (short) hostname of the server
- **httpsPort** The port on which the server listens for incoming HTTPS requests - 443 in most cases
- **id** An integral, unique identifier for the server
- **iloIpAddress** The IPv4 address of the lights-out-management port\(^2\)
- **iloIpGateway** The IPv4 gateway address of the lights-out-management port\(^2\)
- **iloIpNetmask** The IPv4 subnet mask of the lights-out-management port\(^2\)
- **iloPassword** The password of the of the lights-out-management user - displays as ****** unless the requesting user has the ‘admin’ role)\(^2\)
- **iloUsername** The user name for lights-out-management\(^2\)
- **interfaceMtu** The Maximum Transmission Unit (MTU) to configure for interfaceName

See also:

\(^1\) Users with the roles “admin” and/or “operations” will be able to the see servers not eligible for assignment to any given Delivery Service, whereas any other user will only be able to see the servers not eligible for assignment to a Delivery Service their Tenant is allowed to see. For this particular endpoint,

\(^2\) See the Wikipedia article on Out-of-Band Management for more information.
interfaceName  The network interface name used by the server
ip6Address   The IPv6 address and subnet mask of the server - applicable for the
interface interfaceName
ip6Gateway   The IPv6 gateway address of the server - applicable for the interface
interfaceName
ipAddress    The IPv4 address of the server- applicable for the interface
interfaceName
ipGateway    The IPv4 gateway of the server- applicable for the interface
interfaceName
ipNetmask    The IPv4 subnet mask of the server- applicable for the interface
interfaceName
lastUpdated  The time and date at which this server was last updated, in an ISO-like format
mgmtIpAddress The IPv4 address of the server’s management port
mgmtIpGateway The IPv4 gateway of the server’s management port
mgmtIpNetmask The IPv4 subnet mask of the server’s management port
offlineReason A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status (will be empty if not offline)
physLocation The name of the physical location at which the server resides
physLocationId An integral, unique identifier for the physical location at which the server resides
profile       The name of the profile assigned to this server
profileDesc   A description of the profile assigned to this server
profileId     An integral, unique identifier for the profile assigned to this server
rack          A string indicating “rack” location
routerHostName The human-readable name of the router
routerPortName The human-readable name of the router port
status        The Status of the server

See also:
Health Protocol

statusId      An integral, unique identifier for the status of the server

See also:
Health Protocol
tcpPort The default port on which the main application listens for incoming TCP connections - 80 in most cases

type The name of the type of this server

typeId An integral, unique identifier for the type of this server

updPending true if the server has updates pending, false otherwise

# 244: Response Example

```
{
    "response": [
        {
            "cachegroup": "CDN_in_a_Box_Edge",
            "cachegroupId": 7,
            "cdnId": 2,
            "cdnName": "CDN-in-a-Box",
            "domainName": "infra.ciab.test",
            "guid": null,
            "hostName": "edge",
            "httpsPort": 443,
            "id": 10,
            "iloIpAddress": "",
            "iloIpGateway": "",
            "iloIpNetmask": "",
            "iloPassword": "",
            "iloUsername": "",
            "interfaceName": "eth0",
            "interfaceMtu": 1500,
            "ipAddress": "172.16.239.100",
            "ipGateway": "172.16.239.1",
            "ipNetmask": "255.255.255.0",
            "lastUpdated": "2018-10-30 16:01:12+00",
            "mgmtIpAddress": "",
            "mgmtIpGateway": "",
            "mgmtIpNetmask": "",
            "offlineReason": "",
            "physLocation": "Apachecon North America 2018",
            "physLocationId": 1,
            "profile": "ATS_EDGE_TIER_CACHE",
            "profileDesc": "Edge Cache - Apache Traffic Server",
            "profileId": 9,
            "rack": "",
            "routerHostName": "",
            "routerPortName": "",
            "status": "REPORTED",
            "statusId": 3,
            "tcpPort": 80,
            "type": "EDGE",
            "typeId": 11,
        }
    ]
}  
```
"updPending": false
}
]

`deliveryservices/{{ID}}/state`

Deprecated since version 1.1: Use `cachegroup_fallbacks` instead to configure Cache Group fallbacks

**GET**

Retrieves the fail-over state for a Delivery Service.

Auth. Required Yes  
Roles Required None

Response Type Object

**Request Structure**

Table 142: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the Delivery Service being inspected</td>
</tr>
</tbody>
</table>

**Response Structure**

- **enabled**  true if failover has been enabled for this Delivery Service, false otherwise
- **failover** An object describing the failover configuration for this Delivery Service
  - **configured**  true if this failover configuration has been updated by some Traffic Ops user, false otherwise
  - **destination** An object describing the Cache Group within this Delivery Service which will utilize this failover configuration
    - **location** The integral, unique identifier of a Cache Group within this Delivery Service which will utilize this failover configuration

---

1 If a user does not have either the “admin” nor “operations” role, then only Delivery Services assigned to the user’s Tenant will be able to be queried with this endpoint
type The ‘type’ of the Cache Group identified by

enabled true if failover has been enabled for this Delivery Service, false otherwise

locations An array of integral, unique identifiers for Cache Groups to use for failover

# 245: Response Example

```
HTTP /1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 15 Nov 2018 14:54:17 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 18:54:17 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 6dswLNVARYBxXAQjXu8MfnLpQ94b9Hyrl7RoZhF2pw+7BotgU98zhQRQoQuPEwrTVrahTxUyxP2ic
Content-Length: 112

{ "response": {
   "failover": {
      "locations": [],
      "destination": null,
      "configured": false,
      "enabled": false
   },
   "enabled": false
}}
```

deliveryservices/{{ID}}/unassigned_servers

**Caution:** This route does not appear to work properly, and its use is strongly discouraged! Also note that the documentation here is not being updated as a result of this, and may contain out-of-date and/or erroneous information.

GET

Retrieves properties of Edge-tier servers not assigned to a Delivery Service.

**Auth. Required** Yes
Roles Required  “admin” or “operations”\(^1\)

Response Type  Array

Request Structure

Table 143: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Delivery service ID.</td>
</tr>
</tbody>
</table>

Response Structure

cachegroup The cache group name

cachegroupId The cache group id
cdnId Id of the CDN to which the server belongs to
cdnName Name of the CDN to which the server belongs to
domainName The domain name part of the FQDN of the cache
guid An identifier used to uniquely identify the server
hostName The host name part of the cache
httpsPort The HTTPS port on which the main application listens (443 in most cases)
id The server id (database row number)
iloIpAddress The IPv4 address of the lights-out-management port
iloIpGateway The IPv4 gateway address of the lights-out-management port
iloIpNetmask The IPv4 netmask of the lights-out-management port
iloPassword The password of the of the lights-out-management user (displays as ** unless you are an ‘admin’ user)
iloUsername The user name for lights-out-management
interfaceMtu The Maximum Transmission Unit (MTU) to configure for interfaceName

interfaceName The network interface name used for serving traffic
ip6Address The IPv6 address/netmask for interfaceName
ip6Gateway The IPv6 gateway for interfaceName

\(^1\) Users with the roles “admin” and/or “operations” will be able to see servers not assigned to any given Delivery Service, whereas any other user will only be able to see the servers not assigned to Delivery Services their Tenant is allowed to see.
ipAddress  The IPv4 address for interfaceName
ipGateway  The IPv4 gateway for interfaceName
ipNetmask  The IPv4 netmask for interfaceName
lastUpdated  The Time and Date for the last update for this server
mgmtIpAddress  The IPv4 address of the management port (optional
mgmtIpGateway  The IPv4 gateway of the management port (optional
mgmtIpNetmask  The IPv4 netmask of the management port (optional
offlineReason  A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status
physLocation  The physical location name
physLocationId  The physical location id
profile  The assigned profile name
profileDesc  The assigned profile description
profileId  The assigned profile Id
rack  A string indicating rack location
routerHostName  The human readable name of the router
routerPortName  The human readable name of the router port
status  The Status string
statusId  The Status id
tcpPort  The default TCP port on which the main application listens (80 for a cache in most cases
type  The name of the type of this server
typeId  The id of the type of this server
updPending  bool

# 246: Response Example

```json
{
    "response": [
      {
        "cachegroup": "us-il-chicago",
        "cachegroupId": "3",
        "cdnId": "3",
        "cdnName": "CDN-1",
        "domainName": "chi.kabletown.net",
        "guid": null,
      }
    ]
}
```
(continues on next page)
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hostName</code></td>
<td><code>atsec-chi-00</code></td>
</tr>
<tr>
<td><code>id</code></td>
<td><code>19</code></td>
</tr>
<tr>
<td><code>iloIpAddress</code></td>
<td></td>
</tr>
<tr>
<td><code>iloIpGateway</code></td>
<td></td>
</tr>
<tr>
<td><code>iloIpNetmask</code></td>
<td></td>
</tr>
<tr>
<td><code>iloPassword</code></td>
<td><code>********</code></td>
</tr>
<tr>
<td><code>iloUsername</code></td>
<td><code>&quot;&quot;</code></td>
</tr>
<tr>
<td><code>interfaceMtu</code></td>
<td><code>9000</code></td>
</tr>
<tr>
<td><code>interfaceName</code></td>
<td><code>bond0</code></td>
</tr>
<tr>
<td><code>ipAddress</code></td>
<td><code>10.10.2.2</code></td>
</tr>
<tr>
<td><code>ipGateway</code></td>
<td><code>10.10.2.1</code></td>
</tr>
<tr>
<td><code>ipNetmask</code></td>
<td><code>255.255.0.0</code></td>
</tr>
<tr>
<td><code>lastUpdated</code></td>
<td><code>2015-03-08 15:57:32</code></td>
</tr>
<tr>
<td><code>physLocation</code></td>
<td><code>plocation-chi-1</code></td>
</tr>
<tr>
<td><code>physLocationId</code></td>
<td><code>9</code></td>
</tr>
<tr>
<td><code>profile</code></td>
<td><code>EDGE1_CDNL_421_SSL</code></td>
</tr>
<tr>
<td><code>profileDesc</code></td>
<td><code>EDGE1_CDNL_421_SSL profile</code></td>
</tr>
<tr>
<td><code>rack</code></td>
<td><code>RR 119.02</code></td>
</tr>
<tr>
<td><code>routerHostName</code></td>
<td><code>rtr-chi.kabletown.net</code></td>
</tr>
<tr>
<td><code>routerPortName</code></td>
<td><code>2</code></td>
</tr>
<tr>
<td><code>status</code></td>
<td><code>ONLINE</code></td>
</tr>
<tr>
<td><code>statusId</code></td>
<td><code>6</code></td>
</tr>
</tbody>
</table>

(continues on next page)
deliveryservices_regexes

GET

Retrieves routing regular expressions for all *Delivery Services*.

Auth. Required  Yes

Roles Required  None¹

Response Type  Array

Request Structure

No parameters available

Response Structure

**dsName**  The name of the *Delivery Service* represented by this object

**regexes**  An array of objects that represent various routing regular expressions used by dsName

- **pattern**  The actual regular expression - \s are escaped

- **setNumber**  The order in which the regular expression is evaluated against requests

- **type**  The type of regular expression - determines that against which it will be evaluated

# 247: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
```

¹ If tenancy is used, then users (regardless of role) will only be able to see the routing regular expressions used by *Delivery Services* their tenant has permissions to see.
deliveryservices/request

See also:

Delivery Service Requests

POST

Allows a user to submit Delivery Service Requests.

Auth. Required  Yes
Roles Required  None
Response Type  undefined

Request Structure

details  An object describing the actual parameters for the Delivery Service request
customer  Name of the customer associated with the Delivery Service
deepCachingType  An optional string describing when to do Deep Caching for this Delivery Service - one of:
NEVER  Never use deep caching (default)

ALWAYS  Always use deep caching

deliveryProtocol  The protocol used to retrieve content from the CDN - one of:
   • http
   • https
   • http/https

hasNegativeCachingCustomization  true if any customization is required for negative caching, false otherwise

hasOriginACLWhitelist  true if access to the origin is restricted using an Access Control List (ACL or “whitelist”) of IP addresses

hasOriginDynamicRemap  If true, this Delivery Service can dynamically map to multiple origin URLs

hasSignedURLs  If true, this Delivery Service’s URLs are signed

headerRewriteEdge  An optional string containing a header re-write rule to be used at the Edge tier

headerRewriteMid  An optional string containing a header re-write rule to be used at the Mid tier

header RewriteRedirectRouter  An optional string containing a header re-write rule to be used by the Traffic Router

maxLibrarySizeEstimate  A special string that describes the estimated size of the sum total of content available through this Delivery Service

negativeCachingCustomizationNote  A note remarking on the use, customization, or complications associated with negative caching for this Delivery Service

notes  An optional string containing additional instructions or notes regarding the Request

originHeaders  An optional, comma-separated string of header values that must be passed to requests to the Delivery Service’s origin

originTestFile  A URL path to a test file available on the Delivery Service’s origin server

originURL  The URL of the Delivery Service’s origin server

otherOriginSecurity  An optional string describing any and all other origin security measures that need to be considered
for access to the Delivery Service’s origin

**overflowService** An optional string containing the IP address or URL of an overflow point (used if rate limits are met or exceeded)

**peakBPSEstimate** A special string describing the estimated peak data transfer rate of the Delivery Service in Bytes Per Second (BPS)

**peakTPSEstimate** A special string describing the estimated peak transaction rate of the Delivery Service in Transactions Per Second (TPS)

**queryStringHandling** A special string describing how the Delivery Service should treat URLs containing query parameters

**rangeRequestHandling** A special string describing how the Delivery Service should handle range requests

**rateLimitingGBPS** An optional field which, if defined, should contain the maximum allowed data transfer rate for the Delivery Service in GigaBytes Per Second (GBPS)

**rateLimitingTPS** An optional field which, if defined, should contain the maximum allowed transaction rate for the Delivery Service in Transactions Per Second (TPS)

**routingName** The routing name for the Delivery Service, e.g. SomeRoutingName.DeliveryService_xml_id.CDNName.com

**routingType** The Delivery Service’s routing type, should be one of:

- **HTTP** The Traffic Router re-directs clients to cache servers using the HTTP 302 REDIRECT response code
- **DNS** The Traffic Router responds to requests for name resolution of the Delivery Service’s routing name with IP addresses of cache servers
- **STEERING** This Delivery Service routes clients to other Delivery Services - which will in turn (generally) route them to clients
- **ANY_MAP** Some kind of undocumented black magic is used to get clients to... content, probably?

**serviceAliases** An optional array of aliases for this Delivery Service

**serviceDesc** A description of the Delivery Service
**emailTo** The email to which the Delivery Service request will be sent

# 248: Request Example

```json
{
  "emailTo": "foo@bar.com",
  "details": {
    "customer": "XYZ Corporation",
    "contentType": "static",
    "deepCachingType": "NEVER",
    "deliveryProtocol": "http",
    "routingType": "http",
    "routingName": "demo1",
    "serviceDesc": "service description goes here",
    "peakBPSEstimate": "less-than-5-Gbps",
    "peakTPSEstimate": "less-than-1000-TPS",
    "maxLibrarySizeEstimate": "less-than-200-GB",
    "originURL": "http://myorigin.com",
    "hasOriginDynamicRemap": false,
    "originTestFile": "http://origin.infra.ciab.test",
    "hasOriginACLWhitelist": false,
    "originHeaders": "",
    "otherOriginSecurity": "",
    "queryStringHandling": "ignore-in-cache-key-and-pass-up",
    "rangeRequestHandling": "range-requests-not-used",
    "hasSignedURLs": false,
    "hasNegativeCachingCustomization": false,
    "negativeCachingCustomizationNote": "",
    "serviceAliases": [],
    "rateLimitingGBPS": 50,
    "rateLimitingTPS": 5000,
    "overflowService": null,
    "headerRewriteEdge": "",
    "headerRewriteMid": "",
    "headerRewriteRedirectRouter": "",
    "notes": ""
  }
}
```

Response Structure
# 249: Response Example

```
{ "alerts": [{
    "level": "success",
    "text": "Delivery Service request sent to foo@bar.com."
}]
}
```

deliveryservices/sslkeys/add

See also:

In most cases it is preferable to allow Traffic Ops to generate the keys via `deliveryservices/sslkeys/generate`, rather than uploading them manually using this endpoint.

**POST**

Allows user to upload an SSL certificate, csr, and private key for a Delivery Service.

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object (string)

**Request Structure**

- **cdn** The name of the CDN to which the Delivery Service belongs
- **certificate** An object that contains the actual components of the SSL key
  - **crt** The certificate for the Delivery Service identified by **key**
  - **csr** The csr file for the Delivery Service identified by **key**
- **key** The private key for the Delivery Service identified by **key**
- **key** The ‘xml_id’ of the Delivery Service to which these keys will be assigned
- **version** An integer that defines the “version” of the key - which may be thought of as the sequential generation; that is, the higher the number the more recent the key

# 250: Request Example

```
POST /api/1.4/deliveryservices/sslkeys/add HTTP/1.1
Host: trafficops.infra.ciab.test
Content-Type: application/json

{
    "key": "ds-01",

    "alerts": [{
        "level": "success",
        "text": "Delivery Service request sent to foo@bar.com."
    }]
}
```

"version": "1",
"certificate": {
  "key": "some_key",
  "csr": "some_csr",
  "crt": "some_crt"
}
}

Response Structure

# 251: Response Example

```plaintext
HTTP/1.1 200 OK
Content-Type: application/json

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

deliveryservices/sslkeys/generate

POST

Generated an SSL certificate, csr, and private key for a Delivery Service

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object (string)

Request Structure

- **city** An optional field which, if present, will represent the resident city of the generated SSL certificate
- **country** An optional field which, if present, will represent the resident country of the generated SSL certificate
- **hostname** The desired hostname of the Delivery Service

Note: In most cases, this must be the same as the Delivery Service URL'

- **key** The ‘xml_id’ of the Delivery Service for which keys will be generated
organization An optional field which, if present, will represent the organization for which the SSL certificate was generated

state An optional field which, if present, will represent the resident state or province of the generated SSL certificate

businessUnit An optional field which, if present, will represent the business unit for which the SSL certificate was generated

version version of the keys being generated

# 252: Request Example

```
POST /api/1.3/deliveryservices/sslkeys/generate HTTP/1.1
Content-Type: application/json

{
    "key": "ds-01",
    "businessUnit": "CDN Engineering",
    "version": "3",
    "hostname": "tr.ds-01.ott.kabletown.com",
    "country": "US",
    "organization": "Kabletown",
    "city": "Denver",
    "state": "Colorado"
}
```

Response Structure

# 253: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json

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

deliveryservices/{{xml_id}}/servers

Deprecated since version 1.1: Use deliveryserviceserver instead

POST

Assigns cache servers to a Delivery Service.

Auth. Required Yes

Roles Required “admin” or “operations”

1 Users with the roles “admin” and/or “operation” will be able to edit the server assignments of all Delivery Services.
Response Type  Object

Request Structure

Table 144: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>The ‘xml_id’ of the Delivery Service whose server assignments are being edited</td>
</tr>
</tbody>
</table>

**serverNames** An array of hostname of cache servers to assign to this Delivery Service

# 254: Request Example

```plaintext
POST /api/1.4/deliveryservices/test/servers HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 24
Content-Type: application/json

{ "serverNames": [ "edge" ] }
```

Response Structure

**xml_id** The ‘xml_id’ of the Delivery Service to which the servers in serverNames have been assigned

**serverNames** An array of hostnames of cache servers assigned to Delivery Service identified by xml_id

# 255: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: zTpLrWiLM4xRsm8m1BQFB5Kz478Ajl0SyXHgtyWhebCv1YIwWltmkjr0HFgc3GGMZODt+fyzkOYy5
X-Server-Name: traffic_ops_golang/
```

(continues on next page)

Services, whereas any other user will only be able to edit the server assignments of the Delivery Services their Tenant is allowed to edit.
deliveryservices/{{xml_id}}/urisignkeys

DELETE

Deletes URISigning objects for a delivery service.
Authentication Required: Yes
Role(s) Required: admin

Request Route Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>yes</td>
<td>xml_id of the desired delivery service</td>
</tr>
</tbody>
</table>

GET deliveryservices/:xml_id/urisignkeys

Retrieves one or more URISigning objects for a delivery service.
Authentication Required: Yes
Role(s) Required: admin

Request Route Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>yes</td>
<td>xml_id of the desired delivery service</td>
</tr>
</tbody>
</table>

Response Properties
### Parameter Type Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>string</td>
<td>a string describing the issuer of the URI signing object. Multiple URISigning objects may be returned in a response, see example</td>
</tr>
<tr>
<td>renewal_kid</td>
<td>string</td>
<td>a string naming the jwt key used for renewals.</td>
</tr>
<tr>
<td>keys</td>
<td>string</td>
<td>json array of jwt symmetric keys.</td>
</tr>
<tr>
<td>alg</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the jwa encryption algorithm to use with this key, <a href="https://tools.ietf.org/html/rfc7518">RFC 7518</a></td>
</tr>
<tr>
<td>kid</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the unique id for the key as defined in <a href="https://tools.ietf.org/html/rfc7516">RFC 7516</a></td>
</tr>
<tr>
<td>kty</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the key type as defined in <a href="https://tools.ietf.org/html/rfc7516">RFC 7516</a></td>
</tr>
<tr>
<td>k</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the base64 encoded symmetric key see <a href="https://tools.ietf.org/html/rfc7516">RFC 7516</a></td>
</tr>
</tbody>
</table>

### Response Example

```json
{
    "Kabletown URI Authority": {
        "renewal_kid": "Second Key",
        "keys": [
            {
                "alg": "HS256",
                "kid": "First Key",
                "kty": "oct",
                "k": "Kh_RkUMj-fzbD37qBnDf_3e_RvQ3RP9PaSmVEpE24AM"
            },
            {
                "alg": "HS256",
                "kid": "Second Key",
                "kty": "oct",
                "k": "fZBpDBNbk2GqhwoB_3e_DGBAsBxgQZVix04rIoLj7p_RlE"
            }
        ]
    }
}
```

**POST deliveryservices/:xml_id/urisignkeys**

Assigns URISigning objects to a delivery service.

Authentication Required: Yes

Role(s) Required: admin

**Request Route Parameters**
<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>yes</td>
<td>xml_id of the desired delivery service</td>
</tr>
</tbody>
</table>

### Request Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>string</td>
<td>a string describing the issuer of the URI signing object. Multiple URISigning objects may be returned in a response. See example</td>
</tr>
<tr>
<td>renewal_kid</td>
<td>string</td>
<td>a string naming the jwt key used for renewals.</td>
</tr>
<tr>
<td>keys</td>
<td>string array of jwt symmetric keys</td>
<td></td>
</tr>
<tr>
<td>alg</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the jwa encryption algorithm to use with this key. RFC 7518</td>
</tr>
<tr>
<td>kid</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the unique id for the key as defined in RFC 7516</td>
</tr>
<tr>
<td>kty</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the key type as defined in RFC 7516</td>
</tr>
<tr>
<td>k</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the base64 encoded symmetric key see RFC 7516</td>
</tr>
</tbody>
</table>

### Request Example

```json
{
    "Kabletown URI Authority": {
        "renewal_kid": "Second Key",
        "keys": [
            {
                "alg": "HS256",
                "kid": "First Key",
                "kty": "oct",
                "k": "Kh_RkUMj-fzbD37qBnDf_3e_RvQ3RP9PaSmVEpE24AM"
            },
            {
                "alg": "HS256",
                "kid": "Second Key",
                "kty": "oct",
                "k": "f2BpDnBk2GqwoB_DGBAsBxqQ2Vix04riOLJ7p_R1E"
            }
        ]
    }
}
```

**PUT deliveryservices/:xml_id/urisignkeys**

updates URISigning objects on a delivery service.
Authentication Required: Yes
Role(s) Required: admin

Request Route Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>yes</td>
<td>xml_id of the desired delivery service</td>
</tr>
</tbody>
</table>

Request Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>string</td>
<td>a string describing the issuer of the URI signing object. Multiple URISigning objects may be returned in a response, see example</td>
</tr>
<tr>
<td>renewal_kid</td>
<td>string</td>
<td>a string naming the jwt key used for renewals.</td>
</tr>
<tr>
<td>keys</td>
<td>string</td>
<td>json array of jwt symmetric keys</td>
</tr>
<tr>
<td>alg</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the jwa encryption algorithm to use with this key, RFC 7518</td>
</tr>
<tr>
<td>kid</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the unique id for the key as defined in RFC 7516</td>
</tr>
<tr>
<td>kty</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the key type as defined in RFC 7516</td>
</tr>
<tr>
<td>k</td>
<td>string</td>
<td>this parameter repeats for each jwt key in the array and specifies the base64 encoded symmetric key see RFC 7516</td>
</tr>
</tbody>
</table>

Request Example

```json
{
   "Kabletown URI Authority": {
      "renewal_kid": "Second Key",
      "keys": [
         {
            "alg": "HS256",
            "kid": "First Key",
            "kty": "oct",
            "k": "Kh_RkUMj-fzbD37qBnDf_
         },
         {
            "alg": "HS256",
            "kid": "Second Key",
            "kty": "oct",
            "k": "fZBpDBNbk2GqhwoB_
         }
      ]
   }
}
```

(continues on next page)
deliveryservices/xmlId/{{XMLID}}/sslkeys

GET

Retrieves SSL keys for a Delivery Service.

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

Table 145: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLID</td>
<td>The ‘xml_id’ of the desired Delivery Service</td>
</tr>
</tbody>
</table>

Table 146: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>no</td>
<td>The version number of the SSL keys to retrieve</td>
</tr>
<tr>
<td>decode</td>
<td>no</td>
<td>If true, the returned keys will be decoded - if false, they will not be decoded</td>
</tr>
</tbody>
</table>

Caution: There’s almost certainly no good reason to request the private key! Even when “base 64-encoded” do not let ANYONE see this who would be unable to request it themselves!

Response Structure

Changed in version 1.2: Grouped the crt, csr, and key fields under certificate.

5.1. Traffic Ops API 483
**businessUnit** An optional field which, if present, contains the business unit entered by the user when generating the SSL certificate.

**certificate** An object containing the actual generated key, certificate, and signature of the SSL keys

- **crt** Base 64-encoded (or not if the `decode` query parameter was given and `true`) certificate for the Delivery Service identified by `deliveryservice`.

- **csr** Base 64-encoded (or not if the `decode` query parameter was given and `true`) csr file for the Delivery Service identified by `deliveryservice`.

- **key** Base 64-encoded (or not if the `decode` query parameter was given and `true`) private key for the Delivery Service identified by `deliveryservice`.

Caution: There’s almost certainly no good reason to request the private key! Even when “base 64-encoded” do not let ANYONE see this who would be unable to request it themselves!

**cdn** The CDN of the Delivery Service for which the certs were generated.

**city** An optional field which, if present, contains the city entered by the user when generating the SSL certificate.

**country** An optional field which, if present, contains the country entered by the user when generating the SSL certificate.

**deliveryservice** The ‘xml_id’ of the Delivery Service for which the certificate was generated.

**hostname** The hostname generated by Traffic Ops that is used as the common name when generating the certificate - this will be a FQDN for DNS Delivery Services and a wildcard URL for HTTP Delivery Services.

**organization** An optional field which, if present, contains the organization entered by the user when generating certificate.

**state** An optional field which, if present, contains the state entered by the user when generating certificate.

**version** The version of the certificate record in Riak.

deliveryservices/xmlId/{{xmlid}}/sslkeys/delete

---

1 These optional fields will be present in the response if and only if they were specified during key generation; they are optional during key generation and thus cannot be guaranteed to exist or not exist.
GET

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object (string)

Request Structure

Table 147: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlId</td>
<td>yes</td>
<td>The ‘xml_id’ of the desired Delivery Service</td>
</tr>
</tbody>
</table>

Table 148: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>no</td>
<td>The version number of the SSL keys that shall be retrieved</td>
</tr>
</tbody>
</table>

Response Structure

# 256: Response Example

```json
{ "response": "Successfully deleted ssl keys for <xml_id>" }
```

deliveryservices/xmlId/{{xmlid}}/urlkeys

New in version 1.2.

GET

Retrieves URL signing keys for a Delivery Service.

Caution: This method will return the Delivery Service’s PRIVATE URL signing keys! Be wary of using this endpoint and NEVER share the output with anyone who would be unable to see it on their own.

Auth. Required  Yes
Roles Required  None
Response Type  Object
Request Structure

Table 149: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlid</td>
<td>The ‘xml_id’ of the desired Delivery Service</td>
</tr>
</tbody>
</table>

Response Structure

key<N> The private URL signing key for this Delivery Service as a base-64-encoded string, where <N> is the “generation” of the key e.g. the first key will always be named "key0". Up to 16 concurrent generations are retained at any time (<N> is always on the interval [0,16])

# 257: Response Example

```json
{
  "response": {
    "key9": "ZvVQNYpPVQWQV8tjQnU16osm4y7xK4zD",
    "key6": "JhGdpw5X9o8TqHfgezCm0bqb9QPAASL",
    "key8": "yXSdp1T8IeDEEE1CMftzZb9EIw_20wq",
    "key0": "D4AYzJ1AE2nYisA9MwMr03TP2Hji9C",
    "key3": "W90YH1Gc_kY1yw5_10LrkpV9J0zSlneI",
    "key12": "ZbtMb3mrKqfs8hx9_xWBIP_0PWMlUpzc",
    "key2": "0gqEoD07sUsugIQemZbw7mT0tNCwB1sf",
    "key4": "aFJ2Gb7atmxVB8uv7T9S60aDl3ycpGf",
    "key1": "wnWNRlMClz104C7FPtcqHd0xUMQyNFhA",
    "key11": "k6HMz1BHx6htKkypRFfWQhAndQqe50e",
    "key10": "zYONfdD7fGYKj4kLwIj4U0918csuZO0d",
    "key15": "3360cGaiq_layZMc_QhI2teJbaxzTQh",
    "key5": "SIwv3GOhWN7EE9wSwPfj18qE4M07sFxn",
    "key13": "SgKBR6LqE0z8aewZUCVtBcW_8YFclg",
    "key14": "Dtxau8nsw04Yh70kNkBhu2G3P9WrpQJ",
    "key7": "cmKoiIXxGxAxUMdCsWvnGLoIMGmNiuT5I"
}
```

deliveryservices/xmlId/{{xml_id}}/urlkeys/copyFromXmlId/
{{copyFrom_xml_id}}

POST

Allows a user to copy URL signing keys from a specified Delivery Service to another Delivery Service.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object (string)
Request Structure

Table 150: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>The ‘xml_id’ of the <em>Delivery Service</em> to which keys will be copied</td>
</tr>
<tr>
<td>copyFrom_xml_id</td>
<td>The ‘xml_id’ of the <em>Delivery Service from</em> which keys will be copied</td>
</tr>
</tbody>
</table>

Response Structure

# 258: Response Example

```
{
    "response": "Successfully copied and stored keys"
}
```

deliveryservices/xmlId/{{xml_id}}/urlkeys/generate

New in version 1.2.

**POST**

Generates URL signing keys for a *Delivery Service*

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object (string)

Request Structure

Table 151: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_id</td>
<td>The ‘xml_id’ of the desired <em>Delivery Service</em></td>
</tr>
</tbody>
</table>

Response Structure

# 259: Response Example

```
{
    "response": "Successfully generated and stored keys"
}
```
deliveryserviceserver

GET

Retrieve information about the assignment of servers to *Delivery Services*

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

### Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>no</td>
<td>0</td>
<td>The page number for use in pagination - 0 means “no pagination”</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>20</td>
<td>Limits the results to a maximum of this number - if pagination is used, this defines the number of results per page</td>
</tr>
<tr>
<td>orderby</td>
<td>no</td>
<td>“deliveryservice”</td>
<td>Choose the ordering of the results - must be the name of one of the fields of the objects in the response array</td>
</tr>
</tbody>
</table>

#### # 260: Request Example

```
GET /api/1.1/deliveryserviceserver?page=1&limit=2&orderby=lastUpdated HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

Unlike most API endpoints, this will return a JSON response body containing both a “response” object as well as other, top-level fields (besides the optional “alerts” field). For this reason, this section contains a “response” key, which normally is implicit.

**See also:**

*Response Structure*

---

1 While no roles are required, this endpoint *does* respect tenancy permissions (pending GitHub Issue #2978).
limit The maximum size of the response array, also indicative of the number of results per page using the pagination requested by the query parameters (if any) - this should be the same as the limit query parameter (if given)

orderby A string that names the field by which the elements of the response array are ordered - should be the same as the orderby request query parameter (if given)

response An array of objects, each of which represents a server’s Delivery Service assignment

deliveryService The integral, unique identifier of the Delivery Service to which the server identified by server is assigned

lastUpdated The date and time at which the server’s assignment to a Delivery Service was last updated

server The integral, unique identifier of a server which is assigned to the Delivery Service identified by deliveryService

size The page number - if pagination was requested in the query parameters, else 0 to indicate no pagination - of the results represented by the response array. This is named “size” for legacy reasons

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: J7sK8PohQWyTpTrMjjrWdlJwPj+Zyep/xutM25uVosL6cHgi30nXa6VMyOC5Y3vd9r5KLES8rTgR+qUQcZcJ/A==
X-Server-Name: traffic_ops_golang/
Date: Thu, 01 Nov 2018 14:27:45 GMT
Content-Length: 129

{  "orderby": "lastUpdated",
  "response": [
    {
      "server": 8,
      "deliveryService": 1,
      "lastUpdated": "2018-11-01 14:10:38+00"
    }
  ],
  "size": 1,
  "limit": 2
}
POST

Assign a set of one or more servers to a Delivery Service

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

deliveryService  The integral, unique identifier of the Delivery Service to which the servers identified in the servers array will be assigned
replace  If true, any existing assignments for a server identified in the servers array will be overwritten by this request
servers  An array of integral, unique identifiers for servers which are to be assigned to the Delivery Service identified by deliveryService

# 262: Request Example

```
POST /api/1.1/deliveryserviceserver HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 46
Content-Type: application/x-www-form-urlencoded
dsId=1&replace=true&servers=12
```

Response Structure

deliveryService  The integral, unique identifier of the Delivery Service to which the servers identified by the elements of the servers array have been assigned
replace  If true, any existing assignments for a server identified in the servers array have been overwritten by this request
servers  An array of integral, unique identifiers for servers which have been assigned to the Delivery Service identified by deliveryService

2 Users with the “admin” or “operations” roles will be able to modify ALL server-to-Delivery-Service assignments, whereas all other users can only assign servers to the Delivery Services their Tenant has permissions to edit.
# 263: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: D+HhGhoxzaxvka9v2IStoaO2UpX23nz7zZnMbpFHNRO3MawyEaSb3GVUHQyCv6sDgwhp2ZjRggDmct
X-Server-Name: traffic_ops_golang/
Date: Thu, 01 Nov 2018 14:12:49 GMT
Content-Length: 123

{
  "alerts": [
    {
      "text": "server assignements complete",
      "level": "success"
    }
  ],
  "response": {
    "dsId": 1,
    "replace": false,
    "servers": [ 12 ]
  }
}
```

divisions

GET

Returns a JSON representation of all configured Divisions.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Filter for divisions having this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>Filter for divisions with this name</td>
</tr>
</tbody>
</table>
Response Structure

- **id** An integral, unique identifier for this Division
- **lastUpdated** The date and time at which this Division was last modified, in ISO format
- **name** The Division name

# 264: Response Example

```json
{  
  "response": [  
    {  
      "id": 1,  
      "lastUpdated": "2018-11-29 18:38:28+00",  
      "name": "Quebec"  
    },  
    {  
      "id": 2,  
      "lastUpdated": "2018-11-29 18:38:28+00",  
      "name": "USA"  
    }  
  ]
}
```

**POST**

Creates a new Division.

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object
Request Structure

**name**  The name of the new Division

# 265: Request Example

```
POST /api/1.4/divisions HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 16
Content-Type: application/json

{"name": "test"}
```

Response Structure

**id**  An integral, unique identifier for this Division

**lastUpdated**  The date and time at which this Division was last modified, in ISO format

**name**  The Division name

# 266: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: +pJm4c3O+JTaSXNt+LP+u240Ba/SSvSSD0Q4rDc6hcY20FII+iY/WWrMHhpLu1RGKY88bM4YPCMaxGn3FZ9yQ==
X-Server-Name: traffic_ops_golang/
Date: Thu, 29 Nov 2018 19:52:06 GMT
Content-Length: 136

{ "alerts": [ 
  
  "text": "division was created.",
  "level": "success"

  ]

},
"response": {
  "id": 3,
  "lastUpdated": "2018-11-29 19:52:06+00",
```
"name": "test"
}}

divisions/{{ID}}

GET

Get a specific Division.

Deprecated since version 1.1: Use the id query parameter of the GET method of divisions instead.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the requested Division</td>
</tr>
</tbody>
</table>

# 267: Request Example

```plaintext
GET /api/1.4/divisions/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>An integral, unique identifier for this Division</td>
</tr>
<tr>
<td>lastUpdated</td>
<td>The date and time at which this Division was last modified, in ISO format</td>
</tr>
<tr>
<td>name</td>
<td>The Division name</td>
</tr>
</tbody>
</table>

# 268: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
```
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: JTBi9pskjuUA+MSex60beWKE/GyIuRNVy2Yxo6AVe+XnFyvvC3iEVXZkmjiSXg20UXGeSCKAlLCfouQFSS3A==
X-Server-Name: traffic_ops_golang/
Date: Thu, 29 Nov 2018 19:59:57 GMT
Content-Length: 78

```json
{ "response": [
  {
    "id": 1,
    "lastUpdated": "2018-11-29 18:38:28+00",
    "name": "Quebec"
  }
]}
```

**PUT**

Updates a specific Division

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Request Structure**

Table 155: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the requested Division</td>
</tr>
</tbody>
</table>

name The new name of the Division

# 269: Request Example

```bash
PUT /api/1.4/divisions/3 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=
Content-Length: 17
Content-Type: application/json
```
Response Structure

- **id** An integral, unique identifier for this Division
- **lastUpdated** The date and time at which this Division was last modified, in ISO format
- **name** The Division name

### # 270: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: yBd8MzBR/Qbc/xts44WEIFRTqeMKZwUe2ufpm6JH6frh1UjFmYRs3/
B7E5FTruFWRTrvEI1x5EpDmp3f9LjzA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 29 Nov 2018 20:10:36 GMT
Content-Length: 137

{ "alerts": [ 
    { 
      "text": "division was updated."
    },
    "level": "success"
  ],
"response": {
  "id": 3,
  "lastUpdated": "2018-11-29 20:10:36+00",
  "name": "quest"
}}
```

`divisions/{{name}}/regions`

Deprecated since version 1.1: Use the `divisionId` field in the body of a POST request to the `divisions` endpoint
POST

Creates a new region within the specified division.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the division in which to create the new region</td>
</tr>
</tbody>
</table>

name  The name of the new region

# 271: Request Example

```json
POST /api/1.4/divisions/England/regions HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 27
Content-Type: application/json

{
    "name": "Greater_London",
}
```

Response Structure

divisionName  The name of the division which contains the new region

divisionId    The integral, unique identifier of the division which contains the new region

id    An integral, unique identifier for this region

name    The region name

# 272: Response Example

**HTTP/1.1 200 OK**
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-Within, Content-Type, Accept

(continues on next page)
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 06 Dec 2018 00:03:36 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 06 Dec 2018 04:03:36 GMT;
Vary: Accept-Encoding
Whole-Content-Sha512: HlzhY411FBRlLe5D0XN1w+LbU/
Content-Length: 84

{
  "response": {
    "divisionName": "England",
    "divisionId": 3,
    "name": "Greater_London",
    "id": 3
  }
}

federation_resolvers

POST

Creates a new federation resolver.

Auth. Required  Yes
Roles Required  “admin”
Response Type  Object

Request Structure

ipAddress  The IP address of the resolver - may be IPv4 or IPv6

TypeId  The integral, unique identifier of the type of resolver being created - will
        represent one of:
        RESOLVE4  Resolver is for IPv4 addresses and ipAddress is IPv4
        RESOLVE6  Resolver is for IPv6 addresses and ipAddress is IPv6

# 273: Request Example

POST /api/1.4/federation_resolvers HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 39
Content-Type: application/json

{
    "ipAddress": "0.0.0.0",
    "typeId": 35
}

Response Structure

- **id** The integral, unique identifier of the resolver
- **ipAddress** The IP address of the resolver - may be IPv4 or IPv6
- **type** The type of the resolver - one of:
  - **RESOLVE4** Resolver is for IPv4 addresses and ipAddress is IPv4
  - **RESOLVE6** Resolver is for IPv6 addresses and ipAddress is IPv6

# 274: Response Example

HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type,
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 00:41:42 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:41:42 GMT;

time: Wed, 05 Dec 2018 00:41:42 GMT
whole-content-sha512: JaXUP+onw0hG5s+H/
w7w2bNm9a7bgLGDGJRutF8yT0DBAfNR+x7NZ4a0w+5w3RyDj1l1h5zLadQeEcd2j8vwy=
content-length: 151

{
    "alerts": [
        {
            "level": "success",
            "text": "Federation Resolver created [ IP = 0.0.0.0 with id: 1"
        }
    ],
    "response": {

(continues on next page)
federation_resolvers/{{ID}}

DELETE

Deletes a federation resolver.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** undefined

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integral, unique identifier for the federation resolver to be deleted</td>
</tr>
</tbody>
</table>

# 275: Request Example

```bash
DELETE /api/1.4/federation_resolvers/3 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 276: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content- Type, Accept
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
ccontent-type: application/json
date: Wed, 05 Dec 2018 01:06:51 GMT
```
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 05:06:51 GMT;
  path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512: NqAZuZYlF1UWOazbj/
j4gWX7ye0kGGakRRFekK6ShxqXvCxO0dCTyu75qiLPN2wSgr3FGQnp2Sq345sE7In9g==
content-length: 98

{ "alerts": [ 
  { 
    "level": "success",
    "text": "Federation resolver deleted [ IP = ::1/128 with id: 3"
  }
]}

federations

GET

Retrieves a list of federation mappings (aka federation resolvers) for the current user.

  **Auth. Required** Yes  
  **Roles Required** “admin”, “Federation”, “operations”, “Portal”, or “Steering”
  **Response Type** Array

Request Structure

No parameters available.

Response Structure

  **deliveryService** The xml_id that uniquely identifies the Delivery Service that uses the federation mappings in mappings

  **mappings** An array of objects that represent the mapping of a federation’s Canonical Name (CNAME) to one or more resolvers

   **cname** The actual CNAME used by the federation

   **resolve4** An array of IPv4 addresses capable of resolving the federation’s CNAME

   **resolve6** An array of IPv6 addresses capable of resolving the federation’s CNAME

5.1. Traffic Ops API
**ttl** The Time To Live (TTL) of the CNAME in hours

---

### # 277: Response Example

```http
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: ...
X-Server-Name: traffic_ops_golang/
Date: Mon, 03 Dec 2018 17:19:13 GMT
Content-Length: 136

```{ "response": [
    {
      "mappings": [
        {
          "ttl": 300,
          "cname": "blah.blah.",
          "resolve4": [ "0.0.0.0/32"
          ],
          "resolve6": [ "::/128"
          ]
        }
      ],
      "deliveryService": "demo1"
    }
  ]
}
```

**POST**

Allows a user to create federation resolvers for Delivery Services, providing the Delivery Service is within a CDN that has some associated federation.

**Warning:** Confusingly, this endpoint does not create a new federation; to do that, the `cdns/{{name}}/federations` endpoint must be used. Furthermore, the federation must properly be assigned to a Delivery Service using the `federations/{{ID}}/deliveryservices` and assigned to the user creating resolvers using `federations/{{ID}}/users`.

See also:
The *federations/{{ID}}/federation_resolvers* endpoint duplicates this functionality.

**Auth. Required** Yes

**Roles Required** “admin”, “Federation”, “operations”, “Portal”, or “Steering”

**Response Type** Object (string)

**Request Structure**

- **federations** The top-level key that must exist - an array of objects that each describe a set of resolvers for a *Delivery Service’s* federation
  - **deliveryService** The `xml_id` of the *Delivery Service* which will use the federation resolvers specified in `mappings`
  - **mappings** An object containing two arrays of IP addresses to use as federation resolvers
    - **resolve4** An array of IPv4 addresses that can resolve the *Delivery Service’s* federation
    - **resolve6** An array of IPv6 addresses that can resolve the *Delivery Service’s* federation

**# 278: Request Example**

```json
POST /api/1.1/federations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 119
Content-Type: application/json

{
  "federations": [{
    "deliveryService": "demo1",
    "mappings": {
      "resolve4": ["0.0.0.0"],
      "resolve6": ["::"]
    }
  }]
}
```

**Response Structure**

**# 279: Response Example**

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
```

DELETE

Deletes all federation resolvers associated with the logged-in user’s federations.

Auth. Required Yes

Roles Required “admin”, “Federation”, “operations”, “Portal”, or “Steering”

Response Type Object (string)

Request Structure

No parameters available

Response Structure

# 280: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Mon, 03 Dec 2018 17:55:10 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 03 Dec 2018 21:55:10 GMT;
              path=/; HttpOnly
```

{ "response": "admin successfully created federation resolvers." }
PUT

Replaces all federations associated with a user's Delivery Service(s) with those defined inside the request payload.

Auth. Required  Yes

Roles Required  “admin”, “Federation”, “operations”, “Portal”, or “Steering”

Response Type  Object (string)

Request Structure

federations  The top-level key that must exist - an array of objects that each describe a set of resolvers for a Delivery Service’s federation

deliveryService  The ‘xml_id’ of the Delivery Service which will use the federation resolvers specified in mappings

mappings  An object containing two arrays of IP addresses to use as federation resolvers

resolve4  An array of IPv4 addresses that can resolve the Delivery Service’s federation

resolve6  An array of IPv6 addresses that can resolve the Delivery Service’s federation

# 281: Request Example

```
PUT /api/1.4/federations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 113
Content-Type: application/json

{ "federations": [{
   "deliveryService": "demol",
   "mappings": {
      "resolve4": ["192.168.1.1", "192.168.1.2"],
      "resolve6": ["2001:db8::1", "2001:db8::2"]
   }
}]
```

(continues on next page)
"resolve4": ["0.0.0.1"],
"resolve6": ["::1"]
}
}

Response Structure

# 282: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 00:52:31 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:52:30 GMT;
          path=/; HttpOnly
vary: Accept-Encoding, Accept-Encoding
whole-content-sha512: dXg86uD2Un1AeBCeeBLSo2rsYg16NOHHQEc5oM1pw1THOh2HwGdjwB3rPd/
qoYih0xcmnHoEstrEiHmucFev4A==
content-length: 63

{"response": "admin successfully created federation resolvers."}
```

`federations/{{ID}}/deliveryservices`

GET

Retrieves `Delivery Services` assigned to a federation.

Auth. Required  Yes

Roles Required  None

Response Type  Array
Request Structure

Table 158: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the federation to be inspected</td>
</tr>
</tbody>
</table>

# 283: Request Example

GET /api/1.4/federations/1/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **cdn**: The CDN to which this Delivery Service Belongs
- **id**: The integral, unique identifier for the Deliver Service
- **type**: The routing type used by this Delivery Service
- **xmlId**: The ‘xml_id’ which uniquely identifies this Delivery Service

# 284: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 00:44:13 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:44:13 GMT;

{ "response": [
    { "xmlId": "demo1",
      "cdn": "CDN-in-a-Box",
      ...
    }
  ]
```
POST

Assigns one or more *Delivery Services* to a federation.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** Object

**Request Structure**

Table 159: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the federation to be inspected</td>
</tr>
</tbody>
</table>

*dsIds* An array of integral, unique identifiers for *Delivery Services* which will be assigned to this federation

*replace* An optional boolean (default: *false*) which, if *true*, will cause any conflicting assignments already in place to be overridden by this request

**Note:** If *replace* is not given (and/or not *true*), then any conflicts with existing assignments will cause the entire operation to fail.

# 285: Request Example

```
POST /api/1.4/federations/1/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 32
Content-Type: application/json

{
    "dsIds": [1],
    "replace": true
}
```
Response Structure

**dsIds** An array of integral, unique identifiers for *Delivery Services* which are now assigned to this federation

**replace** An optional boolean (default: false) which, if true, means any conflicting assignments already in place were overridden by this request

```yaml
# 286: Response Example

HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
content-type: application/json
set-cookie: mojolicious=...; Path=/; HttpOnly
whole-content-sha512: rVd0nx8G3bR18ub1zw6FTdmwQ7jer4zoqzOZf5tC1ckrR0HEIOH1Azdcmvv0FVE5I0omcHvnrYbza
x-server-name: traffic_ops_golang/
content-length: 137
date: Wed, 05 Dec 2018 00:34:06 GMT

{
    "alerts": [
        {
            "text": "1 delivery service(s) were assigned to the federation 1",
            "level": "success"
        }
    ],
    "response": {
        "dsIds": [1],
        "replace": true
    }
}
```

*federations/{{ID}}/deliveryservices/{{dsID}}*

**DELETE**

Removes a *Delivery Service* from a federation. A *Delivery Service* cannot be removed from a federation if it is the only *Delivery Service* assigned to said federation

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** undefined
**Request Structure**

Table 160: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the federation from which the Delivery Service identified by dsID will be removed</td>
</tr>
<tr>
<td>dsID</td>
<td>The integral, unique identifier of the Delivery Service which will be removed from the federation identified by ID</td>
</tr>
</tbody>
</table>

# 287: Request Example

```
DELETE /api/1.4/federations/1/deliveryservices/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

# 288: Response Example

```json
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 01:06:51 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 05:06:51 GMT;
path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512: NqAZuZYlF1UWOaazbj/j4gWX7ye0kGGapRPFekK6ShxqXvCxEOdCTyu75qiLPN2wSgr3FGQnp2S3q345sE7In9g==
content-length: 98

{ "alerts": [  
  {    
    "level": "success",  
    "text": "Removed delivery service [ test ] from federation [ foo.bar. ]"
  }  
]}
```
federations/{{ID}}/federation_resolvers

GET

Retrieves federation resolvers assigned to a federation.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the federation for which resolvers will be retrieved</td>
</tr>
</tbody>
</table>

# 289: Request Example

```
GET /api/1.1/federations/1/federation_resolvers HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

- **id** The integral, unique identifier of this federation resolver
- **ipAddress** The IP address of the federation resolver - may be IPv4 or IPv6
- **type** The type of resolver - one of:
  - **RESOLVE4** This resolver is for IPv4 addresses (and ipAddress is IPv4)
  - **RESOLVE6** This resolver is for IPv6 addresses (and ipAddress is IPv6)

# 290: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *

cache-control: no-cache, no-store, max-age=0, must-revalidate
```

(continues on next page)
assigns one or more resolvers to a federation.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** Object

### Request Structure

Table 162: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the federation for which resolvers will be retrieved</td>
</tr>
</tbody>
</table>

- **fedResolverIds** An array of integral, unique identifiers for federation resolvers
- **replace** An optional boolean (default: `false`) which, if `true`, will cause any conflicting assignments already in place to be overridden by this request

**Note:** If `replace` is not given (and/or not `true`), then any conflicts with existing assignments will cause the entire operation to fail.
# 291: Request Example

```plaintext
POST /api/1.4/federations/1/federation_resolvers HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 41
Content-Type: application/json

{
    "fedResolverIds": [1],
    "replace": true
}
```

**Response Structure**

- **fedResolverIds**: An array of integral, unique identifiers for federation resolvers
- **replace**: An optionally-present boolean (default: false) which, if true, any conflicting assignments already in place were overridden by this request

# 292: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 00:47:47 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:47:47 GMT; path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512: +JDcRByS3HO6pMg3Gzkvn0w7/v5oRul9e+RxyFlOkJKNH0kZI1yQBS+PJpxDeCgwI19+0poW5dyHPPr9SwbNCA==
content-length: 148

{
    "alerts": [
        {
            "level": "success",
            "text": "1 resolver(s) were assigned to the test. federation"
        }
    ]
}
```

(continues on next page)
"response": {  
    "replace": true,
    "fedResolverIds": [  
        1  
    ]
}

federations/{{ID}}/users

GET

Retrieves users assigned to a federation.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 163: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the federation for which users will be retrieved</td>
</tr>
</tbody>
</table>

Response Structure

- company  The company to which the user belongs
- email  The user’s email address
- fullName  The user’s full name
- id  An integral, unique identifier for the user
- role  The user’s highest role
- username  The user’s short “username”

# 293: Response Example

HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *

(continues on next page)
POST

Assigns one or more users to a federation.

Auth. Required Yes

Roles Required “admin”

Response Type Object

Request Structure

userIds An array of integral, unique identifiers for users which will be assigned to this federation

replace An optional boolean (default: false) which, if true, will cause any conflicting assignments already in place to be overridden by this request

Note: If replace is not given (and/or not true), then any conflicts with existing assignments will cause the entire operation to fail.
# 294: Request Example

```
POST /api/1.4/federations/1/users HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 34
Content-Type: application/json

{
    "userIds": [2],
    "replace": true
}
```

## Response Structure

**userIds**  An array of integral, unique identifiers for users which have been assigned to this federation

**replace**  An optional boolean (default: false) which, if true, caused any conflicting assignments already in place to be overridden by this request

# 295: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST,GET,OPTIONS,PUT,DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 00:29:19 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:29:19 GMT; path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512: 5vPmgOA58aSOGvh+iEllf1gOexbaexg+qE2IPrQ2XH4iSX4jVey9adbGE9a9yaLj9uUMxg77N6ZyDhVqsbQ==
content-length: 137

{
    "alerts": [
        {
            "level": "success",
            "text": "1 user(s) were assigned to the test.quest.
```
federations/{{ID}}/users/{{userID}}

DELETE

Removes a user from a federation.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** undefined

### Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the federation from which the user identified by userID will be removed</td>
</tr>
<tr>
<td>userID</td>
<td>An integral, unique identifier for the user who will be removed from the federation identified by ID</td>
</tr>
</tbody>
</table>

### Request Structure

```http
DELETE /api/1.4/federations/1/users/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

```http
HTTP/1.1 200 OK
access-control-allow-credentials: true
```
access-control-allow-headers: Origin, X-Requested-With, Content-Type, Accept
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE
access-control-allow-origin: *
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json
date: Wed, 05 Dec 2018 01:14:04 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 05:14:04 GMT;
path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512:

```json
{
    "alerts": [
        {
            "level": "success",
            "text": "Removed user [ admin ] from federation [ foo.bar. ]"
        }
    ]
}
```

hwinf

Deprecated since version 1.1: This endpoint still works, but it is unused and serves no purpose. It will always return an empty response array unless the database is manually altered.

**GET**

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array
Request Structure:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>no</td>
<td>An integral, unique identifier of a specific hwinfo object which will be retrieved</td>
</tr>
<tr>
<td>server-HostName</td>
<td>no</td>
<td>The name of the server for which hwinfo objects will be retrieved</td>
</tr>
<tr>
<td>serverId</td>
<td>no</td>
<td>The integral, unique identifier of a server for which hwinfo objects will be retrieved</td>
</tr>
<tr>
<td>description</td>
<td>no</td>
<td>The description of a hwinfo object; only hwinfo objects with descriptions matching this will be retrieved</td>
</tr>
<tr>
<td>val</td>
<td>no</td>
<td>The value of a hwinfo object; only hwinfo objects with values matching this will be retrieved</td>
</tr>
<tr>
<td>lastUpdated</td>
<td>no</td>
<td>Only hwinfo objects that were last updated at this ISO-format date and time will be retrieved</td>
</tr>
</tbody>
</table>

Caution: The `lastUpdated` query parameter doesn’t seem to work properly, and its use is therefore discouraged.

Response Structure

- **description** Freeform description for this specific server’s hardware info
- **lastUpdated** The Time and Date for the last update for this server
- **serverHostName** Hostname for this specific server’s hardware info
- **serverId** Local unique identifier for this specific server’s hardware info
- **val** Freeform value used to track anything about a server’s hardware info

# 298: Response Example

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

(continues on next page)
**isos**

**POST**

Generates an ISO from the requested ISO source.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object - unless the `stream` key is present in the request payload, in which case the actual ISO content will be returned instead of a JSON response string

**Request Structure**

- **dhcp** A string that specifies whether the generated system image will use DHCP IP address leasing; one of:
  - **yes** DHCP will be used, and other network configuration keys need not be present in the request (and are ignored if they are)
  - **no** DHCP will not be used, and the desired network configuration must be specified manually in the request body

- **disk** An optional string that names the block device (under `/dev/`) used for the boot media, e.g. “sda”

- **domainName** The domain part of the system image’s Fully Qualified Domain Name (FQDN)

- **hostName** The host name part of the system image’s FQDN

- **interfaceMtu** A number that specifies the Maximum Transmission Unit (MTU) for the system image’s network interface card - the only valid values of which I’m aware are 1500 or 9000, and this should almost always just be 1500

- **interfaceName** An optional string naming the network interface to be used by the generated system image e.g. “bond0”, “eth0”, etc. If the special name “bond0” is used, a Link Aggregation Control Protocol (LACP) binding configuration will be created and included in the system image

**See also:**

The Link Aggregation Wikipedia page.
ip6Address  An optional string containing the IPv6 address of the generated system image

ip6Gateway  An optional string specifying the IPv6 address of the generated system image’s network gateway - this will be ignored if ipGateway is specified

ipAddress  An optional\(^1\) string containing the IP address of the generated system image

ipGateway  An optional\(^1\) string specifying the IP address of the generated system image’s network gateway

ipNetmask  An optional\(^1\) string specifying the subnet mask of the generated system image

osversionDir  The name of the directory containing the ISO source

See also:

osversions

rootPass  The password used by the generated system image’s root user

stream  An optional string that must be ‘yes’ or ‘no’ (Default: no) - if it is given and is ‘yes’, the response payload will be the content of the ISO rather than the normal JSON response

Note:  This is called ‘stream’ because it is implemented by writing the results of the ISO compression process directly into the TCP streaming socket after sending the necessary HTTP headers. As a result, this is much faster and more space-efficient than “stream”: “no” (which will first write the entire ISO to disk, then return a URL that can be used to download it) and so it is recommended in most cases that this be “yes”.

# 299: Request Example

```
POST /api/1.3/isos HTTP/1.1
Host: some.trafficops.host
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 334
Content-Type: application/json

{
    "osversionDir": "centos72",
    "hostName": "test",
    "domainName": "quest",
    "rootPass": "twelve",
}
```

(continues on next page)

\(^1\) This optional key is required if and only if dhcp is “no”.

5.1. Traffic Ops API 521
"dhcp": "no",
"interfaceMtu": 1500,
"ipAddress": "1.3.3.7",
"ipNetmask": "255.255.255.255",
"ipGateway": "8.0.0.8",
"ip6Address": "1::3::3::7",
"ip6Gateway": "8::8",
"interfaceName": "eth0",
"disk": "hda",
"stream": "no"
}

Response Structure

Assuming the stream key isn’t defined in the request payload JSON object (or it’s "no"), then the following keys will be present in the response object:

- **isoName**  The name of the generated .iso file
- **isoURL**  The URL location of the ISO

# 300: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Fri, 30 Nov 2018 20:27:10 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=e...; expires=Sat, 01 Dec 2018 00:27:10 GMT;
Vary: Accept-Encoding
Whole-Content-Sha512: pdlIVEfbcEiz6+JPWpDl+RVw6j66yzM3l9Bp/4Y19bh0Mh+aXe106Wwq05XnUlLszM/APWRwEYvUHtEdobGSAQ==
Content-Length: 243

{
  "alerts": [
    {
      "level": "success",
      "text": "Generate ISO was successful."
    }
  ],
  "response": {
    "isoURL": "https://some-weird-url.biz.co.uk/iso/test.centos72.iso"
  }
}
```
"isoName": "test.quest-centos72.iso"
}

jobs

GET

Get all jobs (currently limited to invalidate content (PURGE) jobs) sorted by start time (descending).

Auth. Required Yes

Roles Required “operations” or “admin”

Response Type Array

Warning: This endpoint will respect tenancy rules if and only if the dsId query parameter is used.

Request Structure

Table 166: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsId</td>
<td>no</td>
<td>Return only invalidation jobs pending on the Delivery Service identified by this integral, unique identifier</td>
</tr>
<tr>
<td>userId</td>
<td>no</td>
<td>Return only invalidation jobs created by the user identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>

Note: If the dsId parameter is given, an error will be returned if the thereby identified Delivery Service is not visible to the logged-in user’s Tenant

# 301: Request Example

GET /api/1.4/jobs?dsId=1&userId=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Response Structure

- **assetUrl**: A regular expression - matching URLs will be operated upon according to keyword.
- **createdBy**: The username of the user who initiated the job.
- **deliveryService**: The ‘xml_id’ that uniquely identifies the Delivery Service on which this job operates.
- **id**: An integral, unique identifier for this job.
- **keyword**: A keyword that represents the operation being performed by the job:
  - **PURGE**: This job will prevent caching of URLs matching the assetURL until it is removed (or its Time to Live expires).
- **parameters**: A string containing key/value pairs representing parameters associated with the job - currently only uses Time to Live e.g. "TTL:48h".
- **startTime**: The date and time at which the job began, in ISO format.

# 302: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 05 Dec 2018 15:44:07 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 05 Dec 2018 19:44:07 GMT;
                  path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: P9FJyakUTjK2iNawY3N3iljuH5VnJyNkdKRDtzhZN9gf4+HideoGIC19tcyRCDATQyZQ49/
                  BLEIJDAAqTwzA==
Content-Length: 202
```

```json
{
    "response": {
        "parameters": "TTL:48h",
        "keyword": "PURGE",
        "assetUrl": "http://origin.infra.ciab.test/.*\.jpg",
        "createdBy": "admin",
        "startTime": "2018-12-05 15:43:42+00",
        "id": 1,
        "deliveryService": "demo1"
    }
}
```

(continues on next page)
jobs/{{ID}}

GET

Get details about a specific job.

**Auth. Required**  Yes

**Roles Required**  “operations” or “admin”

**Response Type**  Array

### Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the job to be inspected</td>
</tr>
</tbody>
</table>

### Response Structure

- **assetUrl**  A regular expression - matching URLs will be operated upon according to **keyword**
- **createdBy**  The username of the user who initiated the job
- **deliveryService**  The ‘xml_id’ that uniquely identifies the Delivery Service on which this job operates
- **id**  An integral, unique identifier for this job
- **keyword**  A keyword that represents the operation being performed by the job:
  - **PURGE**  This job will prevent caching of URLs matching the assetURL until it is removed (or its Time to Live expires)
- **parameters**  A string containing key/value pairs representing parameters associated with the job - currently only uses Time to Live e.g. "TTL:48h"
- **startTime**  The date and time at which the job began, in ISO format
# 303: Response Example

```
{
    "response": [
        {
            "id": 1,
            "assetUrl": "http:\/\/foo-bar.domain.net\taco.html",
            "deliveryService": "foo-bar",
            "keyword": "PURGE",
            "parameters": "TTL:48h",
            "startTime": "2015-05-14 08:56:36-06",
            "createdBy": "jdog24"
        }
    ]
}
```

**logs**

**Note:** This endpoint’s responses will contain a cookie (last_seen_log) that is used by `logs/newcount` to determine the time of last access. Be sure your client uses cookies properly if you intend to use `logs/newcount` in concert with this endpoint!

**GET**

Fetches a list of changes that have been made to the Traffic Control system

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

Table 168: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>no</td>
<td>An integer number of days of change logs to return</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>The number of records to which to limit the response</td>
</tr>
</tbody>
</table>

# 304: Request Example

```
GET /api/1.4/logs?days=1&limit=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```

(continues on next page)
Response Structure

- **id**: Integral, unique identifier for the Log entry
- **lastUpdated**: Date and time at which the change was made, in ISO format
- **level**: Log categories for each entry, e.g. ‘UICHANGE’, ‘OPER’, ‘APICHANGE’
- **message**: Log detail about what occurred
- **ticketNum**: Optional field to cross reference with any bug tracking systems
- **user**: Name of the user who made the change

# 305: Response Example

```json
{
    "response": [
        {
            "ticketNum": null,
            "level": "APICHANGE",
            "lastUpdated": "2018-11-14 21:40:06.493975+00",
            "user": "admin",
            "id": 444,
            "message": "User [ test ] unlinked from deliveryservice [ 1 | demo1 ]."
        },
        {
            "ticketNum": null,
            "level": "UICHANGE",
            "lastUpdated": "2018-11-14 21:39:06.493975+00",
            "user": "admin",
            "id": 444,
            "message": "User [ test ] unlinked from deliveryservice [ 1 | demo1 ]."
        }
    ]
}
```

(continues on next page)
logs/{{days}}/days

Deprecated since version 1.1: Use logs with the ‘days’ query parameter instead

GET

Fetches a list of changes that have been made to the Traffic Control system in the past days

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 169: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>An integral number of days of change logs to return</td>
</tr>
</tbody>
</table>

Table 170: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>no</td>
<td>The number of rows to which to limit the response</td>
</tr>
</tbody>
</table>

# 306: Request Example

GET /api/1.4/logs/1/days?limit=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: last_seen_log="2018-11-15% 15:11:38"; mojolicious=...
Response Structure

id  Integral, unique identifier for the Log entry
lastUpdated  Date and time at which the change was made, in ISO format
level  Log categories for each entry, e.g. ‘UICHANGE’, ‘OPER’, ‘APICHANGE’
message  Log detail about what occurred
ticketNum  Optional field to cross reference with any bug tracking systems
user  Name of the user who made the change

# 307: Response Example

```json
{
   "response": [
      {
         "ticketNum": null,
         "level": "APICHANGE",
         "lastUpdated": "2018-11-14 21:40:06.493975+00",
         "user": "admin",
         "id": 444,
         "message": "User [ test ] unlinked from deliveryservice [ 1 | demo1 ]."
      },
      {
         "ticketNum": null,
         "level": "APICHANGE",
         "lastUpdated": "2018-11-14 21:37:30.707571+00",
         "user": "admin",
         "id": 443,
         "message": "1 delivery services were assigned to test"
      }
   ]
}
```

(continues on next page)
logs/newcount

**GET**

Gets the number of new changes made to the Traffic Control system - “new” being defined as the last time the client requested either *logs* or *logs/{{days}}/days*.

**Note:** This endpoint’s functionality is implemented by the *logs* and *logs/{{days}}/days* endpoints’ responses setting cookies for the client to use when requesting _this_ endpoint. Take care that your client respects cookies!

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Object

**Request Structure**

No parameters available

**Response Structure**

- **newLogcount**  The integer number of new changes

# 308: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 15 Nov 2018 15:17:35 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 19:17:35 GMT;
         path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Ugdqe8GXKSOExphwbDX/5li+2yRpuhtthpfYMbJaCP7adox3MzmVRi2RxTDL5kwPewrcLi1C088zGITSkB0sc9g==
```
origins

New in version 1.3.

GET

Gets all requested origins.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 171: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheGroup</td>
<td>no</td>
<td>Return only origins within the Cache Group identified by this integral, unique identifier</td>
</tr>
<tr>
<td>coordinate</td>
<td>no</td>
<td>Return only origins located at the geographic coordinates identified by this integral, unique identifier</td>
</tr>
<tr>
<td>deliveryService</td>
<td>no</td>
<td>Return only origins that belong to the Delivery Service identified by this integral, unique identifier</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the origin that has this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Return only origins by this name</td>
</tr>
<tr>
<td>profileId</td>
<td>no</td>
<td>Return only origins which use the Profile identified by this integral, unique identifier</td>
</tr>
<tr>
<td>primary</td>
<td>no</td>
<td>If true, return only origins which are the the primary origin of the Delivery Service to which they belong - if false return only origins which are not the primary origin of the Delivery Service to which they belong</td>
</tr>
<tr>
<td>tenant</td>
<td>no</td>
<td>Return only origins belonging to the tenant identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>
Note: Several fields of origin definitions which are filterable by Query Parameters are allowed to be null. Null values in these fields will be filtered out appropriately by such Query Parameters, but do note that null is not a valid value accepted by any of these Query Parameters, and attempting to pass it will result in an error.

# 309: Request Example

```
GET /api/1.4/origins?name=demo1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- `cachegroup` The name of the *Cache Group* to which the *origin* belongs
- `cachegroupId` An integral, unique identifier for the *Cache Group* to which the *origin* belongs
- `coordinate` The name of a coordinate pair that defines the origin’s geographic location
- `coordinateId` An integral, unique identifier for the coordinate pair that defines the *origin’s* geographic location
- `deliveryService` The ‘xml_id’ of the *Delivery Service* to which the *origin* belongs
- `deliveryServiceId` An integral, unique identifier for the *Delivery Service* to which the *origin* belongs
- `fqdn` The FQDN of the *origin*
- `id` An integral, unique identifier for this *origin*
- `ip6Address` The IPv6 address of the *Origin*
- `ipAddress` The IPv4 address of the *Origin*
- `isPrimary` A boolean value which, when true specifies this *origin* as the ‘primary’ *origin* served by *deliveryService*
- `lastUpdated` The date and time at which this *origin* was last modified
- `name` The name of the *origin*
- `port` The TCP port on which the *origin* listens
- `profile` The name of the *Profile* used by this *origin*
- `profileId` An integral, unique identifier for the *Profile* used by this *origin*
- `protocol` The protocol used by this origin - will be one of ‘http’ or ‘https’
tenant The name of the Tenant that owns this origin

tenantId An integral, unique identifier for the Tenant that owns this origin

### # 310: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: ...
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 15:43:41 GMT
Content-Length: 376

{
  "response": [
    {
      "cachegroup": null,
      "cachegroupId": null,
      "coordinate": null,
      "coordinateId": null,
      "deliveryService": "demo1",
      "deliveryServiceId": 1,
      "fqdn": "origin.infra.ciab.test",
      "id": 1,
      "ip6Address": null,
      "ipAddress": null,
      "isPrimary": true,
      "lastUpdated": "2018-12-10 19:11:32+00",
      "name": "demo1",
      "port": null,
      "profile": null,
      "profileId": null,
      "protocol": "http",
      "tenant": "root",
      "tenantId": 1
    }
  ]
}
```

**POST**

Creates a new origin definition.

5.1. Traffic Ops API 533
Warning: At the time of this writing it is possible to create and/or modify origin definitions assigned to STEERING and CLIENT_STEERING Delivery Services - despite that an origin has no meaning in those contexts. In these cases, the API responses may give incorrect output - see GitHub Issue #3107 for details and updates.

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type Object

Request Structure

cachegroupId An optional, integral, unique identifier that identifies a Cache Group to which the new origin shall belong
coordinateID An optional, integral, unique identifier of a coordinate pair that shall define the origin's geographic location
deliveryServiceId The integral, unique identifier of the Delivery Service to which the new origin shall belong
fqdn The FQDN of the origin
ip6Address An optional string containing the IPv6 address of the origin
ipAddress An optional string containing the IPv4 address of the origin
isPrimary An optional boolean which, if true will set this origin as the ‘primary’ origin served by the Delivery Service identified by deliveryServiceID

Note: Though not specifying this field in this request will leave it as null in the output, Traffic Ops will silently coerce that to its default value: false.

name A human-friendly name of the Origin
port An optional port number on which the origin listens for incoming TCP connections
profileId An optional, integral, unique identifier for a Profile that the new origin shall use
protocol The protocol used by the origin - must be one of ‘http’ or ‘https’
tenantId An optional1, integral, unique identifier for the Tenant which shall own the new origin

1 The tenantId field is required if and only if tenancy is enabled within Traffic Ops.
# 311: Request Example

```bash
POST /api/1.4/origins HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 114
Content-Type: application/json

{
    "deliveryServiceId": 2,
    "fqdn": "example.com",
    "name": "example",
    "port": 80,
    "protocol": "http",
    "tenantId": 1
}
```

## Response Structure

- **cachegroup** The name of the *Cache Group* to which the *origin* belongs
- **cachegroupId** An integral, unique identifier for the *Cache Group* to which the *origin* belongs
- **coordinate** The name of a coordinate pair that defines the origin’s geographic location
- **coordinateId** An integral, unique identifier for the coordinate pair that defines the *origin*’s geographic location
- **deliveryService** The ‘xml_id’ of the *Delivery Service* to which the *origin* belongs
- **deliveryServiceId** An integral, unique identifier for the *Delivery Service* to which the *origin* belongs
- **fqdn** The FQDN of the *origin*
- **id** An integral, unique identifier for this *origin*
- **ip6Address** The IPv6 address of the *Origin*
- **ipAddress** The IPv4 address of the *Origin*
- **isPrimary** A boolean value which, when true specifies this *origin* as the ‘primary’ *origin* served by deliveryService
- **lastUpdated** The date and time at which this *origin* was last modified
- **name** The name of the *origin*
- **port** The TCP port on which the *origin* listens

---

5.1. Traffic Ops API 535
profile  The name of the Profile used by this origin
profileId  An integral, unique identifier for the Profile used by this origin
protocol  The protocol used by this origin - will be one of ‘http’ or ‘https’
tenant  The name of the Tenant that owns this origin
tenantId  An integral, unique identifier for the Tenant that owns this origin

# 312: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: z4gp0MaqYu+gSRORhKT2eObVBuVDVx1rdteRaN5kRL9uJ3hNzUCi4dSKIt0rgNgOEDt6x/ iTYrmWhr/TSHYtmA==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 15:14:27 GMT
Content-Length: 418

{
    "alerts": [
        {
            "text": "origin was created.",
            "level": "success"
        }
    ],
    "response": {
        "cachegroup": null,
        "cachegroupId": null,
        "coordinate": null,
        "coordinateId": null,
        "deliveryService": null,
        "deliveryServiceId": 2,
        "fqdn": "example.com",
        "id": 2,
        "ip6Address": null,
        "ipAddress": null,
        "isPrimary": null,
        "lastUpdated": "2018-12-11 15:14:27+00",
        "name": "example",
        "port": 80,
        "profile": null,
        "profileId": null,
        "protocol": "http",
        "tenant": null,
    }
}
```
PUT

Updates an origin definition.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object

Request Structure

Table 172: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the origin definition being edited</td>
</tr>
<tr>
<td>cachegroupId</td>
<td></td>
<td>An optional, integral, unique identifier that identifies a Cache Group to which the origin shall belong</td>
</tr>
<tr>
<td>coordinateID</td>
<td></td>
<td>An optional, integral, unique identifier of a coordinate pair that shall define the origin’s geographic location</td>
</tr>
<tr>
<td>deliveryServiceId</td>
<td></td>
<td>The integral, unique identifier of the Delivery Service to which the origin shall belong</td>
</tr>
<tr>
<td>fqdn</td>
<td></td>
<td>The FQDN of the origin</td>
</tr>
<tr>
<td>ip6Address</td>
<td></td>
<td>An optional string containing the IPv6 address of the origin</td>
</tr>
<tr>
<td>ipAddress</td>
<td></td>
<td>An optional string containing the IPv4 address of the origin</td>
</tr>
<tr>
<td>isPrimary</td>
<td></td>
<td>An optional boolean which, if true will set this origin as the ‘primary’ origin served by the Delivery Service identified by deliveryServiceID</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>A human-friendly name of the Origin</td>
</tr>
<tr>
<td>port</td>
<td></td>
<td>An optional port number on which the origin listens for incoming TCP connections</td>
</tr>
<tr>
<td>profileId</td>
<td></td>
<td>An optional, integral, unique identifier for a Profile that the new origin shall use</td>
</tr>
<tr>
<td>protocol</td>
<td></td>
<td>The protocol used by the origin - must be one of ‘http’ or ‘https’</td>
</tr>
<tr>
<td>tenantId</td>
<td></td>
<td>An optional, integral, unique identifier for the Tenant which shall own the new origin</td>
</tr>
</tbody>
</table>

5.1. Traffic Ops API
# 313: Request Example

```
PUT /api/1.4/origins?id=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 135
Content-Type: application/json

{
    "deliveryServiceId": 2,
    "fqdn": "example.com",
    "isPrimary": true,
    "name": "example",
    "port": 443,
    "protocol": "https",
    "tenantId": 1
}
```

## Response Structure

- **cachegroup** The name of the *Cache Group* to which the *origin* belongs
- **cachegroupId** An integral, unique identifier for the *Cache Group* to which the *origin* belongs
- **coordinate** The name of a coordinate pair that defines the origin’s geographic location
- **coordinateID** An integral, unique identifier for the coordinate pair that defines the *origin’s* geographic location
- **deliveryService** The ‘xml_id’ of the *Delivery Service* to which the *origin* belongs
- **deliveryServiceId** An integral, unique identifier for the *Delivery Service* to which the *origin* belongs
- **fqdn** The FQDN of the *origin*
- **id** An integral, unique identifier for this *origin*
- **ip6Address** The IPv6 address of the *Origin*
- **ipAddress** The IPv4 address of the *Origin*
- **isPrimary** A boolean value which, when true specifies this *origin* as the ‘primary’ *origin* served by deliveryService
- **lastUpdated** The date and time at which this *origin* was last modified
- **name** The name of the *origin*
- **port** The TCP port on which the *origin* listens
profile  The name of the Profile used by this origin
profileId  An integral, unique identifier for the Profile used by this origin
protocol  The protocol used by this origin - will be one of ‘http’ or ‘https’
tenant  The name of the Tenant that owns this origin
tenantId  An integral, unique identifier for the Tenant that owns this origin

# 314: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Zx7jOa7UAQxRtDenYodvGQSo0Oj4m0yY0AIEUpbdelmYMiNdPYtW82BCmMesFXkmP74nV4HbTUmDH
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 15:40:53 GMT
Content-Length: 420

{  "alerts": [
    {  "text": "origin was updated.",
        "level": "success"
    }
],
"response": {
    "cachegroup": null,
    "cachegroupId": null,
    "coordinate": null,
    "coordinateId": null,
    "deliveryService": null,
    "deliveryServiceId": 2,
    "fqdn": "example.com",
    "id": 2,
    "ip6Address": null,
    "ipAddress": null,
    "isPrimary": true,
    "lastUpdated": "2018-12-11 15:40:53+00",
    "name": "example",
    "port": 443,
    "profile": null,
    "profileId": null,
    "protocol": "https",
    "tenant": null,
    "tenantId": 1
}}
```
DELETE

Deletes an *origin* definition.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>yes</td>
<td>The integral, unique identifier of the <em>origin</em> definition being deleted</td>
</tr>
</tbody>
</table>

# 315: Request Example

```plaintext
DELETE /api/1.4/origins?id=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

# 316: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: fLaY4/nh0yr38x5weBKYg02+aQV621ZroOq9UqUCHLMrH1NMyhOHx+EphPq7JxkJmGY04WCt6VvDyjGWcg
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 17:04:14 GMT
Content-Length: 61

{ "alerts": [ 
  
    "text": "origin was deleted."
  ,
    "level": "success"
  ]
}
```
osversions

See also:

Generate ISO

GET

Gets all available Operating System (OS) versions for ISO generation, as well as the name of
the directory where the “kickstarter” files are found.

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

No parameters available.

Response Structure

This endpoint has no constant keys in its response. Instead, each key in the response object
is the name of an OS, and the value is a string that names the directory where the ISO source
can be found. These directories sit under /var/www/files/ on the Traffic Ops host machine by
default, or at the location defined by the kickstart.files.location parameter, if it is
defined.

# 317: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Fri, 30 Nov 2018 19:14:36 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 30 Nov 2018 23:14:36 GMT;
(path=/); HttpOnly
```
parameterprofile

Deprecated since version 1.1: Use `profileparameters` instead.

**POST**

Create one or more parameter/profile assignments.

**Auth. Required** Yes  
**Roles Required** “admin” or “operations”  
**Response Type** Object

**Request Structure**

- **paramId** The integral, unique identifier for the parameter to be assigned to the profiles identified within the `profileIds` array
- **profileIds** An array of integral, unique identifiers for profiles to which the parameter identified by `paramId` shall be assigned
- **replace** An optional boolean (default: false) which, if `true`, will cause any conflicting profile/parameter assignments to be overridden.

### 318: Request Example

```
POST /api/1.4/profileparameter HTTP/1.1  
Host: trafficops.infra.ciab.test  
User-Agent: curl/7.47.0  
Accept: */*  
Cookie: mojolicious=...  
Content-Length: 38  
Content-Type: application/json

{
  "paramId": 4,
  "CentOS 7.2": "centos72"
}
```
Response Structure

**paramId** The integral, unique identifier for the parameter which has been assigned to the profiles identified within the `profileIds` array

**profileIds** An array of integral, unique identifiers for profiles to which the parameter identified by `paramId` has been assigned

**replace** An optional boolean (default: false) which, if `true`, caused any conflicting profile/parameter assignments to be overridden.

### # 319: Response Example

```json
{
    "alerts": [
        {
            "text": "2 parameters were assigned to the 18 profile",
            "level": "success"
        }
    ],
    "response": {
        "profileId": 18,
        "paramIds": [
            2,
            3
        ],
        "replace": false
    }
}
```
parameters

GET

Gets all parameters configured in Traffic Ops

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 174: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>no</td>
<td>Filter parameters by integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Filter parameters by name</td>
</tr>
<tr>
<td>configFile</td>
<td>no</td>
<td>Filter parameters by configuration file</td>
</tr>
</tbody>
</table>

# 320: Request Example

```
GET /api/1.4/parameters?configFile=records.config&name=location

HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

configFile The base filename to which this parameter belongs

id An integral, unique identifier for this parameter

lastUpdated The date and time at which this parameter was last modified in ISO format

name The parameter name

profiles An array of profile names that use this parameter

secure When true, the parameter value is visible only to “admin”-role users

value The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing
# 321: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: UFO3/jcBFmFZM7CsrsIwTfPc5v8gUixqJm6BNp1boPb4EQBnWNXZh/DbBwhMAOJoeqDIMoD1rLnrVjQGO4AooA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 18:23:39 GMT
Content-Length: 212

{
  "response": [
    {
      "configFile": "records.config",
      "id": 29,
      "lastUpdated": "2018-12-05 17:51:02+00",
      "name": "location",
      "profiles": [
        "ATS_EDGE_TIER_CACHE",
        "ATS_MID_TIER_CACHE"
      ],
      "secure": false,
      "value": "/etc/trafficserver/
    }
  ]
}
```

## POST

Creates one or more new parameters.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Array

### Request Structure

The request body may be in one of two formats, a single parameter object or an array of parameter objects. Each parameter object shall have the following keys:
Caution: At the time of this writing, there is a bug in the Go rewrite of this endpoint such that the “array format” will not be accepted by the server. Watch GitHub issue #3093 for further developments.

name Parameter name

cfgFile The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

secure A boolean value which, when true will prohibit users who do not have the “admin” role from viewing the parameter’s value (at the time of this writing the obfuscation value is defined to be “********”)

value Parameter value

# 322: Request Example - Single Object Format

POST /api/1.4/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 84
Content-Type: application/json

{
    "name": "test",
    "value": "quest",
    "cfgFile": "records.config",
    "secure": false
}

# 323: Request Example - Array Format

POST /api/1.4/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 180
Content-Type: application/json

[
    {
        "name": "test",
        "value": "quest",
        "cfgFile": "records.config",
        "secure": false
    },
    {
        "name": "foo",
        "value": "bar",
        "cfgFile": "path/to/records.config",
        "secure": true
    }
]
Response Structure

- **configFile**: The base filename to which this parameter belongs
- **id**: An integral, unique identifier for this parameter
- **lastUpdated**: The date and time at which this parameter was last modified in ISO format
- **name**: The parameter name
- **profiles**: An array of profile names that use this parameter - should be null immediately after parameter creation
- **secure**: When true, the parameter value is visible only to “admin”-role users
- **value**: The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: eQrl48zWids0kDpfCYmntYMpegjnFxfOVvlBYxxLSfp7P7p6oWX4uiC/+Cfh2X9i3G+MQ36eH95gukJqOBQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 19:18:21 GMT
Content-Length: 212

```json
{
    "alerts": [
        {
            "text": "param was created.",
            "level": "success"
        }
    ],
    "response": {
        "configFile": "records.config",
```
parameters/{{ID}}

GET

Gets details about a specific parameter

Deprecated since version 1.1: Use the `id` query parameter of the `parameters` endpoint instead

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 175: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the parameter which will be deleted</td>
</tr>
</tbody>
</table>

# 325: Request Example

GET /api/1.4/parameters/29 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **configFile**  The `base` filename to which this parameter belongs
- **id**  An integral, unique identifier for this parameter
- **lastUpdated**  The date and time at which this parameter was last modified in ISO format
- **name**  The parameter name
profiles An array of profile names that use this parameter

secure When true, the parameter value is visible only to “admin”-role users

value The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

# 326: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: UFO3/jcBFmFZM7csr1wTfPc5v8gUixQjm6BNp1boPb4EQBnWNXZh/-DbBwhMAOjegDImoD1rLnrVjQGO4AooA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 19:01:54 GMT
Content-Length: 212

{ "response": [ {
   "configFile": "records.config",
   "id": 29,
   "lastUpdated": "2018-12-05 17:51:02+00",
   "name": "location",
   "profiles": [
      "ATS_EDGE_TIER_CACHE",
      "ATS_MID_TIER_CACHE"
   ],
   "secure": false,
   "value": "/etc/trafficserver/"
} ]}
```

PUT

Replaces a parameter.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object

5.1. Traffic Ops API
Request Structure

Table 176: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the parameter which will be deleted</td>
</tr>
<tr>
<td>name</td>
<td>Parameter name</td>
</tr>
<tr>
<td>configFile</td>
<td>The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/fooo”</td>
</tr>
<tr>
<td>secure</td>
<td>A boolean value which, when true will prohibit users who do not have the “admin” role from viewing the parameter’s value (at the time of this writing the obfuscation value is defined to be &quot;********&quot;)</td>
</tr>
<tr>
<td>value</td>
<td>Parameter value</td>
</tr>
</tbody>
</table>

# 327: Request Example

```
PUT /api/1.4/parameters/124 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 81
Content-Type: application/json

{
    "name": "foo",
    "value": "bar",
    "configFile": "records.config",
    "secure": false
}
```

Response Structure

<table>
<thead>
<tr>
<th>configFile</th>
<th>The base filename to which this parameter belongs</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>An integral, unique identifier for this parameter</td>
</tr>
<tr>
<td>lastUpdated</td>
<td>The date and time at which this parameter was last modified in ISO format</td>
</tr>
<tr>
<td>name</td>
<td>The parameter name</td>
</tr>
<tr>
<td>profiles</td>
<td>An array of profile names that use this parameter</td>
</tr>
<tr>
<td>secure</td>
<td>When true, the parameter value is visible only to “admin”-role users</td>
</tr>
<tr>
<td>value</td>
<td>The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be &quot;********&quot;)</td>
</tr>
</tbody>
</table>
value is defined to be "********") but not missing

# 328: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: DMxS2gKceFVKRtezO/N/vsnrC+zI8onASSHaGv5i3wwvUvyt9KEe72gxQd6ZgVcSg3K82pkH6g3UI/WtEfdp5vA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 20:21:07 GMT
Content-Length: 209

{
  "alerts": [
    {
      "text": "param was updated.",
      "level": "success"
    }
  ],
  "response": {
    "configFile": "records.config",
    "id": 125,
    "lastUpdated": "2018-12-05 20:21:07+00",
    "name": "foo",
    "profiles": null,
    "secure": false,
    "value": "bar"
  }
}
```

DELETE

Deletes the specified parameter. If, however, the parameter is associated with one or more profiles, deletion will fail.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  undefined
Request Structure

Table 177: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the parameter which will be deleted</td>
</tr>
</tbody>
</table>

# 329: Request Example

```plaintext
DELETE /api/1.4/parameters/124 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 330: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: hJjQq2Seg7sqWt+jKgp6gwRxUtoVU34PFoc9wEaweXdaIBTn/BscoUuyw2/
+mV8GZPqpeQc1h2E50/0oQhdtHw==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 19:20:30 GMT
Content-Length: 60

```json```

```json
{
    "alerts": [
        {
            "text": "param was deleted.",
            "level": "success"
        }
    ]
}
```

```parameters/{{ID}}/profiles```

Deprecated since version 1.1: Use the `param` query parameter of `profiles` instead.
GET

Retrieves all profiles assigned to the parameter.

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

### Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier that specifies for which parameter shall profiles be listed</td>
</tr>
</tbody>
</table>

#### # 331: Request Structure

```plaintext
GET /api/1.4/parameters/4/profiles HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

- **description**  A description of profile
- **id**  An integral, unique identifier for this profile
- **lastUpdated**  The date and time at which this profile was last updated
- **name**  Profile name
- **routingDisabled**  An integer that defines whether or not Traffic Routers will route to servers using these profiles - can only be one of:
  - 0  Traffic Routers will route traffic to these servers normally
  - 1  Traffic Routers will ignore these servers, and not route traffic to them
- **type**  The profile’s type

#### # 332: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
```

(continues on next page)
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 05 Dec 2018 20:51:23 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 06 Dec 2018 00:51:23 GMT;
  path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
  y5fA9q1VogDGxL66ka+ofTtLo3JiTj+Bdrvc4DnfrjFyzql1+537WySFj1nE0C29Twx51/
  C8JEHy3Byaz/wbfA==
Content-Length: 184

{ "response": [
  {
    "routingDisabled": 0,
    "lastUpdated": "2018-12-05 17:50:49.007102+00",
    "name": "GLOBAL",
    "type": "UNK_PROFILE",
    "id": 1,
    "description": "Global Traffic Ops profile, DO NOT DELETE"
  }
]}

parameters/{{ID}}/unassigned_profiles

Warning: There are very few good reasons to use this endpoint - be sure not limit said use.

GET

Retrieves all profiles to which the specified parameter is NOT assigned to the parameter.

  Auth. Required  Yes
  Roles Required  None
  Response Type  Array
Request Structure

Table 179: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier that specifies for which parameter unassigned profiles shall be listed</td>
</tr>
</tbody>
</table>

# 333: Request Example

GET /api/1.4/parameters/43/unassigned_profiles HTTP/1.1  
Host: trafficops.infra.ciab.test  
User-Agent: curl/7.47.0  
Accept: */*  
Cookie: mojolicious=...

Response Structure

- **description**  A description of profile
- **id**  An integral, unique identifier for this profile
- **lastUpdated**  The date and time at which this profile was last updated
- **name**  Profile name
- **routingDisabled**  An integer that defines whether or not Traffic Routers will route to servers using these profiles - can only be one of:
  - 0  Traffic Routers will route traffic to these servers normally
  - 1  Traffic Routers will ignore these servers, and not route traffic to them
- **type**  The profile’s type

# 334: Response Example

HTTP/1.1 200 OK  
Access-Control-Allow-Credentials: true  
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-  
Type, Accept  
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE  
Access-Control-Allow-Origin: *  
Cache-Control: no-cache, no-store, max-age=0, must-revalidate  
Content-Type: application/json  
Date: Wed, 05 Dec 2018 21:47:48 GMT  
Server: Mojolicious (Perl)  
Set-Cookie: mojolicious=...; expires=Thu, 06 Dec 2018 01:47:48 GMT;  
path=/; HttpOnly  
Vary: Accept-Encoding  
Whole-Content-Sha512: c669pfp21OP2B2xQ1v6RKDbul5cXvAtGnr7wEzW4ku+7E1mNvEyBQFrk+8hFeY/0hWoJcm4/2YGBR39g==
Transfer-Encoding: chunked

```json
{
  "response": [
    {
      "cdn": 1,
      "lastUpdated": "2018-12-05 17:50:49.007102+00",
      "name": "GLOBAL",
      "description": "Global Traffic Ops profile, DO NOT DELETE",
      "cdnName": "ALL",
      "routingDisabled": false,
      "id": 1,
      "type": "UNK_PROFILE"
    },
    {
      "cdn": 1,
      "lastUpdated": "2018-12-05 17:50:49.024653+00",
      "name": "TRAFFIC_ANALYTICS",
      "description": "Traffic Analytics profile",
      "cdnName": "ALL",
      "routingDisabled": false,
      "id": 2,
      "type": "UNK_PROFILE"
    }
  ]
}
```

**Note:** The Response Example above has been truncated to only its first two array elements, as the true output was very long.

`parameters/profile/{{name}}`

Deprecated since version 1.1: Use `profiles/name/{{name}}/parameters` instead

**GET**

Gets details about a specific profile’s parameters

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array
Request Structure

Table 180: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the profile for which parameters will be listed</td>
</tr>
</tbody>
</table>

# 335: Request Example

GET /api/1.4/parameters/profile/GLOBAL HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **configFile** The base filename to which this parameter belongs
- **id** An integral, unique identifier for this parameter
- **lastUpdated** The date and time at which this parameter was last modified in ISO format
- **name** The parameter name
- **profiles** An array of profile names that use this parameter
- **secure** When true, the parameter value is visible only to “admin”-role users
- **value** The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

# 336: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: NudgZXUNyKNpmSf856KEjyy+Pin/bFhG9NoRDAXyBrKt2T5fF52e7sUNzfF15n/ZZsgbx6Tsgfvd7oM6j+eg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 21:08:56 GMT
Content-Length: 542

(continues on next page)
parameters/validate

Deprecated since version 1.1: To check for the existence of a parameter with specific name, value etc., use the query parameters of the parameters endpoint instead.

POST

Returns a successful response and message if a parameter matching the one in the payload exists, and an error response and message if no such parameter is found.

**Auth. Required** Yes

**Roles Required** None
Response Type Object or undefined - no response key is returned if the provided parameter could not be matched

Request Structure

name Parameter name

cconfigFile The base filename of the configuration file to which this parameter belongs e.g. “foo” not “/path/to/foo”

secure When true, the parameter value is visible only to “admin”-role users

value The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

# 337: Request Example

POST /api/1.4/parameters/validate HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 80
Content-Type: application/json

{
    "name": "foo",
    "value": "bar",
    "configFile": "records.config",
    "secure": true
}

Response Structure

configFile The base filename to which this parameter belongs

id An integral, unique identifier for this parameter

lastUpdated The date and time at which this parameter was last modified in ISO format

name The parameter name

profiles An array of profile names that use this parameter

secure When true, the parameter value is visible only to “admin”-role users

value The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing
# 338: Response Example - Parameter Found

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 05 Dec 2018 20:35:42 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 06 Dec 2018 00:35:42 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
   CcsN9WhMPnviPtBacTecILmleMlZxEySwmk3rdCclydPu0cMgef/RVI/aRYe+IDAkWFmpeZHg+g1Edl1R7dfWg==
Content-Length: 149

```json
{
   "alerts": [
      {
         "level": "success",
         "text": "Parameter exists."
      }
   ],
   "response": {
      "value": "bar",
      "name": "foo",
      "secure": 0,
      "id": 125,
      "configFile": "records.config"
   }
}
```

# 339: Response Example - Parameter Not Found

HTTP/1.1 400 Bad Request
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 05 Dec 2018 20:42:10 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 06 Dec 2018 00:42:10 GMT; path=/; HttpOnly
Vary: Accept-Encoding

(continues on next page)
**Note:** This endpoint returns a client-side error response when the parameter was not found - as such any API tools that wish to use this endpoint should be aware that a client-side error response code may not actually mean that an error occurred. However, neither can it be said that a 400 response code means that the parameter wasn’t found; that response code is also returned in the event of _true_ client-side errors e.g. a malformed JSON payload in the request.

### phys_locations

**GET**

Retrieves physical locations

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Filter by integral, unique identifier</td>
</tr>
<tr>
<td>region</td>
<td>Filter by integral, unique identifier of containing region</td>
</tr>
<tr>
<td>name</td>
<td>Filter by name</td>
</tr>
</tbody>
</table>

**# 340: Request Example**

```plaintext
GET /api/1.4/phys_locations?name=CDN_in_a_Box HTTP/1.1
Host: trafficops.infra.ciab.test
```

(continues on next page)
Response Structure

address  The physical location’s street address

city  The name of the city in which the physical location lies

comments  Any and all human-readable comments

e-mail  The email address of the physical location’s poc

id  An integral, unique identifier for the physical location

lastUpdated  The date and time at which the physical location was last updated, in ISO format

name  The name of the physical location

phone  A phone number where the the physical location’s poc might be reached

poc  The name of a “point of contact” for the physical location

region  The name of the region within which the physical location lies

regionId  An integral, unique identifier for the region within which the physical location lies

shortName  An abbreviation of the name

state  An abbreviation of the name of the state or province within which this physical location lies

zip  The zip code of the physical location

# 341: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 0g4b3W1AwXytCnBo8TrEqQi1j2v9oHA17MG9KuwMi65V4sFcMM5qP8dgPsFTunFr00DPl20c7BpUbZs
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 22:19:52 GMT
Content-Length: 275
POST

Creates a new physical location

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

address  The physical location’s street address

city  The name of the city in which the physical location lies

comments  An optional string for containing any and all human-readable comments

email  An optional string containing email address of the physical location’s poc

name  An optional name of the physical location

phone  An optional string containing the phone number where the the physical location’s poc might be reached

poc  The name of a “point of contact” for the physical location

region  An optional string naming the region that contains this physical location¹

¹ The only “region” key that actually matters in the request body is regionId; region is not validated and has no effect - particularly not the effect of re-naming the region - beyond changing the name in the API response.
regionId An integral, unique identifier for the region within which the physical location lies

shortName An abbreviation of the name

state An abbreviation of the name of the state or province within which this physical location lies

zip The zip code of the physical location

# 342: Request Example

```json
POST /api/1.4/phys_locations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 326
Content-Type: application/json

{
    "address": "Buckingham Palace",
    "city": "London",
    "comments": "Buckingham Palace",
    "email": "steve.kingstone@royal.gsx.gov.uk",
    "name": "Great_Britain",
    "phone": "0-843-816-6276",
    "poc": "Her Majesty The Queen Elizabeth Alexandra Mary Windsor II",
    "regionId": 3,
    "shortName": "uk",
    "state": "Westminster",
    "zip": "SW1A 1AA"
}
```

Response Structure

address The physical location’s street address
city The name of the city in which the physical location lies
comments Any and all human-readable comments
e-mail The email address of the physical location’s poc
id An integral, unique identifier for the physical location
lastUpdated The date and time at which the physical location was last updated, in ISO format
name The name of the physical location

to this request. Subsequent requests will reveal the true name of the region. Note that if region is not present in the request body it will be null in the response, but again further requests will show the true region name.
phone  A phone number where the the physical location’s poc might be reached
poc   The name of a “point of contact” for the physical location
region The name of the region within which the physical location lies
regionId An integral, unique identifier for the region within which the physical location lies
shortName An abbreviation of the name
state  An abbreviation of the name of the state or province within which this physical location lies
zip    The zip code of the physical location

# 343: Response Example

```http
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: GZ/BC+AgGp0QNd9oiZy19jtsD8MPoOeyi7PvDz+9YSilYP44gmn5K+Xl1yS0159yHf7O+C11oQVQPS1
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 00:14:47 GMT
Content-Length: 443

{
"alerts": [

    {
      "text": "physLocation was created.",
      "level": "success"
    }
],
"response": {
    "address": "Buckingham Palace",
    "city": "London",
    "comments": "Buckingham Palace",
    "email": "steve.kingstone@royal.gsx.gov.uk",
    "id": 3,
    "lastUpdated": "2018-12-06 00:14:47+00",
    "name": "Great_Britain",
    "phone": "0-843-816-6276",
    "poc": "Her Majesty The Queen Elizabeth Alexandra Mary Windsor II",
    "regionId": 3,
    "region": null,
    "shortName": "uk",
    "state": "Westminster",

```
phys_locations/{{ID}}

GET

Retrieves information about a specific physical location

Deprecated since version 1.1: Use the id query parameter of the phys_locations endpoint instead

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 182: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the physical location being retrieved</td>
</tr>
</tbody>
</table>

# 344: Request Example

```
GET /api/1.4/phys_locations/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

address  The physical location’s street address
city    The name of the city in which the physical location lies
comments  Any and all human-readable comments
e-mail  The email address of the physical location’s poc
id      An integral, unique identifier for the physical location
lastUpdated  The date and time at which the physical location was last updated, in ISO format
name  The name of the physical location
phone  A phone number where the physical location’s poc might be reached
poc  The name of a “point of contact” for the physical location
region  The name of the region within which the physical location lies
regionId  An integral, unique identifier for the region within which the physical location lies
shortName  An abbreviation of the name
state  An abbreviation of the name of the state or province within which this physical location lies
zip  The zip code of the physical location

# 345: Response Example

```
HTTP/1.1  200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 0g4b3W1AwXytCnBo8TreQQij2v9oHA17MG9KuwMig5V4sFcMM5qP8dgPsFTunFr00DPI20c7BpUb2s
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 22:45:47 GMT
Content-Length: 275

{  "response": [  
    {  "address":  "1600 Pennsylvania Avenue NW",  "city":  "Washington",  "comments":  "",  "email":  "",  "id":  2,  "lastUpdated":  "2018-12-05 17:50:58+00",  "name":  "CDN_in_a_Box",  "phone":  "",  "poc":  "",  "regionId":  1,  "region":  "Washington, D.C",  "shortName":  "ciab",  "state":  "DC",  "zip":  "20500"
    }  
  ]}
```
PUT

Updates a physical location

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type Array

Request Structure

Table 183: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the physical location being modified</td>
</tr>
<tr>
<td>address</td>
<td>The physical location’s street address</td>
</tr>
<tr>
<td>city</td>
<td>The name of the city in which the physical location lies</td>
</tr>
<tr>
<td>comments</td>
<td>An optional string for containing any and all human-readable comments</td>
</tr>
<tr>
<td>email</td>
<td>An optional string containing email address of the physical location’s poc</td>
</tr>
<tr>
<td>name</td>
<td>An optional name of the physical location</td>
</tr>
<tr>
<td>phone</td>
<td>An optional string containing the phone number where the the physical location’s poc might be reached</td>
</tr>
<tr>
<td>poc</td>
<td>The name of a “point of contact” for the physical location</td>
</tr>
<tr>
<td>region</td>
<td>An optional string naming the region that contains this physical location</td>
</tr>
<tr>
<td>regionId</td>
<td>An integral, unique identifier for the region within which the physical location lies</td>
</tr>
<tr>
<td>shortName</td>
<td>An abbreviation of the name</td>
</tr>
<tr>
<td>state</td>
<td>An abbreviation of the name of the state or province within which this physical location lies</td>
</tr>
<tr>
<td>zip</td>
<td>The zip code of the physical location</td>
</tr>
</tbody>
</table>

# 346: Request Structure

PUT /api/1.4/phys_locations/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0

(continues on next page)

---

1 The only “region” key that actually matters in the request body is regionId; region is not validated and has no effect - particularly not the effect of re-naming the region - beyond changing the name in the API response to this request. Subsequent requests will reveal the true name of the region. Note that if region is not present in the request body it will be null in the response, but again further requests will show the true region name.
Accept: */*
Cookie: mojolicious=...
Content-Length: 268
Content-Type: application/json

{  
  "address": "1600 Pennsylvania Avenue NW",
  "city": "Washington",
  "comments": "The White House",
  "email": "the@white.house",
  "name": "CDN_in_a_Box",
  "phone": "1-202-456-1414",
  "poc": "Donald J. Trump",
  "regionId": 2,
  "shortName": "ciab",
  "state": "DC",
  "zip": "20500"
}

Response Structure

- **address**  The physical location’s street address
- **city**  The name of the city in which the physical location lies
- **comments**  Any and all human-readable comments
- **email**  The email address of the physical location’s poc
- **id**  An integral, unique identifier for the physical location
- **lastUpdated**  The date and time at which the physical location was last updated, in ISO format
- **name**  The name of the physical location
- **phone**  A phone number where the the physical location’s poc might be reached
- **poc**  The name of a “point of contact” for the physical location
- **region**  The name of the region within which the physical location lies
- **regionId**  An integral, unique identifier for the region within which the physical location lies
- **shortName**  An abbreviation of the name
- **state**  An abbreviation of the name of the state or province within which this physical location lies
- **zip**  The zip code of the physical location
# 347: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: qmKe6OqxjSUH1njlh00HwNR20YnVl0CufqCTdMBcdC1322jk2ICFQsQQ3Xu00R0WSb7h70HCfXqDC3jA1xjA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 23:39:17 GMT
Content-Length: 385

{
    "alerts": [
        {
            "text": "PhysLocation was updated.",
            "level": "success"
        }
    ],
    "response": {
        "address": "1600 Pennsylvania Avenue NW",
        "city": "Washington",
        "comments": "The White House",
        "email": "the@white.house",
        "id": 2,
        "lastUpdated": "2018-12-05 23:39:17+00",
        "name": "CDN_in_a_Box",
        "phone": "1-202-456-1414",
        "poc": "Donald J. Trump",
        "regionId": 2,
        "region": null,
        "shortName": "ciab",
        "state": "DC",
        "zip": "20500"
    }
}
```

DELETE

Deletes a physical location

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  undefined
Request Structure

Table 184: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the physical location being modified</td>
</tr>
</tbody>
</table>

# 348: Request Example

```
DELETE /api/1.4/phys_locations/3 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 349: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: KeW/tEmICwpCG8F0YMTqHdeR9J6W6Z3w/→U+HOSbeCGyaEheC1hIsWlngT3dyfH1tiu8UyzaPB6QrJyXdybBkw==
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 00:28:48 GMT
Content-Length: 67

{ "alerts": [
  {
    "text": "physLocation was deleted.",
    "level": "success"
  }
]}
```

phys_locations/trimmed

GET

Retrieves only the names of physical locations

Auth. Required  Yes
Roles Required  None

Response Type  Array

Request Structure

No parameters available

Response Structure

name  The name of the physical location

# 350: Response Example

```json
{
  "response": [
    {
      "name": "CDN_in_a_Box"
    },
    {
      "name": "Apachecon North America 2018"
    }
  ]
}
```

profileparameter

Deprecated since version 1.1: Use profileparameters instead.

POST

Create one or more profile/parameter assignments.

Auth. Required  Yes
Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

paramIds  An array of integral, unique identifiers for parameters which shall be assigned to the profile identified by profileId

profileId  The integral, unique identifier of a profile to which parameters will be assigned

replace  An optional boolean (default: false) which, if true, will cause any conflicting profile/parameter assignments to be overridden.

# 351: Request Example

POST /api/1.4/profileparameter HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 38
Content-Type: application/json

{
    "profileId": 18,
    "paramIds": [2, 3]
}

Response Structure

paramIds  An array of integral, unique identifiers for parameters which have been assigned to the profile identified by profileId

profileId  The integral, unique identifier of a profile to which parameters have been assigned

replace  An optional boolean (default: false) which, if true, caused any conflicting profile/parameter assignments to be overridden.

# 352: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json

(continues on next page)
profileparameters

GET

Deprecated since version 1.1: To get the profiles associated with a particular parameter, use the param query parameter of profiles instead. To see the parameters associated with a particular profile, refer to the params key in the response of a GET request to profiles/{{ID}} instead.

Retrieves all parameter/profile assignments.

   Auth. Required  Yes
   Roles Required  None
   Response Type   Array

Request Structure

No parameters available
Response Structure

- **lastUpdated**: The date and time at which this profile/parameter association was last modified
- **parameter**: An integral, unique identifier for a parameter assigned to profile
- **profile**: The name of the profile to which the parameter identified by parameter is assigned

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512:+bnMkRgdx4bJoGGlr3mZ1539obj3aQAP8e65FAXgywdRAUFzXZCFM6VNDn7wScXBmvFzFXo9F+MhuS
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 15:09:13 GMT
Transfer-Encoding: chunked

```json
{ "response": [  
  {  
    "lastUpdated": "2018-12-05 17:50:49+00",
    "profile": "GLOBAL",
    "parameter": 4
  },  
  {  
    "lastUpdated": "2018-12-05 17:50:49+00",
    "profile": "GLOBAL",
    "parameter": 5
  }
]}
```

**Note**: The response example for this endpoint has been truncated to only the first two elements of the resulting array, as the output was hundreds of lines long.

**POST**

Associate parameter to profile.

- **Auth. Required**: Yes
- **Roles Required**: “admin” or “operations”
Response Type  Object

Request Structure

This endpoint accepts two formats for the request payload:

Single Object Format  For assigning a single parameter to a single profile

Array Format  For making multiple assignments of parameters to profiles simultaneously

Single Object Format

parameterId  The integral, unique identifier of a parameter to assign to some profile

profileId  The integral, unique identifier of the profile to which the parameter identified by parameterId will be assigned

# 354: Request Example - Single Object Format

```
POST /api/1.4/profileparameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 36
Content-Type: application/json

{
    "profileId": 18,
    "parameterId": 1
}
```

Array Format

Caution:  Array format is broken as of the time of this writing. Follow GitHub Issue #3103 for further developments.

parameterId  The integral, unique identifier of a parameter to assign to some profile

profileId  The integral, unique identifier of the profile to which the parameter identified by parameterId will be assigned
# 355: Request Example - Array Format

```plaintext
POST /api/1.4/profileparameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 88
Content-Type: application/json

[
    {
        "profileId": 18,
        "parameterId": 2
    },
    {
        "profileId": 18,
        "parameterId": 3
    }
]
```

## Response Structure

**lastUpdated**  The date and time at which the profile/parameter assignment was last modified, in ISO format

**parameter**  Name of the parameter which is assigned to profile

**parameterId**  The integral, unique identifier of the assigned parameter

**profile**  Name of the profile to which the parameter is assigned

**profileId**  The integral, unique identifier of the profile to which the parameter identified by parameterId is assigned

---

# 356: Response Example - Single Object Format

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: eDmIwlzX44fZdxLRFH4DNa8acGAK5fQv9Y70A2eeQHFekU4evwcsQ4WeHcH013/wPTGlpyC0gwLo8LQpUxWQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 13:50:11 GMT
Content-Length: 166
```

(continues on next page)
profileparameters/{{profileID}}/{{parameterID}}

DELETE

Deletes a profile/parameter association.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profileID</td>
<td>The integral, unique identifier of the profile from which a parameter shall be removed</td>
</tr>
<tr>
<td>parameterID</td>
<td>The integral, unique identifier of the parameter which shall be removed from the profile identified by profileID</td>
</tr>
</tbody>
</table>

# 357: Request Example

```
DELETE /api/1.4/profileparameters/18/129 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**
# 358: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: JQuBqHyT9MnNwO9NSIDVQhkRtXdeAJc95W1pF2dwQeoBFmf0Y8knXm3/O/rbJDEoUC7DhUQN1acYIsqmqz4qQQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 15:00:15 GMT
Content-Length: 71

{ "alerts": [
    {
        "text": "profileParameter was deleted.",
        "level": "success"
    }
]}
```

### profiles

**GET**

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

### Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn</td>
<td>no</td>
<td>Used to filter profiles by the integral, unique identifier of the CDN to which they belong</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Filters profiles by integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Filters profiles by name</td>
</tr>
<tr>
<td>param</td>
<td>no</td>
<td>Used to filter profiles by the integral, unique identifier of a parameter associated with them</td>
</tr>
</tbody>
</table>

5.1. Traffic Ops API
# 359: Request Example

```
GET /api/1.4/profiles?name=ATS_EDGE_TIER_CACHE HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **cdn**: The integral, unique identifier of the CDN to which this profile belongs
- **cdnName**: The CDN name
- **description**: A description of the profile
- **id**: The integral, unique identifier of this profile
- **lastUpdated**: The date and time at which this profile was last updated
- **name**: The name of the profile
- **routingDisabled**: A boolean which, if `true` will disable Traffic Router’s routing to servers using this profile
- **type**: The name of the ‘type’ of the profile

# 360: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: QEpKM/DwHBRvue9K7XKrpwKFKhW6yCMQ2vSQgxE7dWFGJaqC4KOU092bsJU/5fjI9q1B+1uMT2kz6mFb1Wzp/w==
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Dec 2018 20:40:31 GMT
Content-Length: 220

{
    "response": [
        {
            "id": 9,
            "lastUpdated": "2018-12-05 17:51:00+00",
            "name": "ATS_EDGE_TIER_CACHE",
            "description": "Edge Cache - Apache Traffic Server",
            "cdnName": "CDN-in-a-Box",
        }
    ]
}
```
POST

Creates a new profile.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

**Request Structure**

- **name**  Name of the new profile
- **description**  A description of the new profile
- **cdn**  The integral, unique identifier of the CDN to which the profile shall be assigned
- **type**  The type of the profile
- **routingDisabled**  A boolean which, if `true`, will prevent the Traffic Router from directing traffic to any servers assigned this profile

### 361: Request Example

```json
POST /api/1.4/profiles HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 125
Content-Type: application/json

{
    "name": "test",
    "description": "A test profile for API examples",
    "cdn": 2,
    "type": "UNK_PROFILE",
    "routingDisabled": true
}
```
Response Structure

- **cdn**: The integral, unique identifier of the CDN to which this profile belongs
- **cdnName**: The CDN name
- **description**: A description of the profile
- **id**: The integral, unique identifier of this profile
- **lastUpdated**: The date and time at which this profile was last updated
- **name**: The name of the profile
- **routingDisabled**: A boolean which, if `true`, will disable Traffic Router’s routing to servers using this profile
- **type**: The name of the ‘type’ of the profile

---

**HTTP/1.1 200 OK**

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: UGV3PCnYBY0J3siICR0f9VVRNdUK1+9zsDDP6T9yt6t+AoHckHe6bvzo1i9to/fgHc2zz519Nc1ro4taJUDD8g==
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Dec 2018 21:24:49 GMT
Content-Length: 251
```

```
{
    "alerts": [
        {
            "text": "profile was created.",
            "level": "success"
        }
    ],
    "response": {
        "id": 16,
        "lastUpdated": "2018-12-07 21:24:49+00",
        "name": "test",
        "description": "A test profile for API examples",
        "cdnName": null,
        "cdn": 2,
        "routingDisabled": true,
        "type": "UNK_PROFILE"
    }
}
```
profiles/{{ID}}

GET

Deprecated since version 1.1: Use the id query parameter of a GET request to profiles/{{ID}} instead.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 187: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The integral, unique identifier of the profile to be retrieved</td>
</tr>
</tbody>
</table>

# 363: Request Example

GET /api/1.1/profiles/9 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- cdn The integral, unique identifier of the CDN to which this profile belongs
- cdnName The CDN name
- description A description of the profile
- id The integral, unique identifier of this profile
- lastUpdated The date and time at which this profile was last updated
- name The name of the profile
- params An array of parameters in use by this profile
  - configFile The base filename to which this parameter belongs
  - id An integral, unique identifier for this parameter
  - lastUpdated The date and time at which this parameter was last modified in ISO format
  - name The parameter name
profiles An array of profile names that use this parameter

secure When true, the parameter value is visible only to "admin"-role users

value The parameter value - if secure is true and the user does not have the "admin" role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

routingDisabled A boolean which, if true will disable Traffic Router’s routing to servers using this profile

type The name of the ‘type’ of the profile

# 364: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: LCdG7AgeHqL4wpGraaoN8ks+/gYW//h1Q2OVBEcK+T9/IC6tbJ3DWOgWX4u4dpudID5mhRwBzcYvyyXWj3qA==
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Dec 2018 21:06:30 GMT
Transfer-Encoding: chunked

{
  "response": [
    {
      "id": 9,
      "lastUpdated": "2018-12-05 17:51:00+00",
      "name": "ATS_EDGE_TIER_CACHE",
      "description": "Edge Cache - Apache Traffic Server",
      "cdnName": "CDN-in-a-Box",
      "cdn": 2,
      "routingDisabled": false,
      "type": "ATS_PROFILE",
      "params": [
        {
          "configFile": "records.config",
          "id": 9,
          "lastUpdated": null,
          "name": "CONFIG proxy.config.config_dir",
          "profiles": null,
          "secure": false,
          "value": "STRING /etc/trafficserver"
        }
      ]
    }
  ]
}
```
"configFile": "records.config",
"id": 10,
"lastUpdated": null,
"name": "CONFIG proxy.config.admin.user_id",
"profiles": null,
"secure": false,
"value": "STRING ats"
}
]]
}}

Note: The response example for this endpoint has been truncated to only the first two elements of the resulting params array, as the output was hundreds of lines long.

**PUT**

Replaces the specified profile with the one in the response payload

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
<td>The integral, unique identifier of the profile being modified</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>New of the name profile</td>
</tr>
<tr>
<td><strong>description</strong></td>
<td>A new description of the new profile</td>
</tr>
<tr>
<td><strong>cdn</strong></td>
<td>The integral, unique identifier of the CDN to which the profile shall be assigned</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>The type of the profile</td>
</tr>
</tbody>
</table>

**Warning:** Changing this will likely break something, be VERY careful when modifying this value

**routingDisabled** A boolean which, if true, will prevent the Traffic Router from directing traffic to any servers assigned this profile
# 365: Request Example

```
PUT /api/1.4/profiles/16 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 125
Content-Type: application/json

{
    "name": "test",
    "description": "A test profile for API examples",
    "cdn": 2,
    "type": "UNK_PROFILE",
    "routingDisabled": true
}
```

Response Structure

- **cdn** The integral, unique identifier of the CDN to which this profile belongs
- **cdnName** The CDN name
- **description** A description of the profile
- **id** The integral, unique identifier of this profile
- **lastUpdated** The date and time at which this profile was last updated
- **name** The name of the profile
- **routingDisabled** A boolean which, if true will disable Traffic Router’s routing to servers using this profile
- **type** The name of the ‘type’ of the profile

# 366: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-...
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Pnf+G9G3+/edt4bBPsvyGZHsN2aFEqphaGSminjR1RmMpWtuLAA20WZDUo3nX0Q081c2GCuFuEh9wMF2Vjeppg=
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Dec 2018 21:45:06 GMT
```

(continues on next page)
DELETE

Allows user to delete a profile.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

Request Structure

Table 189: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the profile being deleted</td>
</tr>
</tbody>
</table>

# 367: Request Example

```
DELETE /api/1.4/profiles/16 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure
# 368: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: HNmJkZaNW9yil08/\-3Tnq25F1lH6Rp+jgp3KI46FZdojLYcu+8jEhDL1lokoirdrHyU4Rl3hjC10uN7PVvWDA==
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Dec 2018 21:55:33 GMT
Content-Length: 62

{
  "alerts": [
    {
      "text": "profile was deleted.",
      "level": "success"
    }
  ]
}
```

`profiles/{{ID}}/parameters`

**GET**

Deprecated since version 1.1: Refer to the `params` key in the response of `profiles/{{ID}}` instead

Retrieves all parameters assigned to the profile.

Auth. Required  Yes
Roles Required  None
Response Type  Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the profile for which parameters will be listed</td>
</tr>
</tbody>
</table>
# 369: Request Example

GET /api/1.4/parameters/profile/GLOBAL HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **configFile**: The base filename to which this parameter belongs
- **id**: An integral, unique identifier for this parameter
- **lastUpdated**: The date and time at which this parameter was last modified in ISO format
- **name**: The parameter name
- **profiles**: An array of profile names that use this parameter
- **secure**: When true, the parameter value is visible only to “admin”-role users
- **value**: The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

# 370: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 21:08:56 GMT
Content-Length: 542

```
{
    "response": {
        "configFile": "global",
        "id": 4,
        "lastUpdated": "2018-12-05 17:50:49+00",
        "name": "tm.instance_name",
        "secure": false,
    }
}
```

(continues on next page)
POST

Deprecated since version 1.1: Use `profiles/name/{{name}}/parameters` instead

Associate parameters to a profile. If the parameter does not exist, create it and associate to the profile. If the parameter already exists, associate it to the profile. If the parameter is already associated with the profile, keep the association.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier for the profile to which parameters will be assigned</td>
</tr>
</tbody>
</table>
This endpoint accepts two formats for the request payload:

**Single Object Format**  For assigning a single parameter to a single profile

**Parameter Array Format**  For making multiple assignments of parameters to profiles simultaneously

---

**Warning:** Most API endpoints dealing with parameters treat `secure` as a boolean value, whereas this endpoint takes the legacy approach of treating it as an integer. Be careful when passing data back and forth, as boolean values will **not** be accepted by this endpoint!

---

### Single Parameter Format

- **configFile**  The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

- **name**  Parameter name

- **secure**  An integer which, when any number other than 0, will prohibit users who do not have the “admin” role from viewing the parameter’s value (at the time of this writing the obfuscation value is defined to be "********")

- **value**  Parameter value

---

# 371: Response Example - Single Parameter Format

```json
POST /api/1.1/profiles/18/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 99
Content-Type: application/json

{
    "name": "test",
    "configFile": "quest",
    "value": "A test parameter for API examples",
    "secure": 0
}
```

### Parameter Array Format

- **configFile**  The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

- **name**  Parameter name
secure  An integer which, when any number other than 0, will prohibit users who do not have the “admin” role from viewing the parameter’s value (at the time of this writing the obfuscation value is defined to be "********")

value  Parameter value

# 372: Request Example - Parameter Array Format

```plaintext
POST /api/1.1/profiles/18/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 212
Content-Type: application/json

[
   {
      "name": "test",
      "configFile": "quest",
      "value": "A test parameter for API examples",
      "secure": 0
   },
   {
      "name": "foo",
      "configFile": "bar",
      "value": "Another test parameter for API examples",
      "secure": 0
   }
]
```

Response Structure

parameters  An array of objects representing the parameters which have been assigned

configFile  The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

name  Parameter name

secure  An integer which, when any number other than 0, will prohibit users who do not have the “admin” role from viewing the parameter’s value (at the time of this writing the obfuscation value is defined to be "********")

value  Parameter value

profileId  The integral, unique identifier for the profile to which the parameter(s) have been assigned

profileName  Name of the profile to which the parameter(s) have been assigned
# 373: Response Example - Single Parameter Format

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: R2QUyCaNvKvVv/PNVNmEd/ma5h/iP1fMJ1qhV+x2jE/
\-zNpHJ1KVxt6s3bt8nnHv61DF/6t5k1rzQ0mbW5U8bpg==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 14:45:28 GMT
Content-Length: 253

{
  "alerts": [
    {
      "text": "Assign parameters successfully to profile test",
      "level": "success"
    }
  ],
  "response": {
    "parameters": [
      {
        "configFile": "quest",
        "name": "test",
        "secure": 0,
        "value": "A test parameter for API examples",
        "id": 126
      }
    ],
    "profileId": 18,
    "profileName": "test"
  }
}
```

**Note:** The format of the request does not affect the format of the response. parameters will be an array either way.

```
profiles/{{ID}}/unassigned_parameters
```

**Warning:** There are very few good reasons to use this endpoint - be sure not limit said use.
GET

Retrieves all parameters NOT assigned to the specified profile.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

### Request Structure

Table 192: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the profile for which unassigned parameters will be listed</td>
</tr>
</tbody>
</table>

# 374: Request Example

```
GET /api/1.4/profiles/9/unassigned_parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

- **configFile** The base filename to which this parameter belongs
- **id** An integral, unique identifier for this parameter
- **lastUpdated** The date and time at which this parameter was last modified in ISO format
- **name** The parameter name
- **profiles** An array of profile names that use this parameter
- **secure** When true, the parameter value is visible only to “admin”-role users
- **value** The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

# 375: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
```
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: iO7YHU+0spCPSaR6cDrVlQwxDSS1GoSyi8K6ng4eemuxqOxB9fPdpBpXN8w+xmxf2Zw8MLXHv5S6cf
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 21:37:50 GMT
Transfer-Encoding: chunked

```json
{ "response": [
    {
        "configFile": "parent.config",
        "id": 1,
        "lastUpdated": "2018-12-05 17:50:47+00",
        "name": "mso.parent_retry",
        "secure": false,
        "value": "simple_retry"
    },
    {
        "configFile": "parent.config",
        "id": 2,
        "lastUpdated": "2018-12-05 17:50:47+00",
        "name": "mso.parent_retry",
        "secure": false,
        "value": "unavailable_server_retry"
    }
]}
```

**Note:** The response example for this endpoint has been truncated to only the first two elements of the resulting array, as the output was hundreds of lines long.

---

**profiles/name/{{name}}/copy/{{copy}}**

**POST**

Copy profile to a new profile. The new profile name must not exist.

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object
Request Structure

Table 193: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the new profile</td>
</tr>
<tr>
<td>copy</td>
<td>The name of profile from which the copy will be made</td>
</tr>
</tbody>
</table>

# 376: Request Example

```
POST /api/1.4/profiles/name/GLOBAL_copy/copy/GLOBAL HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **description** The description of the new profile
- **id** An integral, unique identifier for the new profile
- **idCopyFrom** The integral, unique identifier for the profile from which the copy was made
- **name** The name of the new profile
- **profileCopyFrom** The name of the profile from which the copy was made

# 377: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type,
                            Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Fri, 07 Dec 2018 22:03:54 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Sat, 08 Dec 2018 02:03:54 GMT;
           path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: r6V9viEZu11WCns0AUGEx1MtxjjXiu8SZV0tSQjeq7ZJDL15s8fMmjJdR/
                     HRWduHn7Ax6GzYhoKwnIjMyc7ZWg==
Content-Length: 252
```
 profiles/name/{{name}}/parameters

GET

Retrieves all parameters associated with a given profile

Auth. Required  Yes
Roles Required  None
Response Type  None

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the profile for which parameters will be listed</td>
</tr>
</tbody>
</table>

# 378: Request Example

GET /api/1.4/profiles/name/GLOBAL/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

**configFile**  The *base* filename to which this parameter belongs
id  An integral, unique identifier for this parameter

lastUpdated  The date and time at which this parameter was last modified in ISO format

name  The parameter name

profiles  An array of profile names that use this parameter

secure  When true, the parameter value is visible only to “admin”-role users

value  The parameter value - if secure is true and the user does not have the “admin” role this will be obfuscated (at the time of this writing the obfuscation value is defined to be "********") but not missing

**Response Example**

```json
{
  "response": [
    {
      "configFile": "global",
      "id": 4,
      "lastUpdated": "2018-12-05 17:50:49+00",
      "name": "tm.instance_name",
      "secure": false,
      "value": "Traffic Ops CDN"
    },
    {
      "configFile": "global",
      "id": 5,
      "lastUpdated": "2018-12-05 17:50:49+00",
      "name": "tm.toolname",
      "secure": false,
      "value": "Traffic Ops"
    },
    {
      "configFile": "global",
      "id": 6,
      "lastUpdated": "2018-12-05 17:50:51+00",
      "name": "use_tenancy",
    }
  ]
}
```

(continues on next page)
POST

Associate parameters to a profile. If the parameter does not exist, create it and associate to the profile. If the parameter already exists, associate it to the profile. If the parameter is already associated with the profile, keep the association.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the profile to which parameters will be assigned</td>
</tr>
</tbody>
</table>

This endpoint accepts two formats for the request payload:

**Single Parameter Format**  Specify a single parameter to assign to the specified profile

**Parameter Array Format**  Specify multiple parameters to assign to the specified profile

**Warning:** Most API endpoints dealing with parameters treat `secure` as a boolean value, whereas this endpoint takes the legacy approach of treating it as an integer. Be careful when passing data back and forth, as boolean values will **not** be accepted by this endpoint!
Single Parameter Format

**configFile**  The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

**name**  Parameter name

**secure**  An integer which, when any number other than 0, will prohibit users who do not have the “admin” role from viewing the parameter’s **value** (at the time of this writing the obfuscation value is defined to be "********")

**value**  Parameter value

# 379: Request Example - Single Parameter Format

```
POST /api/1.4/profiles/name/test/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 99
Content-Type: application/json

{
    "name": "test",
    "configFile": "quest",
    "value": "A test parameter for API examples",
    "secure": 0
}
```

Parameter Array Format

**configFile**  The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

**name**  Parameter name

**secure**  An integer which, when any number other than 0, will prohibit users who do not have the “admin” role from viewing the parameter’s **value** (at the time of this writing the obfuscation value is defined to be "********")

**value**  Parameter value

# 380: Request Example - Parameter Array Format

```
POST /api/1.4/profiles/name/test/parameters HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 212

(continues on next page)
```
Response Structure

parameters An array of objects representing the parameters which have been assigned

configFile The base filename of the configuration file to which this parameter shall belong e.g. “foo” not “/path/to/foo”

name Parameter name

secure An integer which, when any number other than 0, will prohibit users who do not have the “admin” role from viewing the parameter’s value (at the time of this writing the obfuscation value is defined to be "*******")

value Parameter value

profileId The integral, unique identifier for the profile to which the parameter(s) have been assigned

profileName Name of the profile to which the parameter(s) have been assigned

# 381: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-...
Note: The format of the request does not affect the format of the response. parameters will be an array either way.

```
profiles/{{profile}}/configfiles/ats/{{filename}}
```

See also:

The servers/{{server}}/configfiles/ats endpoint

GET

Returns the requested configuration file for download.
Auth. Required Yes

Roles Required “operations”

Response Type NOT PRESENT - endpoint returns custom text/plain response (represents the contents of the requested configuration file)

Request Structure

Table 196: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile</td>
<td>string or integer</td>
<td>Either the name or integral, unique, identifier of a profile</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The name of a configuration file used by profile</td>
</tr>
</tbody>
</table>

# 382: Request Example

GET /api/1.4/profiles/ATS_MID_TIER_CACHE/configfiles/ats/volume.config HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

Note: If the file identified by filename doesn’t exist at the profile, a JSON response will be returned and the alerts array will contain a "level": "error" node which suggests other scopes to check for the configuration file.

# 383: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: text/plain; charset=UTF-8
Date: Thu, 15 Nov 2018 15:23:44 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 19:23:44 GMT;
Vary: Accept-Encoding
profiles/trimmed

GET

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available

Response Structure

name The name of the profile

# 384: Response Example

```json
{
  "response": [
    "name": "The name of the profile"
  ]
}
```
regions

GET

Retrieves information about regions

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

Table 197: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>division</td>
<td>no</td>
<td>Filter regions by the integral, unique identifier of the division which contains them</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Filter regions by integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Filter regions by name</td>
</tr>
</tbody>
</table>

# 385: Request Example

GET /api/1.4/regions?division=1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Response Structure

divisionName  The name of the division which contains this region

divisionId  The integral, unique identifier of the division which contains this region

id  An integral, unique identifier for this region

lastUpdated  The date and time at which this region was last updated, in ISO format

name  The region name

# 386: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512:nSYbR+frXaxhY17dWgf0Udo2A5iXEnwvED1CPbk7ZNWK03I3TOhtmCQx9ABnJ6xKYnl6t6EKM6opVTR2X
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 01:58:38 GMT
Content-Length: 117

{
    "response": [
        {
            "divisionName": "Quebec",
            "division": 1,
            "id": 2,
            "lastUpdated": "2018-12-05 17:50:58+00",
            "name": "Montreal"
        }
    ]
}
```

POST

Creates a new region

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object
Request Structure

**division**  The integral, unique identifier of the division which shall contain the new region

**divisionName**  The name of the division which shall contain the new region

**name**  The name of the region

---

# 387: Request Example

```
POST /api/1.4/regions HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 65
Content-Type: application/json

{
    "name": "Manchester",
    "division": "4",
    "divisionName": "England"
}
```

---

Response Structure

**divisionName**  The name of the division which contains this region

**divisionId**  The integral, unique identifier of the division which contains this region

**id**  An integral, unique identifier for this region

**lastUpdated**  The date and time at which this region was last updated, in ISO format

**name**  The region name

---

# 388: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
```

---

1 The only “division” key that actually matters in the request body is **division**; **divisionName** is not validated and has no effect - particularly not the effect of re-naming the division - beyond changing the name in the API response to this request. Subsequent requests will reveal the true name of the division. Note that if **divisionName** is not present in the request body it will be **null** in the response, but again further requests will show the true division name (provided it has been assigned to a division).
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512:
\textasciitilde ezxk+iP7o7KE7zpWmGc0j8nz5k+1wAzY0HiNiA2xswTQrt+N+6CqQqUV2r9G1HAsPNr0HF2PhYs/
\textasciitilde Xr7DrY0w0A==
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 02:14:45 GMT
Content-Length: 178

{
    "alerts": [
        {
            "text": "region was created.",
            "level": "success"
        }
    ],
    "response": {
        "divisionName": "England",
        "division": 3,
        "id": 5,
        "lastUpdated": "2018-12-06 02:14:45+00",
        "name": "Manchester"
    }
}

regions/{{ID}}

GET

Retrieves a specific region

Deprecated since version 1.1: Use the \texttt{id} query parameter of a GET request to the \texttt{regions} endpoint

\begin{verbatim}
Auth. Required  Yes
Roles Required  None
Response Type  Array
\end{verbatim}

Request Structure

Table 198: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the region to inspect</td>
</tr>
</tbody>
</table>
# 389: Request Example

```
GET /api/1.4/regions/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>Region ID.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Region name.</td>
</tr>
<tr>
<td>division</td>
<td>string</td>
<td>Division ID.</td>
</tr>
<tr>
<td>divisionName</td>
<td>string</td>
<td>Division name.</td>
</tr>
</tbody>
</table>

# 390: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: nSYbR+frRXaxhYl7dWgf0Udo2AsiXEnwvED1CPbk7ZNWK03I3T0htmCQx9ABnJJ6xKYNlt6EKMeopVT
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 02:07:17 GMT
Content-Length: 117

```{ "response": [ 
   {
       "divisionName": "Quebec",
       "division": 1,
       "id": 2,
       "lastUpdated": "2018-12-05 17:50:58+00",
       "name": "Montreal"
   }
]}
```

**PUT**

Updates a region

---

5.1. Traffic Ops API

---
Authentication Required: Yes
Role(s) Required: admin or oper

Request Structure

Table 199: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the region to update</td>
</tr>
</tbody>
</table>

**division**  The new integral, unique identifier of the division which shall contain the region¹

**divisionName**  The new name of the division which shall contain the region¹

**name**  The new name of the region

# 391: Request Example

```plaintext
PUT /api/1.4/regions/5 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 60
Content-Type: application/json

{
    "name": "Leeds",
    "division": 3,
    "divisionName": "England"
}
```

Response Structure

**divisionName**  The name of the division which contains this region

**divisionId**  The integral, unique identifier of the division which contains this region

**id**  An integral, unique identifier for this region

**lastUpdated**  The date and time at which this region was last updated, in ISO format

¹ The only “division” key that actually matters in the request body is `division`; `divisionName` is not validated and has no effect - particularly not the effect of re-naming the division - beyond changing the name in the API response to this request. Subsequent requests will reveal the true name of the division. Note that if `divisionName` is not present in the request body it will be `null` in the response, but again further requests will show the true division name (provided it has been assigned to a division).
name  The region name

# 392: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: ...
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 02:23:40 GMT
Content-Length: 173

{
  "alerts": [
    {
      "text": "region was updated.",
      "level": "success"
    }
  ],
  "response": {
    "divisionName": "England",
    "division": 3,
    "id": 5,
    "lastUpdated": "2018-12-06 02:23:40+00",
    "name": "Leeds"
  }
}
```

### regions/:region_name/phys_locations

Deprecated since version 1.1: Instead specify the regionId field in the body of a POST request to `phys_locations`.

**POST**

Creates a new physical location within the specified region.

**Auth. Required**  Yes

**Roles Required**  "admin" or "operations"

**Response Type**  Object
Request Structure

Table 200: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the region in which the new physical location will be created</td>
</tr>
<tr>
<td>address</td>
<td>The physical location’s street address</td>
</tr>
<tr>
<td>city</td>
<td>The name of the city in which the physical location lies</td>
</tr>
<tr>
<td>comments</td>
<td>An optional string for containing any and all human-readable comments</td>
</tr>
<tr>
<td>email</td>
<td>An optional string containing email address of the physical location’s poc</td>
</tr>
<tr>
<td>name</td>
<td>An optional name of the physical location</td>
</tr>
<tr>
<td>phone</td>
<td>An optional string containing the phone number where the physical location’s poc might be reached</td>
</tr>
<tr>
<td>poc</td>
<td>The name of a “point of contact” for the physical location</td>
</tr>
<tr>
<td>shortName</td>
<td>An abbreviation of the name</td>
</tr>
<tr>
<td>state</td>
<td>An abbreviation of the name of the state or province within which this physical location lies</td>
</tr>
<tr>
<td>zip</td>
<td>The zip code of the physical location</td>
</tr>
</tbody>
</table>

# 393: Request Structure

```
POST /api/1.4/regions/Greater_London/phys_locations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 326
Content-Type: application/json

{
    "address": "Buckingham Palace",
    "city": "London",
    "comments": "Buckingham Palace",
    "email": "steve.kingstone@royal.gsx.gov.uk",
    "name": "Great_Britain",
    "phone": "0-843-816-6276",
    "poc": "Her Majesty The Queen Elizabeth Alexandra Mary Windsor II",
    "regionId": 3,
    "shortName": "uk",
    "state": "Westminster",
    "zip": "SW1A 1AA"
}
```
Response Structure

- **address** The physical location’s street address
- **city** The name of the city in which the physical location lies
- **comments** Any and all human-readable comments
- **email** The email address of the physical location’s poc
- **id** An integral, unique identifier for the physical location
- **name** The name of the physical location
- **phone** A phone number where the the physical location’s poc might be reached
- **poc** The name of a “point of contact” for the physical location
- **regionId** An integral, unique identifier for the region within which the physical location lies
- **regionName** The name of the region within which the physical location lies
- **shortName** An abbreviation of the name
- **state** An abbreviation of the name of the state or province within which this physical location lies
- **zip** The zip code of the physical location

# 394: Response Example

```json
{
  "response": {
    "regionName": "Greater_London",
    "poc": "Her Majesty The Queen Elizabeth Alexandra Mary Windsor II",
    "name": "Great_Britain",
    "comments": "Buckingham Palace",
  }
}
```

(continues on next page)
"phone": "0-843-816-6276",
"state": "Westminster",
"regionId": 3,
"email": "steve.kingstone@royal.gsx.gov.uk",
"zip": "SW1A 1AA",
"city": "London",
"id": 4,
"address": "Buckingham Palace",
"shortName": "uk"
}}

## roles

### GET

Retrieves all user roles.

**Auth. Required** Yes  
**Roles Required** None  
**Response Type** Array

### Request Structure

No parameters available.

### Response Structure

- **description**: A description of the role
- **id**: The integral, unique identifier for this role
- **name**: The name of the role
- **privLevel**: An integer that allows for comparison between roles

### # 395: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
```
servercheck

**Caution:** This page is a stub! Much of it may be missing or just downright wrong - it needs a lot of love from people with the domain knowledge required to update it.

**POST**

Post a server check result to the “serverchecks” table.

**Auth. Required** Yes

**Roles Required** None\(^1\)

**Response Type** undefined

**Request Structure**

- **host_name** The hostname of the server to which this “servercheck” refers
- **id** An integral, unique identifier for this “servercheck”
- **servercheck_short_name** The short name for the check for which the “servercheck” has been measured

---

\(^1\) No roles are required to use this endpoint, however access is controlled by username. Only the reserved user extension is permitted the use of this endpoint.
The value of the “servercheck”

# 396: Request Example

```plaintext
POST /api/1.1/servercheck HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 113
Content-Type: application/json

{
    "id": 1,
    "host_name": "edge",
    "servercheck_short_name": "test",
    "value": "A test servercheck for API examples"
}
```

Response Structure

# 397: Response Example

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

servers

GET

Retrieves properties of all servers across all CDNs.

Auth. Required Yes
Roles Required None
Response Type Array
Request Structure

Table 201: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachegroup</td>
<td>no</td>
<td>Return only those servers within the Cache Group identified by this integral, unique identifier</td>
</tr>
<tr>
<td>dsId</td>
<td>no</td>
<td>Return only those servers assigned to the Delivery Service identified by this integral, unique identifier</td>
</tr>
<tr>
<td>hostName</td>
<td>no</td>
<td>Return only those servers that have this (short) hostname</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the server with this integral, unique identifier</td>
</tr>
<tr>
<td>profileId</td>
<td>no</td>
<td>Return only those servers that are using the profile identified by this integral, unique identifier</td>
</tr>
<tr>
<td>status</td>
<td>no</td>
<td>Return only those servers with this status - see Health Protocol</td>
</tr>
<tr>
<td>type</td>
<td>no</td>
<td>Return only servers of this 'type'</td>
</tr>
</tbody>
</table>

# 398: Request Example

GET /api/1.4/servers?hostName=mid HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **cachegroup**  The name of the Cache Group to which this server belongs
- **cachegroupId**  The integral, unique identifier of the Cache Group to which this server belongs
- **cdnId**  The integral, unique identifier of the CDN to which the server belongs
- **cdnName**  Name of the CDN to which the server belongs
- **domainName**  The domain part of the server’s Fully Qualified Domain Name (FQDN)
- **guid**  An identifier used to uniquely identify the server
  - Deprecated since version 1.1: This is a legacy key which only still exists for compatibility reasons - it should always be null
- **hostName**  The (short) hostname of the server
- **httpsPort**  The port on which the server listens for incoming HTTPS connections/requests
- **id**  An integral, unique identifier for this server
iloIpAddress  The IPv4 address of the server’s Integrated Lights-Out (ILO) service

iloIpGateway  The IPv4 gateway address of the server’s ILO service

iloIpNetmask  The IPv4 subnet mask of the server’s ILO service

iloPassword  The password of the of the server’s ILO service user - displays as simply ******** if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)

iloUsername  The user name for the server’s ILO service

interfaceMtu  The Maximum Transmission Unit (MTU) to configured on

interfaceName  The name of the primary network interface used by the server

ip6Address  The IPv6 address and subnet mask of interfaceName

ip6Gateway  The IPv6 address of the gateway used by interfaceName

ipAddress  The IPv4 address of interfaceName

ipGateway  The IPv4 address of the gateway used by interfaceName

ipNetmask  The IPv4 subnet mask used by interfaceName

lastUpdated  The date and time at which this server description was last modified

mgmtIpAddress  The IPv4 address of some network interface on the server used for ‘management’

mgmtIpGateway  The IPv4 address of a gateway used by some network interface on the server used for ‘management’

mgmtIpNetmask  The IPv4 subnet mask used by some network interface on the server used for ‘management’

offlineReason  A user-entered reason why the server is in ADMIN_DOWN or OF-FLINE status

physLocation  The name of the physical location where the server resides

physLocationId  An integral, unique identifier for the physical location where the server resides

profile  The name of the profile this server uses

profileDesc  A description of the profile this server uses

profileId  An integral, unique identifier for the profile used by this server

revalPending  A boolean value which, if true indicates that this server has pending content invalidation/revalidation

rack  A string indicating “server rack” location

For more information see the Wikipedia page on Lights-Out management.
**routerHostName** The human-readable name of the router responsible for reaching this server

**routerPortName** The human-readable name of the port used by the router responsible for reaching this server

**status** The status of the server

See also:

*Health Protocol*

**statusId** The integral, unique identifier of the status of this server

See also:

*Health Protocol*

**tcpPort** The port on which this server listens for incoming TCP connections

Note: This is typically thought of as synonymous with “HTTP port”, as the port specified by `httpsPort` may also be used for incoming TCP connections.

**type** The name of the ‘type’ of this server

**typeId** The integral, unique identifier of the ‘type’ of this server

**updPending** A boolean value which, if true, indicates that the server has updates of some kind pending, typically to be acted upon by Traffic Ops ORT

**xmppId** An identifier to be used in XMPP communications with the server - in nearly all cases this will be the same as `hostName`

**xmppPasswd** The password used in XMPP communications with the server

### # 399: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: WyapQctUlhzjzEALka5qBiZRXZ58Ml6MJS7wBeGyJS2UzbL3W61N/4kvAZtPrP4qM0WQWz6Jjbf7Y5lNRA5U==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 16:13:31 GMT
Content-Length: 939

{ "response": [ (continues on next page) }
```
```json
{
    "cachegroup": "CDN_in_a_Box_Mid",
    "cachegroupId": 6,
    "cdnId": 2,
    "cdnName": "CDN-in-a-Box",
    "domainName": "infra.ciab.test",
    "guid": null,
    "hostName": "mid",
    "httpsPort": 443,
    "id": 10,
    "iloIpAddress": "",
    "iloIpGateway": "",
    "iloIpNetmask": "",
    "iloPassword": "",
    "iloUsername": "",
    "interfaceMtu": 1500,
    "interfaceName": "eth0",
    "ip6Address": "fc01:9400:1000:8::120",
    "ip6Gateway": "fc01:9400:1000:8::1",
    "ipAddress": "172.16.239.120",
    "ipGateway": "172.16.239.1",
    "ipNetmask": "255.255.255.0",
    "lastUpdated": "2018-12-05 18:45:05+00",
    "mgmtIpAddress": "",
    "mgmtIpGateway": "",
    "mgmtIpNetmask": "",
    "offlineReason": "",
    "physLocation": "Apachecon North America 2018",
    "physLocationId": 1,
    "profile": "ATS_MID_TIER_CACHE",
    "profileDesc": "Mid Cache - Apache Traffic Server",
    "profileId": 10,
    "rack": "",
    "revalPending": false,
    "routerHostName": "",
    "routerPortName": "",
    "status": "REPORTED",
    "statusId": 3,
    "tcpPort": 80,
    "type": "MID",
    "typeId": 12,
    "updPending": false,
    "xmppId": "mid",
    "xmppPasswd": ""
}
]}
```
POST

Allows a user to create a new server.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

- **cachegroupId** The integral, unique identifier of the Cache Group to which this server shall belong
- **cdnId** The integral, unique identifier of the CDN to which the server shall belong
- **domainName** The domain part of the server’s Fully Qualified Domain Name (FQDN)
- **hostName** The (short) hostname of the server
- **httpsPort** An optional port number on which the server listens for incoming HTTPS connections/requests
- **iloIpAddress** An optional IPv4 address of the server’s Integrated Lights-Out (ILO) service
- **iloIpGateway** An optional IPv4 gateway address of the server’s ILO service
- **iloIpNetmask** An optional IPv4 subnet mask of the server’s ILO service
- **iloPassword** An optional string containing the password of the server’s ILO service - displays as simply ****** if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)
- **iloUsername** An optional string containing the user name for the server’s ILO service
- **interfaceMtu** The Maximum Transmission Unit (MTU) to configured on *interfaceName*

**Note:** In virtually all cases this ought to be 1500. Further note that the only acceptable values are 1500 and 9000.

- **interfaceName** The name of the primary network interface used by the server
- **ip6Address** An optional IPv6 address and subnet mask of *interfaceName*
- **ip6Gateway** An optional IPv6 address of the gateway used by *interfaceName*
- **ipAddress** The IPv4 address of *interfaceName*
- **ipGateway** The IPv4 address of the gateway used by *interfaceName*
ipNetmask  The IPv4 subnet mask used by interfaceName

gmtIpAddress  An optional IPv4 address of some network interface on the server used for ‘management’

gmtIpGateway  An optional IPv4 address of a gateway used by some network interface on the server used for ‘management’

gmtIpNetmask  An optional IPv4 subnet mask used by some network interface on the server used for ‘management’

physLocationId  An integral, unique identifier for the physical location where the server resides

profileId  An integral, unique identifier for the profile used by this server

revalPending  A boolean value which, if true indicates that this server has pending content invalidation/revalidation

rack  An optional string indicating “server rack” location

routerHostName  An optional string containing the human-readable name of the router responsible for reaching this server

routerPortName  An optional string containing the human-readable name of the port used by the router responsible for reaching this server

statusId  The integral, unique identifier of the status of this server

See also:

Health Protocol

tcpPort  An optional port number on which this server listens for incoming TCP connections

Note:  This is typically thought of as synonymous with “HTTP port”, as the port specified by httpsPort may also be used for incoming TCP connections.

typeId  The integral, unique identifier of the ‘type’ of this server

updPending  A boolean value which, if true, indicates that the server has updates of some kind pending, typically to be acted upon by Traffic Ops ORT

xmppId  An optional identifier to be used in XMPP communications with the server - in nearly all cases this should be the same as hostName

xmppPasswd  An optional password used in XMPP communications with the server

# 400: Request Example

```
POST /api/1.4/servers HTTP/1.1
Host: trafficops.infra.ciab.test
```
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 599
Content-Type: application/json

{
    "cachegroupId": 6,
    "cdnId": 2,
    "domainName": "infra.ciab.test",
    "hostName": "test",
    "httpsPort": 443,
    "iloIpAddress": "",
    "iloIpGateway": "",
    "iloIpNetmask": "",
    "iloPassword": "",
    "iloUsername": "",
    "interfaceMtu": 1500,
    "interfaceName": "eth0",
    "ip6Address": '::1",
    "ip6Gateway": '::2",
    "ipAddress": "0.0.0.1",
    "ipGateway": "0.0.0.2",
    "ipNetmask": "255.255.255.0",
    "mgmtIpAddress": "",
    "mgmtIpGateway": "",
    "mgmtIpNetmask": "",
    "offlineReason": "",
    "physLocationId": 1,
    "profileId": 10,
    "routerHostName": "",
    "routerPortName": "",
    "statusId": 3,
    "tcpPort": 80,
    "typeId": 12,
    "updPending": false
}

Response Structure

**cachegroup**  The name of the Cache Group to which this server belongs

**cachegroupId**  The integral, unique identifier of the Cache Group to which this server belongs

**cdnId**  The integral, unique identifier of the CDN to which the server belongs

**cdnName**  Name of the CDN to which the server belongs
domainName  The domain part of the server’s Fully Qualified Domain Name (FQDN)

guid  An identifier used to uniquely identify the server

   Deprecated since version 1.1: This is a legacy key which only still exists for compatibility reasons - it should always be null

hostName  The (short) hostname of the server

httpsPort  The port on which the server listens for incoming HTTPS connections/requests

id  An integral, unique identifier for this server

iloIpAddress  The IPv4 address of the server’s Integrated Lights-Out (ILO) service

iloIpGateway  The IPv4 gateway address of the server’s ILO service

iloIpNetmask  The IPv4 subnet mask of the server’s ILO service

iloPassword  The password of the of the server’s ILO service user - displays as simply ****** if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)

iloUsername  The user name for the server’s ILO service

interfaceMtu  The Maximum Transmission Unit (MTU) to configured on
   interfaceName

interfaceName  The name of the primary network interface used by the server

ip6Address  The IPv6 address and subnet mask of interfaceName

ip6Gateway  The IPv6 address of the gateway used by interfaceName

ipAddress  The IPv4 address of interfaceName

ipGateway  The IPv4 address of the gateway used by interfaceName

ipNetmask  The IPv4 subnet mask used by interfaceName

lastUpdated  The date and time at which this server description was last modified

mgmtIpAddress  The IPv4 address of some network interface on the server used for ‘management’

mgmtIpGateway  The IPv4 address of a gateway used by some network interface on the server used for ‘management’

mgmtIpNetmask  The IPv4 subnet mask used by some network interface on the server used for ‘management’

offlineReason  A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status

physLocation  The name of the physical location where the server resides
physLocationId  An integral, unique identifier for the physical location where the server resides

profile  The name of the profile this server uses

profileDesc  A description of the profile this server uses

profileId  An integral, unique identifier for the profile used by this server

revalPending  A boolean value which, if true indicates that this server has pending content invalidation/revalidation

rack  A string indicating “server rack” location

routerHostName  The human-readable name of the router responsible for reaching this server

routerPortName  The human-readable name of the port used by the router responsible for reaching this server

status  The status of the server

See also:

Health Protocol

statusId  The integral, unique identifier of the status of this server

See also:

Health Protocol

tcpPort  The port on which this server listens for incoming TCP connections

Note:  This is typically thought of as synonymous with “HTTP port”, as the port specified by httpsPort may also be used for incoming TCP connections.

type  The name of the ‘type’ of this server

typeId  The integral, unique identifier of the ‘type’ of this server

updPending  A boolean value which, if true, indicates that the server has updates of some kind pending, typically to be acted upon by Traffic Ops ORT

xmppId  An identifier to be used in XMPP communications with the server - in nearly all cases this will be the same as hostName

xmppPasswd  The password used in XMPP communications with the server

# 401: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie

(continues on next page)
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: mcGmmu5ONDg3jmvlkItcw6jxiT1ecmPYuj2fmKiZrn5ThKjsSadeJIynaeOK0XVUjHuYHtdynSqx
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 17:44:04 GMT
Content-Length: 850

{
    "alerts": [
        {
            "text": "server was created."
        },
        "level": "success"
    ],
    "response": {
        "cacheGroup": "null",
        "cacheGroupId": 6,
        "cdnId": 2,
        "cdnName": "null",
        "domainName": "infra.ciab.test",
        "guid": "null",
        "hostName": "test",
        "httpsPort": 443,
        "id": 13,
        "iloIpAddress": "",
        "iloIpGateway": "",
        "iloIpNetmask": "",
        "iloPassword": "",
        "iloUsername": "",
        "interfaceMtu": 1500,
        "interfaceName": "eth0",
        "ip6Address": "::1",
        "ip6Gateway": "::2",
        "ipAddress": "0.0.0.1",
        "ipGateway": "0.0.0.2",
        "ipNetmask": "255.255.255.0",
        "lastUpdated": "2018-12-10 17:44:04+00",
        "mgmtIpAddress": "",
        "mgmtIpGateway": "",
        "mgmtIpNetmask": "",
        "offlineReason": "",
        "physLocation": "null",
        "physLocationId": 1,
        "profile": "null",
        "profileDesc": "null",
        "profileId": 10,
        "rack": "null",
    }
servers/hostname/{{name}}/details

Deprecated since version 1.1: Use the hostName query parameter of the servers endpoint instead.

GET

Retrieves the details of a server.

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

Table 202: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The (short) hostname of the server being inspected</td>
</tr>
</tbody>
</table>

Response Structure

- **cachegroup**  The name of the Cache Group to which this server belongs
- **cdnName**  Name of the CDN to which the server belongs
- **deliveryservices**  An array of integral, unique identifiers for Delivery Services to which this server belongs
- **domainName**  The domain part of the server’s Fully Qualified Domain Name (FQDN)
guid  An identifier used to uniquely identify the server

   Deprecated since version 1.1: This is a legacy key which only still exists for compatibility reasons - it should always be null

hostName  The (short) hostname of the server

httpsPort  The port on which the server listens for incoming HTTPS connections/requests

id  An integral, unique identifier for this server

iloIpAddress  The IPv4 address of the server’s Integrated Lights-Out (ILO) service

iloIpGateway  The IPv4 gateway address of the server’s ILO service

iloIpNetmask  The IPv4 subnet mask of the server’s ILO service

iloPassword  The password of the of the server’s ILO service user - displays as simply ****** if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)

iloUsername  The user name for the server’s ILO service

interfaceMtu  The Maximum Transmission Unit (MTU) to configured on

interfaceName  The name of the primary network interface used by the server

ip6Address  The IPv6 address and subnet mask of interfaceName

ip6Gateway  The IPv6 address of the gateway used by interfaceName

ipAddress  The IPv4 address of interfaceName

ipGateway  The IPv4 address of the gateway used by interfaceName

ipNetmask  The IPv4 subnet mask used by interfaceName

offlineReason  A user-entered reason why the server is in ADMIN_DOWN or OFLINE status

physLocation  The name of the physical location where the server resides

profile  The name of the profile this server uses

profileDesc  A description of the profile this server uses

rack  A string indicating “server rack” location

routerHostName  The human-readable name of the router responsible for reaching this server

routerPortName  The human-readable name of the port used by the router responsible for reaching this server

1 For more information see the Wikipedia page on Lights-Out management.
status  The status of the server

See also:

Health Protocol

tcpPort  The port on which this server listens for incoming TCP connections

Note:  This is typically thought of as synonymous with “HTTP port”, as the port specified by httpsPort may also be used for incoming TCP connections.

type  The name of the ‘type’ of this server

xmppId  An identifier to be used in XMPP communications with the server - in nearly all cases this will be the same as hostName

xmppPasswd  The password used in XMPP communications with the server

# 402: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
   Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Mon, 10 Dec 2018 17:11:53 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 10 Dec 2018 21:11:53 GMT;
   path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: ZDeQrG0D7Q3Wy3ZEUT9t21QQ9F9Yc3RR/
   Qr91n22UniYubdhdkNhi3B+LYP5ZkVg8ByrVPPFyx6Na0iiBTGTQ==
Content-Length: 800

{  "response": {  "profile": "ATS_EDGE_TIER_CACHE",  "xmppPasswd": "",
"physLocation": "Apachecon North America 2018",  "cachegroup": "CDN_in_a_Box_Edge",
"interfaceName": "eth0",  "id": 9,
  "tcpPort": 80,
  "httpsPort": 443,
  "ipGateway": "172.16.239.1",
  "ip6Address": "fc01:9400:1000:8::100",
  "xmppId": "edge",
  "mgmtIpNetmask": "";
}(continues on next page)
servers/{{hostname}}/update_status

New in version 1.3.

Note: This endpoint only truly has meaning for cache servers, though it will return a valid response for any server configured in Traffic Ops.

GET

Retrieves information regarding pending updates and revalidation jobs for a given server

Auth. Required  Yes

Roles Required  None

Response Type  undefined - this endpoint will return a top-level array containing the response, as opposed to within a response object
Request Structure

Table 203: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>The (short) hostname of the server being inspected</td>
</tr>
</tbody>
</table>

# 403: Request Example

```text
GET /api/1.4/servers/edge/update_status HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

Each object in the returned array\(^1\) will contain the following fields:

- **host_id** The integral, unique identifier for the server for which the other fields in this object represent the pending updates and revalidation status
- **host_name** The (short) hostname of the server for which the other fields in this object represent the pending updates and revalidation status
- **parent_pending** A boolean telling whether or not the parents of this server have pending updates
- **parent_reval_pending** A boolean telling whether or not the parents of this server have pending revalidation jobs
- **reval_pending** `true` if the server has pending revalidation jobs, `false` otherwise
- **status** The name of the status of this server

See also:

*Health Protocol* gives more information on how these statuses are used, and the GET method of the *statuses* endpoint can be used to retrieve information about all server statuses configured in Traffic Ops.

- **upd_pending** `true` if the server has pending updates, `false` otherwise
- **use_reval_pending** A boolean which tells ORT whether or not this version of Traffic Ops should use pending revalidation jobs

Note: This field was introduced to give ORT the ability to work with Traffic Control versions 1.x and 2.x seamlessly - as of Traffic Control v3.0 there is

\(^1\) Despite that the returned object is an array, exactly one server's information is requested and thus returned. That is to say, the array should always have a length of exactly one.
no reason for this field to ever be false.

# 404: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Content-Length: 174

[{
    "host_name": "edge",
    "upd_pending": false,
    "reval_pending": false,
    "use_reval_pending": true,
    "host_id": 10,
    "status": "REPORTED",
    "parent_pending": false,
    "parent_reval_pending": false
}]
```

servers/{{ID}}

GET

Deprecated since version 1.1: Use the id query parameter of a GET request to servers instead.

Retrieves properties of a specific server.

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Array
Request Structure

Table 204: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a server</td>
</tr>
</tbody>
</table>

# 405: Request Example

```
GET /api/1.4/servers?id=10 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **cachegroup**  The name of the Cache Group to which this server belongs
- **cachegroupId**  The integral, unique identifier of the Cache Group to which this server belongs
- **cdnId**  The integral, unique identifier of the CDN to which the server belongs
- **cdnName**  Name of the CDN to which the server belongs
- **domainName**  The domain part of the server’s Fully Qualified Domain Name (FQDN)
- **guid**  An identifier used to uniquely identify the server
  - Deprecated since version 1.1: This is a legacy key which only still exists for compatibility reasons – it should always be `null`
- **hostName**  The (short) hostname of the server
- **httpsPort**  The port on which the server listens for incoming HTTPS connections/requests
- **id**  An integral, unique identifier for this server
- **iloIpAddress**  The IPv4 address of the server’s Integrated Lights-Out (ILO) service
- **iloIpGateway**  The IPv4 gateway address of the server’s ILO service
- **iloIpNetmask**  The IPv4 subnet mask of the server’s ILO service
- **iloPassword**  The password of the of the server’s ILO service user – displays as simply `********` if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)

---

1 For more information see the Wikipedia page on Lights-Out management.
iloUsername The user name for the server’s ILO service

interfaceMtu The Maximum Transmission Unit (MTU) to configured on interfaceName

interfaceName The name of the primary network interface used by the server

ip6Address The IPv6 address and subnet mask of interfaceName

ip6Gateway The IPv6 address of the gateway used by interfaceName

ipAddress The IPv4 address of interfaceName

ipGateway The IPv4 address of the gateway used by interfaceName

ipNetmask The IPv4 subnet mask used by interfaceName

lastUpdated The date and time at which this server description was last modified

mgmtIpAddress The IPv4 address of some network interface on the server used for ‘management’

mgmtIpGateway The IPv4 address of a gateway used by some network interface on the server used for ‘management’

mgmtIpNetmask The IPv4 subnet mask used by some network interface on the server used for ‘management’

offlineReason A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status

physLocation The name of the physical location where the server resides

physLocationId An integral, unique identifier for the physical location where the server resides

profile The name of the profile this server uses

profileDesc A description of the profile this server uses

profileId An integral, unique identifier for the profile used by this server

revalPending A boolean value which, if true indicates that this server has pending content invalidation/revalidation

rack A string indicating “server rack” location

routerHostName The human-readable name of the router responsible for reaching this server

routerPortName The human-readable name of the port used by the router responsible for reaching this server

status The status of the server

See also:

Health Protocol
statusId  The integral, unique identifier of the status of this server

See also:

*Health Protocol*

tcpPort  The port on which this server listens for incoming TCP connections

Note: This is typically thought of as synonymous with "HTTP port", as the port specified by httpsPort may also be used for incoming TCP connections.

type  The name of the ‘type’ of this server
typeId  The integral, unique identifier of the ‘type’ of this server

updPending  A boolean value which, if true, indicates that the server has updates of some kind pending, typically to be acted upon by Traffic Ops ORT

xmppId  An identifier to be used in XMPP communications with the server - in nearly all cases this will be the same as hostName

xmppPasswd  The password used in XMPP communications with the server

# 406: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: WyapQctUIhjzEALka5qbBiZR58BMc6MJSwjBeGyJS2UzbL3W6lN/4kvAZtPrP4qMWQBWz6JjbF7Y51NRAASUmQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 16:40:01 GMT
Content-Length: 939

{
  "response": [
    {
      "cachegroup": "CDN_in_a_Box_Mid",
      "cachegroupId": 6,
      "cdnId": 2,
      "cdnName": "CDN-in-a-Box",
      "domainName": "infra.ciab.test",
      "guid": null,
      "hostName": "mid",
      "httpsPort": 443,
      "id": 10,
```

(continues on next page)
"iloIpAddress": "",
"iloIpGateway": "",
"iloIpNetmask": "",
"iloPassword": "",
"iloUsername": "",
"interfaceMtu": 1500,
"interfaceName": "eth0",
"ip6Address": "fc01:9400:1000:8::120",
"ip6Gateway": "fc01:9400:1000:8::1",
"ipAddress": "172.16.239.120",
"ipGateway": "172.16.239.1",
"ipNetmask": "255.255.255.0",
"lastUpdated": "2018-12-05 18:45:05+00",
"mgmtIpAddress": "",
"mgmtIpGateway": "",
"mgmtIpNetmask": "",
"offlineReason": "",
"physLocation": "Apachecon North America 2018",
"physLocationId": 1,
"profile": "ATS_MID_TIER_CACHE",
"profileDesc": "Mid Cache - Apache Traffic Server",
"profileId": 10,
"rack": "",
"revalPending": false,
"routerHostName": "",
"routerPortName": "",
"status": "REPORTED",
"statusId": 3,
"tcpPort": 80,
"type": "MID",
"typeId": 12,
"updPending": false,
"xmppId": "mid",
"xmppPasswd": ""}

**PUT**

Allow user to edit a server.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object
Request Structure

Table 205: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a server</td>
</tr>
<tr>
<td>cachegroupId</td>
<td>The integral, unique identifier of the Cache Group to which this server shall belong</td>
</tr>
<tr>
<td>cdnId</td>
<td>The integral, unique identifier of the CDN to which the server shall belong</td>
</tr>
<tr>
<td>domainName</td>
<td>The domain part of the server’s Fully Qualified Domain Name (FQDN)</td>
</tr>
<tr>
<td>hostName</td>
<td>The (short) hostname of the server</td>
</tr>
<tr>
<td>httpsPort</td>
<td>An optional port number on which the server listens for incoming HTTPS connections/requests</td>
</tr>
<tr>
<td>iloIpAddress</td>
<td>An optional IPv4 address of the server’s Integrated Lights-Out (ILO) service¹</td>
</tr>
<tr>
<td>iloIpGateway</td>
<td>An optional IPv4 gateway address of the server’s ILO service¹</td>
</tr>
<tr>
<td>iloIpNetmask</td>
<td>An optional IPv4 subnet mask of the server’s ILO service¹</td>
</tr>
<tr>
<td>iloPassword</td>
<td>An optional string containing the password of the server’s ILO service user¹ - displays as simply ****** if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)</td>
</tr>
<tr>
<td>iloUsername</td>
<td>An optional string containing the user name for the server’s ILO service¹</td>
</tr>
<tr>
<td>interfaceMtu</td>
<td>The Maximum Transmission Unit (MTU) to configured on interfaceName</td>
</tr>
</tbody>
</table>

Note: In virtually all cases this ought to be 1500. Further note that the only acceptable values are 1500 and 9000.

interfaceName  The name of the primary network interface used by the server
ip6Address     An optional IPv6 address and subnet mask of interfaceName
ip6Gateway     An optional IPv6 address of the gateway used by interfaceName
ipAddress      The IPv4 address of interfaceName
ipGateway      The IPv4 address of the gateway used by interfaceName
ipNetmask      The IPv4 subnet mask used by interfaceName
mgmtIpAddress  An optional IPv4 address of some network interface on the server used for ‘management’
**mgmtIpGateway** An optional IPv4 address of a gateway used by some network interface on the server used for ‘management’

**mgmtIpNetmask** An optional IPv4 subnet mask used by some network interface on the server used for ‘management’

**physLocationId** An integral, unique identifier for the physical location where the server resides

**profileId** An integral, unique identifier for the profile used by this server

**revalPending** A boolean value which, if `true` indicates that this server has pending content invalidation/revalidation

**rack** An optional string indicating “server rack” location

**routerHostName** An optional string containing the human-readable name of the router responsible for reaching this server

**routerPortName** An optional string containing the human-readable name of the port used by the router responsible for reaching this server

**statusId** The integral, unique identifier of the status of this server

See also:

*Health Protocol*

**tcpPort** An optional port number on which this server listens for incoming TCP connections

---

**Note:** This is typically thought of as synonymous with “HTTP port”, as the port specified by `httpsPort` may also be used for incoming TCP connections.

**typeId** The integral, unique identifier of the ‘type’ of this server

**updPending** A boolean value which, if `true`, indicates that the server has updates of some kind pending, typically to be acted upon by Traffic Ops ORT

**xmppId** An optional identifier to be used in XMPP communications with the server - in nearly all cases this should be the same as `hostName`

**xmppPasswd** An optional password used in XMPP communications with the server

---

# 407: Request Example

```
PUT /api/1.4/servers/13 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 599
```

(continues on next page)
Content-Type: application/json

```
{
    "cachegroupId": 6,
    "cdnId": 2,
    "domainName": "infra.ciab.test",
    "hostName": "quest",
    "httpsPort": 443,
    "iloIpAddress": "",
    "iloIpGateway": "",
    "iloIpNetmask": "",
    "iloPassword": "",
    "iloUsername": "",
    "interfaceMtu": 1500,
    "interfaceName": "eth0",
    "ip6Address": "::1",
    "ip6Gateway": "::2",
    "ipAddress": "0.0.0.1",
    "ipGateway": "0.0.0.2",
    "ipNetmask": "255.255.255.0",
    "mgmtIpAddress": "",
    "mgmtIpGateway": "",
    "mgmtIpNetmask": "",
    "offlineReason": "",
    "physLocationId": 1,
    "profileId": 10,
    "routerHostName": "",
    "routerPortName": "",
    "statusId": 3,
    "tcpPort": 80,
    "typeId": 12,
    "updPending": true
}
```

### Response Structure

- **cachegroup** The name of the Cache Group to which this server belongs

- **cachegroupId** The integral, unique identifier of the Cache Group to which this server belongs

- **cdnId** The integral, unique identifier of the CDN to which the server belongs

- **cdnName** Name of the CDN to which the server belongs

- **domainName** The domain part of the server’s Fully Qualified Domain Name (FQDN)

- **guid** An identifier used to uniquely identify the server
Deprecation: This is a legacy key which only still exists for compatibility reasons - it should always be null

**hostName**  
The (short) hostname of the server

**httpsPort**  
The port on which the server listens for incoming HTTPS connections/requests

**id**  
An integral, unique identifier for this server

**iloIpAddress**  
The IPv4 address of the server’s Integrated Lights-Out (ILO) service

**iloIpGateway**  
The IPv4 gateway address of the server’s ILO service

**iloPassword**  
The password of the of the server’s ILO service - displays as simply ****** if the currently logged-in user does not have the ‘admin’ or ‘operations’ role(s)

**iloUsername**  
The user name for the server’s ILO service

**interfaceMtu**  
The Maximum Transmission Unit (MTU) to configured on

**interfaceName**  
The name of the primary network interface used by the server

**ip6Address**  
The IPv6 address and subnet mask of

**ip6Gateway**  
The IPv6 address of the gateway used by

**ipAddress**  
The IPv4 address of

**ipGateway**  
The IPv4 address of the gateway used by

**ipNetmask**  
The IPv4 subnet mask used by

**lastUpdated**  
The date and time at which this server description was last modified

**mgmtIpAddress**  
The IPv4 address of some network interface on the server used for ‘management’

**mgmtIpGateway**  
The IPv4 address of a gateway used by some network interface on the server used for ‘management’

**mgmtIpNetmask**  
The IPv4 subnet mask used by some network interface on the server used for ‘management’

**offlineReason**  
A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status

**physLocation**  
The name of the physical location where the server resides

**physLocationId**  
An integral, unique identifier for the physical location where the server resides

**profile**  
The name of the profile this server uses

**profileDesc**  
A description of the profile this server uses
**profileId**  An integral, unique identifier for the profile used by this server

**revalPending**  A boolean value which, if true indicates that this server has pending content invalidation/revalidation

**rack**  A string indicating “server rack” location

**routerHostName**  The human-readable name of the router responsible for reaching this server

**routerPortName**  The human-readable name of the port used by the router responsible for reaching this server

**status**  The status of the server

    See also:

    Health Protocol

**statusId**  The integral, unique identifier of the status of this server

    See also:

    Health Protocol

**tcpPort**  The port on which this server listens for incoming TCP connections

    **Note:** This is typically thought of as synonymous with “HTTP port”, as the port specified by httpsPort may also be used for incoming TCP connections.

**type**  The name of the ‘type’ of this server

**typeId**  The integral, unique identifier of the ‘type’ of this server

**updPending**  A boolean value which, if true, indicates that the server has updates of some kind pending, typically to be acted upon by Traffic Ops ORT

**xmppId**  An identifier to be used in XMPP communications with the server - in nearly all cases this will be the same as hostName

**xmppPasswd**  The password used in XMPP communications with the server

---

### # 408: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 9lGAMCCC9I/bOpuBSyf3ACffjHeRuXCTuxrA/oU78uYzW5FePTq5PHSSnsnqKG5E0vWg0Rko0CwguGeNc9IT0w==
```

(continues on next page)
Apache Traffic Control Documentation, Release 3.0.0

(continued from previous page)

X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 17:58:57 GMT
Content-Length: 848
{ "alerts": [
{
"text": "server was updated.",
"level": "success"
}
],
"response": {
"cachegroup": null,
"cachegroupId": 6,
"cdnId": 2,
"cdnName": null,
"domainName": "infra.ciab.test",
"guid": null,
"hostName": "quest",
"httpsPort": 443,
"id": 13,
"iloIpAddress": "",
"iloIpGateway": "",
"iloIpNetmask": "",
"iloPassword": "",
"iloUsername": "",
"interfaceMtu": 1500,
"interfaceName": "eth0",
"ip6Address": "::1",
"ip6Gateway": "::2",
"ipAddress": "0.0.0.1",
"ipGateway": "0.0.0.2",
"ipNetmask": "255.255.255.0",
"lastUpdated": "2018-12-10 17:58:57+00",
"mgmtIpAddress": "",
"mgmtIpGateway": "",
"mgmtIpNetmask": "",
"offlineReason": "",
"physLocation": null,
"physLocationId": 1,
"profile": null,
"profileDesc": null,
"profileId": 10,
"rack": null,
"revalPending": null,
"routerHostName": "",
"routerPortName": "",
"status": null,
"statusId": 3,
"tcpPort": 80,
(continues on next page)

642

Chapter 5. APIs


"type": "",
"typeId": 12,
"updPending": true,
"xmppId": null,
"xmppPasswd": null
}
}

DELETE

Allow user to delete server through api.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  undefined

Request Structure

Table 206: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a server</td>
</tr>
</tbody>
</table>

# 409: Request Example

```
DELETE /api/1.4/servers/13 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 410: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: JZdjKJYWN9w9NF6VE/rVkGUqEcyeKB2ADkkI4LNDmpgLwu53bR/IAA+4uwrow0zuba/4MSEhHKshutziypSxPg==
```
servers/{{ID}}/deliveryservices

GET

Retrieves all Delivery Services assigned to a specific server.

Auth. Required   Yes
Roles Required   None
Response Type   Array

Response Structure

 active  true if the Delivery Service is active, false otherwise
anonymousBlockingEnabled  true if Anonymous Blocking has been configured for the Delivery Service, false otherwise
cacheurl  A setting for a deprecated feature of now-unsupported ATS versions

Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later

ccrDnsTtl  The TTL of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons

cdnId  The integral, unique identifier of the CDN to which the Delivery Service belongs

cdnName  Name of the CDN to which the Delivery Service belongs

checkPath  The path portion of the URL to check connections to this Delivery Service’s origin server

consistentHashRegMatch If defined, this is a regex used for the Pattern-Based Consistent Hashing feature. It is only applicable for HTTP and Steering Delivery Services

New in version 1.5.

displayName  The display name of the Delivery Service

dnsBypassCname  Domain name to overflow requests for HTTP Delivery Services - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp  The IPv4 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassIp6  The IPv6 IP to use for bypass on a DNS Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

dnsBypassTtl  The time for which a DNS bypass of this Delivery Service shall remain active

dscp  The DSCP with which to mark traffic as it leaves the CDN and reaches clients

edgeHeaderRewrite  Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite ATS plugin

fqPacingRate  The Fair-Queuing Pacing Rate in Bytes per second set on the all TCP connection sockets in the Delivery Service (see tc-fq_codel(8) for more information) - Linux only

geoLimit  The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

3 This only applies to DNS-routed Delivery Services

5.1. Traffic Ops API
0 None - no limitations
1 Only route when the client’s IP is found in the Coverage Zone File
2 Only route when the client’s IP is found in the Coverage Zone File, or when the client can be determined to be from the United States of America

**Warning:** This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

geoLimitCountries A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this Delivery Service

geoLimitRedirectUrl A URL to which clients blocked by Regional Geographic Blocking or the geoLimit settings will be re-directed

geoProvider An integer that represents the provider of a database for mapping IPs to geographic locations; currently only the following values are supported:
0 The “Maxmind” GeoIP2 database (default)
1 Neustar GeoPoint IP address database

**Warning:** It’s not clear whether Neustar databases are actually supported; this is an old option and compatibility may have been broken over time.

globalMaxMbps The maximum global bandwidth allowed on this Delivery Service. If exceeded, traffic will be routed to dnsBypassIp (or dnsBypassIp6 for IPv6 traffic) for DNS Delivery Services and to httpBypassFqdn for HTTP Delivery Services

globalMaxTps The maximum global transactions per second allowed on this Delivery Service. When this is exceeded traffic will be sent to the dnsBypassIp (and/or dnsBypassIp6) for DNS Delivery Services and to the httpBypassFqdn for HTTP Delivery Services

httpBypassFqdn The HTTP destination to use for bypass on an HTTP Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

id An integral, unique identifier for this Delivery Service

infoUrl This is a string which is expected to contain at least one URL pointing to more information about the Delivery Service. Historically, this has been used to link relevant JIRA tickets
initialDispersion The number of cache servers between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1\(^1\).

ipv6RoutingEnabled If true, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache server; if false all addresses will be IPv4, regardless of the client connection\(^1\).

lastUpdated The date and time at which this Delivery Service was last updated, in a ctime(3) -like format.

logsEnabled If true, logging is enabled for this Delivery Service, otherwise it is disabled.

longDesc A description of the Delivery Service.

longDesc1 A field used when more detailed information that that provided by longDesc is desired.

longDesc2 A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired.

matchList An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service.

pattern A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped).

setNumber An integral, unique identifier for the set of types to which the type field belongs.

type The Type of match performed using pattern to determine whether or not to use this Delivery Service.

HOST_REGEXP Use the Delivery Service if pattern matches the Host: HTTP header of an HTTP request\(^1\).

HEADER_REGEXP Use the Delivery Service if pattern matches an HTTP header (both the name and value) in an HTTP request\(^1\).

PATH_REGEXP Use the Delivery Service if pattern matches the request path of this Delivery Service’s URL.

STEERING_REGEXP Use the Delivery Service if pattern matches the xml_id of one of this Delivery Service’s “Steering” target Delivery Services.

maxDnsAnswers The maximum number of IPs to put in responses to A/AAAA DNS record requests (0 means all available)\(^3\).

---

\(^1\) This only applies to HTTP-routed Delivery Services.

---

5.1. Traffic Ops API 647
midHeaderRewrite  Rewrite operations to be performed on TCP headers at the
Edge-tier cache level - used by the Header Rewrite ATS plugin

missLat  The latitude to use when the client cannot be found in the Coverage Zone
File or a geographic IP lookup

missLong  The longitude to use when the client cannot be found in the Coverage
Zone File or a geographic IP lookup

multiSiteOrigin true if the Multi Site Origin feature is enabled for this Delivery
Service, false otherwise²

orgServerFqdn  The URL of the Delivery Service’s origin server for use in re-
trieving content from the origin server

Note:  Despite the field name, this must truly be a full URL - including the
protocol (e.g. http:// or https://) - NOT merely the server’s FQDN

originShield  An “origin shield” is a forward proxy that sits between Mid-tier
cache servers and the origin and performs further caching beyond what’s
offered by a standard CDN. This field is a string of FQDNs to use as origin
shields, delimited by |

profileDescription  The description of the Traffic Router Profile with which this
Delivery Service is associated

profileId  The integral, unique identifier for the Traffic Router Profile with which
this Delivery Service is associated

profileName  The name of the Traffic Router Profile with which this Delivery Ser-
vice is associated

protocol  The protocol which clients will use to communicate with Edge-tier cache
servers¹ - this is an integer on the interval [0-2] where the values have these
meanings:

0  HTTP

1  HTTPS

2  Both HTTP and HTTPS

queryStringIgnore  Tells cache servers whether or not to consider URLs with different
query parameter strings to be distinct - this is an integer on the interval [0-2]
where the values have these meanings:

0  URLs with different query parameter strings will be considered distinct for
    caching purposes, and query strings will be passed upstream to the
    origin

1  URLs with different query parameter strings will be considered identical for
    caching purposes, and query strings will be passed upstream to the
    origin

² See Use Multi-Site Origin Feature
Query strings are stripped out by Edge-tier cache servers, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin.

**rangeRequestHandling** Tells caches how to handle range requests\(^4\) - this is an integer on the interval \([0,2]\) where the values have these meanings:

- **0** Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache server.
- **1** Use the background_fetch plugin to service the range request while caching the whole object.
- **2** Use the experimental cache_range_requests plugin to treat unique ranges as unique objects.

**regexRemap** A regular expression “remap rule” to apply to this Delivery Service at the Edge tier.

See also:

The Apache Trafficserver documentation for the Regex Remap plugin.

**regionalGeoBlocking** true if Regional Geo Blocking is in use within this Delivery Service, false otherwise.

See also:

See Configure Regional Geo-blocking (RGB) for more information.

**remapText** Additional, raw text to add to the line for this Delivery Service for cache servers.

See also:

The Apache Trafficserver documentation for the Regex Remap plugin.

**signed** true if token-based authentication is enabled for this Delivery Service, false otherwise.

**signingAlgorithm** Type of URL signing method to sign the URLs, basically comes down to one of two plugins or null:

- **null** Token-based authentication is not enabled for this Delivery Service.
- **url_sig**: URL Signing token-based authentication is enabled for this Delivery Service.
- **uri_signing**: URI Signing token-based authentication is enabled for this Delivery Service.

See also:

The Apache Trafficserver documentation for the url_sig plugin and the draft RFC for uri_signing - note, however that the current implementation of uri_signing uses Draft 12 of that RFC document, NOT the latest.

\(^4\) These fields are required for HTTP-routed and DNS-routed Delivery Services, but are optional for (and in fact may have no effect on) STEERING and ANY_MAP Delivery Services.
sslKeyVersion  This integer indicates the generation of keys in use by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated

Warning:  This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

tenantId  The integral, unique identifier of the Tenant who owns this Delivery Service

trRequestHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)

trResponseHeaders  If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it’s a template where __RETURN__ translates to a carriage return and line feed (\r\n)

type  The name of the routing type of this Delivery Service e.g. “HTTP”
typeId  The integral, unique identifier of the routing type of this Delivery Service

xmlId  A unique string that describes this Delivery Service - exists for legacy reasons, but is used heavily by Traffic Control components

# 412: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: heK6DafnKW6KdyqQ7ITJqcSt1i3ixkWYjnbQ2EzR8ZU6T1ij3Takr6CNr0BcD5yWFVNd8mvMPcj5X
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 16:53:04 GMT
Content-Length: 1129

{  "response": [  ]
  "active": true,
  "cacheurl": null,
  "ccrDnsTtl": null,
  "cdnId": 2,
  "checkPath": null,
  "deepCachingType": null,
  "displayName": "Demo 1",

(continues on next page)
"dnsBypassCname": null,
"dnsBypassIp": null,
"dnsBypassIp6": null,
"dnsBypassTtl": null,
"dscp": 0,
"edgeHeaderRewrite": null,
"fqPacingRate": null,
"geoLimit": 0,
"geoLimitCountries": null,
"geoLimitRedirectURL": null,
"geoProvider": 0,
"globalMaxMbps": null,
"globalMaxTps": null,
"httpBypassFqdn": null,
"id": 1,
"infoUrl": null,
"initialDispersion": 1,
"ipv6RoutingEnabled": true,
"lastUpdated": "2018-12-05 17:51:00+00",
"logsEnabled": true,
"longDesc": "Apachecon North America 2018",
"longDesc1": null,
"longDesc2": null,
"maxDnsAnswers": null,
"midHeaderRewrite": null,
"missLat": 42,
"missLong": -88,
"multiSiteOrigin": false,
"multiSiteOriginAlgo": null,
"originShield": null,
"orgServerFqdn": "http://origin.infra.ciab.test",
"profileDescription": null,
"profileId": null,
"protocol": 0,
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regexRemap": null,
"regionalGeoBlocking": false,
"remapText": null,
"routingName": "video",
"signingAlgorithm": null,
"sslKeyVersion": null,
"trRequestHeaders": null,
"trResponseHeaders": null,
"tenantId": 1,
"typeId": 1,
"xmlId": "demo1"}
servers/{{ID}}/queue_update

Deprecated since version 1.1: Use the PUT method of the servers/{{ID}} endpoint instead.

POST

Queue or dequeue updates for a specific server.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

Table 208: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the server on which updates are being queued or dequeued</td>
</tr>
</tbody>
</table>

**action**  A string describing what action to take regarding server updates; one of:

queue  Enqueue updates for the server, propagating configuration changes to the actual server
dequeue  Cancels any pending updates on the server

# 413: Request Example

```
POST /api/1.1/servers/13/queue_update HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 22
Content-Type: application/json

{
    "action": "dequeue"
}
```

Response Structure

**action**  The action processed, one of:

queue  Enqueued updates for the server, propagating configuration changes to the actual server
**dequeue**  Canceled any pending updates on the server

**serverId**  The integral, unique identifier of the server on which action was taken

### # 414: Response Example

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Mon, 10 Dec 2018 18:20:04 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 10 Dec 2018 22:20:04 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 9Mmo9hIFZyF5gAvfdJD//VH9eNg1HVlinXt88H0g1JSHhwNd8gMxaFyC+f9XZfiNAoGd1MKi1934ZJGmaIR6qQ==
Content-Length: 49
{
    "response": {
        "serverId": "13",
        "action": "dequeue"
    }
}
```

**servers/{{ID}}/status**

Deprecated since version 1.1: Use the PUT method of servers/{{ID}} instead.

**PUT**

Updates server status and queues updates on all child caches if server type is EDGE or MID. Also, captures offline reason if status is set to ADMIN_DOWN or OFFLINE and prepends offline reason with the user that initiated the status change.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  undefined
Request Structure

Table 209: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the server whose status is being changed</td>
</tr>
</tbody>
</table>

**offlineReason**  A string containing the reason for the status change

**status**  The name or integral, unique identifier of the server’s new status

### #415: Request Example

```plaintext
PUT /api/1.4/servers/13/status HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 56
Content-Type: application/json

{
    "status": "ADMIN_DOWN",
    "offlineReason": "Bad drives"
}
```

Response Structure

### #416: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Mon, 10 Dec 2018 18:08:44 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 10 Dec 2018 22:08:44 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: LS1jCo5eMVXxmeYDo1O2LgLYazocSggR5hynNoLcPmMov9u2s3ulksPdQtG1N3aS+VM9tMgsCrahFP
Content-Length: 158

{
    "alerts": [ ]
}
```

(continues on next page)
"level": "success",
"text": "Updated status [ ADMIN_DOWN ] for quest infra.ciab.test [ admin: Bad drives ] and queued updates on all child caches"
]}

servers/\{{server}\}/configfiles/ats

See also:
The servers/\{{server}\}/configfiles/ats/\{{filename\}}, cdns/\{{cdn\}}/configfiles/ats/\{{filename\}}, and profiles/\{{profile\}}/configfiles/ats/\{{filename\}} endpoints.

GET

Gets a list of the configuration files used by server

Auth. Required Yes

Roles Required “operations”

Response Type NOT PRESENT - endpoint returns custom application/json response

Request Structure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>string or integer</td>
<td>Either the name or integral, unique, identifier of a server</td>
</tr>
</tbody>
</table>

Response Structure

info An object that provides information about server as it is understood by Traffic Ops

cdnId The integral, unique, identifier of the CDN to which server is assigned
cdnName The name of the CDN to which server is assigned
profileId The integral, unique, identifier of the profile used by server
profileName The name of the profile used by server
serverId An integral, unique, identifier for server

serverIpv4 IPv4 address of the server

serverTcpPort The port number on which server listens for incoming TCP connections

toRevProxyUrl An optional field which, if present, gives a URL that resolves to a proxy for Traffic Ops which server ought to use rather than directly contacting

toUrl A full URL that resolves to the Traffic Ops instance

configFiles An array of objects which each represent a configuration file used by the server

apiUri An optional field which, if present, gives a path relative to the Traffic Ops instance (or reverse proxy when applicable) URL where the actual file’s contents may be retrieved

fnameOnDisk The filename of the configuration file as stored on the server

location The directory location of the configuration file as stored on the server

scope The “scope” of the configuration file, which will be one of:

“cdns” The file is used by all caches in the CDN

“profiles” The file is used by all servers with the same profile

“servers” The most specific grouping of servers which use this file is simply a collection of distinct servers

url An optional field which, if present, gives the full URL used to retrieve the actual file’s contents

Changed in version Traffic: Control 2.0 Elements of the "configFile" array may no longer have the "contents" key - all file contents are now retrieved via a network request

# 417: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

(continues on next page)

1 Exactly one of these fields is guaranteed to exist for any given configuration file - although “apiUrl” is far more common.
{ "info": {   "profileId": 9,   "toUrl": null,   "serverIpv4": "172.16.239.100",   "serverTcpPort": 80,   "serverName": "edge",   "cdnId": 2,   "cdnName": "CDN-in-a-Box",   "serverId": 10,   "profileName": "ATS_EDGE_TIER_CACHE" },  "configFiles": [   {     "fnameOnDisk": "astats.config",     "location": "/etc/trafficserver",     "apiUri": "/api/1.2/profiles/ATS_EDGE_TIER_CACHE/configfiles/ats/astats.config",     "scope": "profiles"   },   {     "fnameOnDisk": "cache.config",     "location": "/etc/trafficserver",     "apiUri": "/api/1.2/profiles/ATS_EDGE_TIER_CACHE/configfiles/ats/cache.config",     "scope": "profiles"   },   {     "fnameOnDisk": "cacheurl_foo.config",     "location": "/etc/trafficserver",     "apiUri": "/api/1.2/cdns/CDN-in-a-Box/configfiles/ats/cacheurl_foo.config",     "scope": "cdns"   },   {     "fnameOnDisk": "hdr_rw_foo.config",     "location": "/etc/trafficserver",     "apiUri": "/api/1.2/cdns/CDN-in-a-Box/configfiles/ats/hdr_rw_foo.config",     "scope": "cdns"   }  ]}
"scope": "cdns"
},
{
  "fnameOnDisk": "hosting.config",
  "location": "/etc/trafficserver/",
  "apiUri": "/api/1.2/servers/edge/configfiles/ats/hosting.config",
  "scope": "servers"
},
{
  "fnameOnDisk": "ip_allow.config",
  "location": "/etc/trafficserver",
  "apiUri": "/api/1.2/servers/edge/configfiles/ats/ip_allow.config",
  "scope": "servers"
},
{
  "fnameOnDisk": "parent.config",
  "location": "/etc/trafficserver/",
  "apiUri": "/api/1.2/servers/edge/configfiles/ats/parent.config",
  "scope": "servers"
},
{
  "fnameOnDisk": "plugin.config",
  "location": "/etc/trafficserver/",
  "apiUri": "/api/1.2/profiles/ATS_EDGE_TIER_CACHE/configfiles/ats/plugin.config",
  "scope": "profiles"
},
{
  "fnameOnDisk": "records.config",
  "location": "/etc/trafficserver/",
  "apiUri": "/api/1.2/profiles/ATS_EDGE_TIER_CACHE/configfiles/ats/records.config",
  "scope": "profiles"
},
{
  "fnameOnDisk": "regex_remap_foo.config",
  "location": "/etc/trafficserver/",
  "apiUri": "/api/1.2/cdns/CDN-in-a-Box/configfiles/ats/regex_remap_foo.config",
  "scope": "cdns"
},
{
  "fnameOnDisk": "regex_revalidate.config",
  "location": "/etc/trafficserver/",
  "apiUri": "/api/1.2/cdns/CDN-in-a-Box/configfiles/ats/regex_revalidate.config",
  "scope": "cdns"
}
Note: Some DSCP-related files like e.g. set_dscp_0.config have been removed from this response, which otherwise reflects a stock CDN-in-a-Box configuration. This was done both for brevity’s sake, and due to the expectation that these will disappear from the default configuration in the (hopefully near) future.

servers/{{server}}/configfiles/ats/{{filename}}

See also:
The servers/{{server}}/configfiles/ats endpoint

GET

Returns the requested configuration file for download.

Auth. Required  Yes

Roles Required  “operations”

Response Type  NOT PRESENT - endpoint returns custom text/plain response (represents the contents of the requested configuration file)
Request Structure

Table 211: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>string or integer</td>
<td>Either the name or integral, unique, identifier of a server</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The name of a configuration file used by server</td>
</tr>
</tbody>
</table>

# 418: Request Example

```
GET /api/1.2/servers/edge/configfiles/ats/hosting.config HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=... 
```

Response Structure

**Note:** If the file identified by `filename` does exist, but is configured at a higher level than “server”, a JSON response will be returned and the `alerts` array will contain a "level": "error" node which identifies the correct scope of the configuration file.

# 419: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Date: Thu, 15 Nov 2018 15:32:25 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 15 Nov 2018 19:32:25 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: EmhHogPfcxQq2zHmFFJtjwzZiUHNqOZvE572Se/H/54gwarkKjm89+xJr7fQbfytc7xWYAzwfjNl6LfbM0hg==
Content-Length: 107
```

**# DO NOT EDIT - Generated for edge by Traffic Ops (trafficops.infra.ciab.test:443) on Thu Nov 15 15:32:25 UTC 2018**

hostname=* volume=1
servers/status

GET

Retrieves an aggregated view of all server statuses across all CDNs

- **Auth. Required**: Yes
- **Roles Required**: None
- **Response Type**: Object

**Request Structure**

No parameters available.

**Response Structure**

- **status**: Every key in the `response` object will be the name of a valid server status, with a value that is the number of servers with that status. If there are no servers with a given status, that status will not appear as a key.

  - **See also**:

    - `statuses` can be queried to retrieve all possible server statuses, as well as to create new statuses or modify existing statuses.

  ```
  HTTP/1.1 200 OK
  Access-Control-Allow-Credentials: true
  Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
  Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
  Access-Control-Allow-Origin: *
  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
  Content-Type: application/json
  Date: Mon, 04 Feb 2019 16:22:14 GMT
  Server: Mojolicious (Perl)
  Set-Cookie: mojolicious=...; expires=Mon, 04 Feb 2019 20:22:14 GMT; path=/; HttpOnly
  Vary: Accept-Encoding
  Whole-Content-Sha512: M072YRXvtNwjnCfntv/W3AsSpOhC17Cpm0UDznOcXxwwgRYSGXz2MoeeovXSNyim62FJJoQJo1ccRSAW9ZMcA==
  Content-Length: 38

  { "response": { "REPORTED": 2, "ONLINE": 9 } }
  ```

5.1. Traffic Ops API
servers/totals

GET

Retrieves a count of each type of server across all CDNs.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available.

Response Structure

count  The number of servers of this type configured in this instance of Traffic Ops

type  The name of the type servers herein counted

# 421: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Mon, 10 Dec 2018 17:02:02 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 10 Dec 2018 21:02:02 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: J4wy8zf+LX44/qWIbvziWHCcDZpUJ9GopOVuVqPbVHUCh1Vl9o8FnE7T+V0639n9Xyw9k10NcaGIqASA+O9Rzg==
Content-Length: 305

{
    "response": [
        {
            "count": 1,
            "type": "EDGE"
        },
        {
            "count": 1,
            "type": "MID"
        }
    ]
```

(continues on next page)
snapshot/{{name}}

**PUT**

Performs a CDN snapshot. Effectively, this propagates the new *configuration* of the CDN to its *operating state*, which replaces the output of the `cdns/{{name}}/snapshot` endpoint with the output of the `cdns/{{name}}/snapshot/new` endpoint.

**Note:** Snapshotting the CDN also deletes all HTTPS certificates for every *Delivery Service* which has been deleted since the last *Snapshot*.  

---

5.1. Traffic Ops API
Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  undefined

Request Structure

Table 212: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the CDN for which a snapshot shall be taken</td>
</tr>
</tbody>
</table>

# 422: Request Example

```
PUT /api/1.4/snapshot/CDN-in-a-Box HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 423: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-
Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXcg/
SpIdNs6c5H0NE8XYXysP+DGNKHfuwvY7kxvUdBeoG1ODJ6+SfaPg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 12 Dec 2018 22:00:18 GMT
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

staticdnsentries

GET

Retrieve all static DNS entries configured within Traffic Control

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 213: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>no</td>
<td>Return only static DNS entries that operate on this address/CNAME</td>
</tr>
<tr>
<td>cachegroup</td>
<td>no</td>
<td>Return only static DNS entries assigned to this Cache Group</td>
</tr>
<tr>
<td>cachegroupId</td>
<td>no</td>
<td>Return only static DNS entries assigned to the Cache Group identified by this integral, unique identifier</td>
</tr>
<tr>
<td>delivery-service</td>
<td>no</td>
<td>Return only static DNS entries that apply within the domain of the Delivery Service with this 'xml_id'</td>
</tr>
<tr>
<td>delivery-serviceId</td>
<td>no</td>
<td>Return only static DNS entries that apply within the domain of the Delivery Service identified by this integral, unique identifier</td>
</tr>
<tr>
<td>host</td>
<td>no</td>
<td>Return only static DNS entries that resolve this FQDN</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the static DNS entry with this integral, unique identifier</td>
</tr>
<tr>
<td>ttl</td>
<td>no</td>
<td>Return only static DNS entries with this TTL</td>
</tr>
<tr>
<td>type</td>
<td>no</td>
<td>Return only static DNS entries of this type</td>
</tr>
<tr>
<td>typeId</td>
<td>no</td>
<td>Return only static DNS entries of the type identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>

# 424: Request Example

```plaintext
GET /api/1.4/staticdnselements?address=foo.bar HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **address**: If `typeId` identifies a CNAME type record, this is the Canonical Name (CNAME) of the server, otherwise it is the IP address to which `host` shall be resolved.

- **cachegroup**: An optional string containing the name of a Cache Group which will service this static DNS entry

**Note**: This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.
**cachegroupId** An optional, integral, unique identifier of a Cache Group which will service this static DNS entry

**Note:** This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

**deliveryservice** The name of a *Delivery Service* under the domain of which this static DNS entry shall be active

**deliveryserviceId** The integral, unique identifier of a *Delivery Service* under the domain of which this static DNS entry shall be active

**host** If `typeId` identifies a CNAME type record, this is an alias for the CNAME of the server, otherwise it is the Fully Qualified Domain Name (FQDN) which shall resolve to `address`

**id** An integral, unique identifier for this static DNS entry

**ttl** The Time To Live (TTL) of this static DNS entry in seconds

**type** The name of the type of this static DNS entry

**typeId** The integral, unique identifier of the type of this static DNS entry

---

**HTTP/1.1 200 OK**

Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Px1zTH3ihg+hfmADGcap0Juuud39fGsx5Y3CzqaFNmRwFu1ZLMzOsyo0EN2pb7vp0tp16/zeIUYAC3dbsBwOmA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:04:33 GMT
Content-Length: 226

```json
{
  "response": [
    {
      "address": "foo.bar",
      "cachegroup": null,
      "cachegroupId": null,
      "deliveryservice": "demol",
      "deliveryserviceId": 1,
      "host": "test",
      "id": 2,
      "lastUpdated": "2018-12-10 19:59:56+00",
    }
  ]
}
```

(continues on next page)
POST

New in version 1.3.

Creates a new, static DNS entry.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

address  If typeId identifies a CNAME type record, this is the Canonical Name (CNAME) of the server, otherwise it is the IP address to which host shall be resolved

 cachegroupId An optional, integral, unique identifier of a Cache Group which will service this static DNS entry

Note: This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

deliveryserviceId The integral, unique identifier of a Delivery Service under the domain of which this static DNS entry shall be active

host  If typeId identifies a CNAME type record, this is an alias for the CNAME of the server, otherwise it is the Fully Qualified Domain Name (FQDN) which shall resolve to address

ttl The Time To Live (TTL) of this static DNS entry in seconds

typeId The integral, unique identifier of the type of this static DNS entry

# 426: Request Example

```
POST /api/1.4/staticdnsentries HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```
Content-Length: 92
Content-Type: application/json

```json
{
    "address": "test.quest",
    "deliveryserviceId": 1,
    "host": "test",
    "ttl": 300,
    "typeId": 40
}
```

### Response Structure

- **address** If `typeId` identifies a CNAME type record, this is the Canonical Name (CNAME) of the server, otherwise it is the IP address to which `host` shall be resolved.

- **cachegroup** An optional string containing the name of a Cache Group which will service this static DNS entry.

  **Note:** This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

- **cachegroupId** An optional, integral, unique identifier of a Cache Group which will service this static DNS entry.

  **Note:** This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

- **deliveryservice** The name of a Delivery Service under the domain of which this static DNS entry shall be active.

- **deliveryserviceId** The integral, unique identifier of a Delivery Service under the domain of which this static DNS entry shall be active.

- **host** If `typeId` identifies a CNAME type record, this is an alias for the CNAME of the server, otherwise it is the Fully Qualified Domain Name (FQDN) which shall resolve to `address`.

- **id** An integral, unique identifier for this static DNS entry.

- **ttl** The Time To Live (TTL) of this static DNS entry in seconds.

- **type** The name of the type of this static DNS entry.

- **typeId** The integral, unique identifier of the type of this static DNS entry.
# 427: Response Example

```http
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 8dcJyjw2NJ2x0L9Oz16P7g/
7j5A1ljpylY6Y+rRVQ2wGcwYI3yiGPrz6ur0qKzqgEBBsh8aPF44WTHAR9jUJdg==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 19:54:19 GMT
Content-Length: 282

{ "alerts": [  
    {  
        "text": "staticDNSEntry was created.",  
        "level": "success"  
    }  
],  
"response": {  
    "address": "test.quest",  
    "cachegroup": null,  
    "cachegroupId": null,  
    "deliveryservice": null,  
    "deliveryserviceId": 1,  
    "host": "test",  
    "id": 2,  
    "lastUpdated": "2018-12-10 19:54:19+00",  
    "ttl": 300,  
    "type": null,  
    "typeId": 40  
}  
}  
```

**PUT**

New in version 1.3.

Updates a static DNS entry.

Authentication Required: Yes

Role(s) Required: admin or oper
Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The integral, unique identifier of the static DNS entry to modify</td>
</tr>
<tr>
<td>address</td>
<td>If typeId identifies a CNAME type record, this is the Canonical Name</td>
</tr>
<tr>
<td></td>
<td>(CNAME) of the server, otherwise it is the IP address to which host shall</td>
</tr>
<tr>
<td></td>
<td>be resolved</td>
</tr>
<tr>
<td>cachegroupId</td>
<td>An optional, integral, unique identifier of a Cache Group which</td>
</tr>
<tr>
<td></td>
<td>will service this static DNS entry</td>
</tr>
<tr>
<td>deliveryserviceId</td>
<td>The integral, unique identifier of a Delivery Service under the</td>
</tr>
<tr>
<td></td>
<td>domain of which this static DNS entry shall be active</td>
</tr>
<tr>
<td>host</td>
<td>If typeId identifies a CNAME type record, this is an alias for the CNAME of</td>
</tr>
<tr>
<td></td>
<td>the server, otherwise it is the Fully Qualified Domain Name (FQDN) which</td>
</tr>
<tr>
<td></td>
<td>shall resolve to address</td>
</tr>
<tr>
<td>ttl</td>
<td>The Time To Live (TTL) of this static DNS entry in seconds</td>
</tr>
<tr>
<td>typeId</td>
<td>The integral, unique identifier of the type of this static DNS entry</td>
</tr>
</tbody>
</table>

Note: This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

deliveryserviceId The integral, unique identifier of a Delivery Service under the domain of which this static DNS entry shall be active

host If typeId identifies a CNAME type record, this is an alias for the CNAME of the server, otherwise it is the Fully Qualified Domain Name (FQDN) which shall resolve to address

ttl The Time To Live (TTL) of this static DNS entry in seconds

typeId The integral, unique identifier of the type of this static DNS entry

# 428: Request Example

```
PUT /api/1.4/staticdnse
tid=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 89
Content-Type: application/json

{
    "address": "foo.bar",
    "deliveryserviceId": 1,
    "host": "test",
    "ttl": 300,
    "typeId": 40
}
```
Response Structure

**address**  If `typeId` identifies a CNAME type record, this is the Canonical Name (CNAME) of the server, otherwise it is the IP address to which `host` shall be resolved

**cachegroup**  An optional string containing the name of a Cache Group which will service this static DNS entry

*Note:* This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

**cachegroupId**  An optional, integral, unique identifier of a Cache Group which will service this static DNS entry

*Note:* This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.

**deliveryservice**  The name of a *Delivery Service* under the domain of which this static DNS entry shall be active

**deliveryserviceId**  The integral, unique identifier of a *Delivery Service* under the domain of which this static DNS entry shall be active

**host**  If `typeId` identifies a CNAME type record, this is an alias for the CNAME of the server, otherwise it is the Fully Qualified Domain Name (FQDN) which shall resolve to `address`

**id**  An integral, unique identifier for this static DNS entry

**ttl**  The Time To Live (TTL) of this static DNS entry in seconds

**type**  The name of the type of this static DNS entry

**typeId**  The integral, unique identifier of the type of this static DNS entry

---

# 429: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: +FaYmpn1IIzVSbq0nosw29NZcV9xPh1VgWuUqXUyiDihVUSzX4jrdAloRDgzDvKsYQB8LSkPdGHwt1
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 19:59:56 GMT
```

(continues on next page)
DELETE

New in version 1.3.
Delete staticdnsentries.

Auth. Required  Yes
Roles Required “admin” or “operations”
Response Type undefined

Request Structure

Table 215: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The integral, unique identifier of the static DNS entry to delete</td>
</tr>
</tbody>
</table>

# 430: Request Example

```
DELETE /api/1.4/staticdnsentries?id=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```
**Response Structure**

# 431: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: g6uqHPU44LuTtgU2hatazrVCpcpNWVc9kWJQ0YRuiVLDnsm39KOB/xt3XM6j0/X3WyIiawnNspkxRC85LJHwFA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:05:52 GMT
Content-Length: 69

{
  "alerts": [
    {
      "text": "staticDNSEntry was deleted.",
      "level": "success"
    }
  ]
}
```

**statuses**

**GET**

Retrieves a list of all server statuses.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>no</td>
<td>Return only statuses with this exact description</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the status with this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Return only statuses with this name</td>
</tr>
</tbody>
</table>
# 432: Request Example

```sh
GET /api/1.4/statuses?name=REPORTED HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

## Response Structure

- **description** A short description of the status
- **id** The integral, unique identifier of this status
- **lastUpdated** The date and time at which this status was last modified, in ISO format
- **name** The name of the status

# 433: Response Example

```sh
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: dHNip9kpTGGS1w39/fWcFehNktgmXZus8XaufnmDpv0PyG/3fK/KfoCO3ZOj9V74/CCffps7doEygWeL/xRtKA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:56:59 GMT
Content-Length: 150

```

```json
{
  "response": [
    {
      "description": "Server is online and reported in the health protocol.",
      "id": 3,
      "lastUpdated": "2018-12-10 19:11:17+00",
      "name": "REPORTED"
    }
  ]
}
```

`statuses/{{ID}}`

Deprecated since version 1.1: Use the `id` query parameter of `statuses/{{ID}}` instead.
GET

Retrieves information about a particular status

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the status being inspected</td>
</tr>
</tbody>
</table>

# 434: Request Example

```plaintext
GET /api/1.4/statuses/3 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

description A short description of the status
id The integral, unique identifier of this status
lastUpdated The date and time at which this status was last modified, in ISO format
name The name of the status

# 435: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: dHNip9kpTGG5lw39/4fWcFehNktgmXZus8XaufnmDpv0PyG/3fK/KfoCO3Z0j9V74/CCffps7doEygWeL/xRtKA==
```
steering/{{ID}}/targets

GET

Get all targets for a steering Delivery Service.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 218: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a steering Delivery Service for which targets shall be listed</td>
</tr>
</tbody>
</table>

Table 219: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>Return only the target mappings that target the Delivery Service identified by this integral, unique identifier</td>
</tr>
</tbody>
</table>

# 436: Request Structure

GET /api/1.1/steering/2/targets?target=1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0

(continues on next page)
Response Structure

**deliveryService**  The ‘xml_id’ of the steering Delivery Service

**deliveryServiceId**  An integral, unique identifier for the steering Delivery Service

**target**  The ‘xml_id’ of this target Delivery Service

**targetId**  An integral, unique identifier for this target Delivery Service

**type**  The routing type of this target Delivery Service

**typeId**  An integral, unique identifier for the routing type of this target Delivery Service

**value**  The ‘weight’ attributed to this steering target

# 437: Response Example

```json
{
  "response": [
    {
      "deliveryService": "test",
      "deliveryServiceId": 2,
      "target": "demo1",
      "targetId": 1,
      "type": "HTTP",
      "typeId": 1,
      "value": 100
    }
  ]
}
```

5.1. Traffic Ops API 677
POST

Create a steering target.

**Auth. Required**  Yes

**Roles Required**  Portal, Steering, Federation, “operations” or “admin”

**Response Type**  Object

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a steering <em>Delivery Service</em> to which a target shall be added</td>
</tr>
</tbody>
</table>

**targetId**  The integral, unique identifier of a *Delivery Service* which shall be a new steering target for the *Delivery Service* identified by the ID path parameter

**typeId**  The integral, unique identifier of the routing type of the new target *Delivery Service*

**value**  The ‘weight’ which shall be attributed to the new target *Delivery Service*

### # 438: Request Example

```
POST /api/1.1/steering/2/targets HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 43
Content-Type: application/json

{
    "targetId": 1,
    "value": 100,
    "typeId": 1
}
```

**Response Structure**

- **deliveryService**  The ‘xml_id’ of the steering *Delivery Service*
- **deliveryServiceId**  An integral, unique identifier for the steering *Delivery Service*
- **targetId**  An integral, unique identifier for the newly added target *Delivery Service*
**type**  The routing type of the newly added target *Delivery Service*

**typeld**  An integral, unique identifier for the routing type of the newly added target *Delivery Service*

**value**  The ‘weight’ attributed to the new steering target

---

**HTTP/1.1 200 OK**

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: +dTvfzrnOhdwAOOMyY28r0+gFV5z+3aAB12FfAMziTYcU+p2rDanrJzMXpKWIL5Q/+oCUB2pJDrt9hRCFKT4oGYw==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 21:22:17 GMT
Content-Length: 196
```

```json
{
    "alerts": [
        {
            "text": "steeringtarget was created.",
            "level": "success"
        }
    ],
    "response": {
        "deliveryService": "test",
        "deliveryServiceId": 2,
        "target": "demo1",
        "targetId": 1,
        "type": "HTTP",
        "typeId": 1,
        "value": 100
    }
}
```

---

**steering/{{ID}}/targets/{{targetID}}**

**GET**

Deprecated since version 1.1: Use the `target` query parameter of a GET request to the `steering/{{ID}}/targets` endpoint instead.

Get a single target for a specific STEERING *Delivery Service*.

**Auth. Required**  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 221: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a steering Delivery Service</td>
</tr>
<tr>
<td>targetID</td>
<td>The integral, unique identifier of a Delivery Service which is a target of the Delivery Service identified by ID</td>
</tr>
</tbody>
</table>

# 440: Request Example

```
GET /api/1.1/steering/2/targets/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

deliveryService  The ‘xml_id’ of the steering Delivery Service
deliveryServiceId  An integral, unique identifier for the steering Delivery Service
target  The ‘xml_id’ of this target Delivery Service
targetId  An integral, unique identifier for this target Delivery Service
type  The routing type of this target Delivery Service
typeId  An integral, unique identifier for the routing type of this target Delivery Service
value  The ‘weight’ attributed to this steering target

# 441: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: utlJK4oYS2l6Ff7NzAqRuQeMEtazYn3rM3N1ux2XgTLxvSys1Hy0mJrwDExSU05gVMDrgYCLzrZEvP...
```
PUT

Updates a steering target.

**Auth. Required** Yes

**Roles Required** Portal, Steering, Federation, “operations” or “admin”

**Response Type** Object

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a steering <em>Delivery Service</em></td>
</tr>
<tr>
<td>targetID</td>
<td>The integral, unique identifier of a <em>Delivery Service</em> which is a target of the <em>Delivery Service</em> identified by ID</td>
</tr>
</tbody>
</table>

**typeId** The integral, unique identifier of the routing type of the target *Delivery Service*

**value** The ‘weight’ which shall be attributed to the target *Delivery Service*

# 442: Request Example

```
PUT /api/1.4/steering/2/targets/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 26
```
Response Structure

**deliveryService**  The ‘xml_id’ of the steering Delivery Service

**deliveryServiceId**  An integral, unique identifier for the steering Delivery Service

**target**  The ‘xml_id’ of this target Delivery Service

**targetId**  An integral, unique identifier for this target Delivery Service

**type**  The new routing type of this target Delivery Service

**typeId**  An integral, unique identifier for the new routing type of this target Delivery Service

**value**  The new ‘weight’ attributed to this steering target

# 443: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: AfXsIRzdtU3HZYkr93qBMVT2RJ5oTF2u5sKYnd+DSqxZ+RQxY6vXtCupnnXCF9dxMt5QXRW1EFOW/ FBG61FrTg==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 14:34:22 GMT
Content-Length: 194

{ "alerts": [
    {
        "text": "steeringtarget was updated.",
        "level": "success"
    }
],
"response": {
    "deliveryService": "test",
    "deliveryServiceId": 2,
    "target": "test",
    "targetId": 2,
    "type": "test",
    "typeId": 2,
    "value": 1
}
```
DELETE

Removes a specific target mapping from a specific Delivery Service

Auth. Required  Yes
Roles Required  Portal, Steering, Federation, “operations” or “admin”
Response Type  undefined

Request Structure

Table 223: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of a steering Delivery Service - a target of which shall be deleted</td>
</tr>
<tr>
<td>targetID</td>
<td>The integral, unique identifier of a Delivery Service which is a target to be removed of the Delivery Service identified by ID</td>
</tr>
</tbody>
</table>

# 444: Request Example

DELETE /api/1.4/steering/2/targets/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

# 445: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
system/info

GET

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

No parameters available.

Response Structure

parameters  An object containing information about the Traffic Ops server

Note: These are all the parameters in the GLOBAL profile, so the keys below are merely those present by default required for Traffic Control to operate

default_geo_miss_latitude  The default latitude used when geographic lookup of an IP address fails

default_geo_miss_longitude  The default longitude used when geographic lookup of an IP address fails

tm.logourl  This is the URL of the logo for Traffic Ops and is assumed relative to tm.url if it is a relative path
**tm.instance_name**  The name of the Traffic Ops instance; typically used when multiple instances are active

**tm.toolname**  The name of the Traffic Ops tool (usually “Traffic Ops”) - used in several API endpoints and written in comment headers on most Apache Traffic Server (ATS) configuration files generated by Traffic Ops

**tm.url**  The URL for this Traffic Ops instance - used in several API endpoints and written in comment headers on most Apache Traffic Server (ATS) configuration files generated by Traffic Ops

**use_reval_pending**  A string containing an integer which represents a boolean value (hold your applause); one of:

- **“0”**  Do not use pending revalidations - this effectively prohibits the use of “Content Invalidation Jobs”

- **“1”**  Use pending revalidations - this effectively enables the use of “Content Invalidation Jobs”

**use_tenancy**  A string containing an integer which represents a boolean value; one of:

- **“0”**  Do not use tenancy - this effectively disables all *tenant* endpoints and removes tenancy restrictions on origins and Delivery Services

- **“1”**  Use tenancy - this effectively enables all *tenant* endpoints and enforces tenancy restrictions on origins and Delivery Services

---

**HTTP/1.1 200 OK**

Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: ObxOxK1jrC1/ ...
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 19:06:01 GMT
Content-Length: 285

```json
{
  "response": {
    "parameters": {
      "default_geo_miss_latitude": "0",
      "default_geo_miss_longitude": "-1"
    }
  }
}
```

(continues on next page)
"tm.instance_name": "CDN-In-A-Box",
"tm.logourl": "/images/tc_logo.png",
"tm.toolname": "Traffic Ops",
"tm.url": "https://trafficops.infra.ciab.test:443/",
"use_reval_pending": "0",
"use_tenancy": "1"
}
}

## tenants

### GET

Get all requested tenants.

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

### Request Structure

Table 224: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>If true, return only active tenants; if false return only inactive tenants</td>
</tr>
<tr>
<td>id</td>
<td>Return only tenants with this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>Return only tenants with this name</td>
</tr>
</tbody>
</table>

### # 447: Request Example

```
GET /api/1.4/tenants?name=root HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

### Response Structure

- **active**  A boolean which indicates whether or not the tenant is active
- **id**  The integral, unique identifier of this tenant
- **name**  This tenant’s name
- **parentId**  The integral, unique identifier of this tenant’s parent
parentName  The name of the parent of this tenant

# 448: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Yzr6TfhxgpZ3pbbrr4TRG4wC3PlnHDdzgs2igtz/
1ppLSy2MzugqaGW4y5yzwz1T3+7q6HWej7GQZt1XIVeZQ==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 19:57:58 GMT
Content-Length: 106

{ "response": [
    {
        "id": 1,
        "name": "root",
        "active": true,
        "lastUpdated": "2018-12-10 19:11:17+00",
        "parentId": null
    }
]}
```

**POST**

**Warning:** Inactive tenants cannot be deleted or modified in any way, so be careful whenever setting `active` to `false`. This is a known bug in Traffic Ops, and is tracked by GitHub Issue #2732.

Create a new tenant.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

**Request Structure**

- **active**  An optional boolean - default: `false` - which indicates whether or not the tenant shall be immediately active

- **name**  The name of the tenant
**parentId** The integral, unique identifier of the parent of this tenant

---

### # 449: Request Example

```
POST /api/1.4/tenants HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 48
Content-Type: application/json

{
    "active": true,
    "name": "test",
    "parentId": 1
}
```  

---

### Response Structure

- **active** A boolean which indicates whether or not the tenant is active
- **id** The integral, unique identifier of this tenant
- **name** This tenant’s name
- **parentId** The integral, unique identifier of this tenant’s parent

---

### # 450: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 19:37:16 GMT
Content-Length: 162

{
    "alerts": [
        {
            "text": "tenant was created."
        },
        {
            "level": "success"
        }
    ],
```  

(continues on next page)
"response": {
  "id": 9,
  "name": "test",
  "active": true,
  "lastUpdated": "2018-12-11 19:37:16+00",
  "parentId": 1
}

`tenants/{{ID}}`

**GET**

Deprecated since version 1.1: Use the `id` query parameter of a GET request to `tenants` instead.

Get a specific tenant.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

### Request Structure

#### Table 225: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the tenant being inspected</td>
</tr>
</tbody>
</table>

#### # 451: Request Structure

<table>
<thead>
<tr>
<th>GET</th>
<th><code>/api/1.4/tenants/1</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>Host:</td>
<td>trafficops.infra.ciab.test</td>
</tr>
<tr>
<td>User-Agent:</td>
<td>curl/7.47.0</td>
</tr>
<tr>
<td>Accept:</td>
<td><em>/</em></td>
</tr>
<tr>
<td>Cookie:</td>
<td>mojolicious=...</td>
</tr>
</tbody>
</table>

### Response Structure

- **active** A boolean which indicates whether or not the tenant is active
- **id** The integral, unique identifier of this tenant
- **name** This tenant’s name
- **parentId** The integral, unique identifier of this tenant’s parent
# 452: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: Yzr6Tfhxgp23pbbrr4TRG4wC3PlnHDDzgs2igtz/
    lppLSy2MzugqaGW4y5yzwz15T3+7q6HWej7GQ2t1XIVeZQ==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 20:00:28 GMT
Content-Length: 106

{
    "response": [     
        {
            "id": 1,
            "name": "root",
            "active": true,
            "lastUpdated": "2018-12-10 19:11:17+00",
            "parentId": null
        }
    ]
}
```

**PUT**

**Warning:** Inactive tenants cannot be deleted or modified in any way by this request method, so be careful whenever setting `active` to false. This is a known bug in Traffic Ops, and is tracked by GitHub Issue #2732.

Updates a specific tenant.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

### Request Structure

Table 226: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the tenant being modified</td>
</tr>
</tbody>
</table>
active  An optional boolean - default: false - which indicates whether or not the tenant shall be immediately active

name    The name of the tenant

parentId The integral, unique identifier of the parent of this tenant

# 453: Request Example

```plaintext
PUT /api/1.4/tenants/9 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 59
Content-Type: application/json

{
    "active": true,
    "name": "quest",
    "parent_id": 3
}
```

Response Structure

active  A boolean which indicates whether or not the tenant is active

id      The integral, unique identifier of this tenant

name    This tenant’s name

parentId The integral, unique identifier of this tenant’s parent

# 454: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 928qyd2Bfgw21Wv85rqjLpyeT3djkfRVD1/xpKConu1NrZs2czJKrrwZA7X6lw==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 20:30:54 GMT
Content-Length: 163

{ "alerts": [ 
    
] }
```
Warning: Inactive tenants cannot be deleted or modified in any way by this request method, so be careful whenever setting `active` to `false`. This is a known bug in Traffic Ops, and is tracked by GitHub Issue #2732.

Deletes a specific tenant.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  undefined

Request Structure

Table 227: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier for the tenant being deleted</td>
</tr>
</tbody>
</table>

# 455: Request Example

```
DELETE /api/1.4/tenants/9 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure
# 456: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: KU0XIbFoD0Cy06kzH2Gl59pBqie/TEFJgh33mssGNwXJZ1RkTLaSTHT8Df4X+p0s7uauZH10akGvaA0UTiN/vg==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 20:40:31 GMT
Content-Length: 61

{
   "alerts": [
      {
         "text": "tenant was deleted.",
         "level": "success"
      }
   ]
}
```

to_extensions

See also:

*Managing Traffic Ops Extensions*

GET

Retrieves the list of Traffic Ops extensions.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available.

Response Structure

*additional_config_json* A string containing a JSON-encoded object with extra configuration options... inside a JSON object...
### description
A short description of the extension

**Note:** This is, unfortunately, null for all default plugins

<table>
<thead>
<tr>
<th>id</th>
<th>An integral, unique identifier for this extension definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>info_url</td>
<td>A URL where info about this extension may be found</td>
</tr>
<tr>
<td>isactive</td>
<td>An integer describing the boolean notion of whether or not the plugin is active; one of:</td>
</tr>
<tr>
<td>0</td>
<td>disabled</td>
</tr>
<tr>
<td>1</td>
<td>enabled</td>
</tr>
<tr>
<td>name</td>
<td>The name of the extension</td>
</tr>
<tr>
<td>script_file</td>
<td>The base filename of the script that runs for the extension</td>
</tr>
<tr>
<td>servercheck_shortname</td>
<td>The name of the column in the table at ‘Monitor’ -&gt; ‘Cache Checks’ in Traffic Portal, where “Check Extension” output is displayed</td>
</tr>
</tbody>
</table>

**Note:** This field has meaning only for “Check Extensions”

| type | The type of extension - there are a set number of allowed values which are not recorded anywhere at the time of this writing |
| version | A (hopefully) semantic version number describing the version of the plugin |

#### # 457: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Tue, 11 Dec 2018 20:51:48 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 12 Dec 2018 00:51:48 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: n73jg9XR4V5Cwqq56Rf3wuIi99k3mM5u2NAjcZ/9gQu8vAFym1n2zgReJ+wT11lvjIsHpxCOVVXV7+5UGakLgA==
Transfer-Encoding: chunked

{ "response": [ (continues on next page) }
POST

Creates a new Traffic Ops extension.

Auth. Required  Yes

Roles Required  None

Response Type  undefined

Request Structure

**additional_config_json** An optional string containing a JSON-encoded object with extra configuration options... inside a JSON object...

**description** A short description of the extension

---

1 No roles are required to use this endpoint, however access is controlled by username. Only the reserved user `extension` is permitted the use of this endpoint.
**info_url**  A URL where info about this extension may be found

**isactive**  An integer describing the boolean notion of whether or not the plugin is active; one of:

- 0 disabled
- 1 enabled

**name**  The name of the extension

**script_file**  The base filename of the script that runs for the extension

See also:  

*Managing Traffic Ops Extensions* for details on where the script should be located on the Traffic Ops server

**servercheck_shortname**  The name of the column in the table at ‘Monitor’ -> ‘Cache Checks’ in Traffic Portal, where “Check Extension” output is displayed

---

**Note:**  This field has meaning only for “Check Extensions”

**type**  The type of extension - there are a set number of allowed values which are not recorded anywhere at the time of this writing

**version**  A (hopefully) semantic version number describing the version of the plugin

---

# 458: Request Example

```json
POST /api/1.4/to_extensions HTTP/1.1
Host: ipcdn-cache-51.cdnlab.comcast.net:6443
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 208
Content-Type: application/json

{
    "name": "test",
    "version": "0.0.1-1",
    "info_url": "",
    "script_file": "",
    "isactive": "0",
    "description": "A test extension for API examples",
    "servercheck_short_name": "test",
    "type": "CHECK_EXTENSION_NUM"
}
```

---

696 Chapter 5. APIs
Response Structure

# 459: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 12 Dec 2018 16:37:44 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 12 Dec 2018 20:37:44 GMT;
Vary: Accept-Encoding
Whole-Content-Sha512: 7M67PYnli6WzGQFS3g8Gh1SOyg6VENZMqm/kUffOTLLPfuWSEuSLA65R5R+VyJiNjdqOG5Bp78mk+YqcqtVGw==
Content-Length: 89

{ "supplemental":
  { "id": 5
  },
  "alerts": [{
    "level": "success",
    "text": "Check Extension Loaded."
  }]
}
```

to_extensions/{{ID}}/delete

**POST**

Deletes a Traffic Ops extension definition. This does not delete the actual extension file.

**Auth. Required** Yes

**Roles Required** None

**Response Type** undefined

---

1 No roles are required to use this endpoint, however access is controlled by username. Only the reserved user extension is permitted the use of this endpoint.

5.1. Traffic Ops API
Request Structure

Table 228: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the extension definition to be deleted</td>
</tr>
</tbody>
</table>

# 460: Request Example

```plaintext
POST /api/1.4/to_extensions/16/delete HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 461: Response Example

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Wed, 12 Dec 2018 16:33:52 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 12 Dec 2018 20:33:52 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: ~EB0NuM65azbGzaehDAODP3NFqWbByIza1XQhgwtsW2WTXyK/DxQtncep0YiJXy00tH9H+n6BBfojBOb5h0dFPA==
Content-Length: 60

{ "alerts": [
    {
        "level": "success",
        "text": "Extension deleted."
    }
]}
```
**traffic_monitor/stats**

Deprecated since version TrafficControl: 3.0.0 This endpoint was used by the now-deprecated Traffic Ops UI, and will likely be removed in the future!

**Caution:** This page is a stub! Much of it may be missing or just downright wrong - it needs a lot of love from people with the domain knowledge required to update it.

**GET**

**Auth. Required** Yes

**Roles Required** None

**Response Type** NOT PRESENT - this endpoint returns a special, custom JSON response

**Request Structure**

No parameters available.

**Response Structure**

*aaData* An array of data points of some kind

---

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Date: Mon, 03 Dec 2018 14:44:14 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 03 Dec 2018 18:44:14 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: yRHVMHW+Y78HgaU/UVcrcADg9Jw3ScP+IQEEVqy3R/0A757Wm2ZpmGDECDkDp7crWckabMntHRlaf/6hWJPoQ==
Content-Length: 57

{ "aaData": [ ]
```

(continues on next page)
types

GET

Retrieves all of the types of things configured in Traffic Ops. Yes, that is as specific as a description of a ‘type’ can be.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 229: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the type that is identified by this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Return only types with this name</td>
</tr>
<tr>
<td>useInTable</td>
<td>no</td>
<td>Return only types that are used to identify the type of the object stored in the Traffic Ops database table that has this name</td>
</tr>
</tbody>
</table>

GET /api/1.4/types?name=TC_LOC HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Response Structure

- **description** A short description of this type
- **id** An integral, unique identifier for this type
- **lastUpdated** The date and time at which this type was last updated, in ISO format
- **name** The name of this type
- **useInTable** The name of the Traffic Ops database table that contains objects which are grouped, identified, or described by this type

# 464: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: EH8jo8OrCu79Tz9xpgT3YRyKJ/p2NcTmbS3huwtgRBryHz9H6q2LQja59RIPaVSq32xsU6QhTaox5nBkQ9LPSAA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 12 Dec 2018 22:59:22 GMT
Content-Length: 168

{ "response": [  
  {
      "id": 48,
      "lastUpdated": "2018-12-12 16:26:41+00",
      "name": "TC_LOC",
      "description": "Location for Traffic Control Component Servers",
      "useInTable": "cachegroup"
  }
]}
```

types/{{ID}}

GET

Deprecated since version 1.1: Use the `id` query parameter of a GET request to the `types` endpoint instead.

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Array
Request Structure

Table 230: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the type being inspected</td>
</tr>
</tbody>
</table>

# 465: Request Example

```
GET /api/1.4/types/48 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **description** A short description of this type
- **id** An integral, unique identifier for this type
- **lastUpdated** The date and time at which this type was last updated, in ISO format
- **name** The name of this type
- **useInTable** The name of the Traffic Ops database table that contains objects which are grouped, identified, or described by this type

# 466: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: EH8jo8OrCu79Tz9xpT3YRyKJ/p2NcTmbS3huwtgRByHz9H6q2LQja59RIPAeVSq3ZxsU6QhTaox5nBkQ9LPSAA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 12 Dec 2018 23:50:13 GMT
Content-Length: 168

{ "response": [
    {
        "id": 48,
        "lastUpdated": "2018-12-12 16:26:41+00",
        "name": "TC_LOC",
    }
]}
```
types/trimmed

Deprecated since version 1.1: A type is not a large object. Just get the names from types instead.

GET

Retrieves only the names of all of the types of things configured in Traffic Ops. Yes, that is as specific as a description of a ‘type’ can be.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available

Response Structure

name The name of the type

# 467: Response Example

HTTP/1.1 200 OK
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Set-Cookie: mojolicious=...; expires=Thu, 13 Dec 2018 03:37:01 GMT; path=/; HttpOnly
Whole-Content-SHA512: Wh4z9VkNcO18UzSTM77N+JFx5bP8yxRR4rglfZIH4ODI+0suOD36YhePUMMqM16DIIWjrnkj+ioju
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Date: Wed, 12 Dec 2018 23:37:01 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Content-Length: 1104
Note: The response example for this endpoint has been truncated to only the first two elements of the resulting array, as the output was hundreds of lines long.

user/current

GET

Deprecated since version 1.4: As a username is needed to log in, any administrator or application must necessarily know the current username at any given time. Thus, use the username query parameter of a GET request to users instead.

Retrieves the profile for the authenticated user.

Auth. Required Yes

Roles Required None

Response Type Object

Request Structure

No parameters available.

Response Structure

addressLine1 The user’s address - including street name and number

addressLine2 An additional address field for e.g. apartment number

city The name of the city wherein the user resides

company The name of the company for which the user works

country The name of the country wherein the user resides
email The user’s email address

fullName The user’s full name, e.g. “John Quincy Adams”

gid A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user - now it is always null

id An integral, unique identifier for this user

lastUpdated The date and time at which the user was last modified, in ISO format

newUser A meta field with no apparent purpose that is usually null unless explicitly set during creation or modification of a user via some API endpoint

phoneNumber The user’s phone number

postalCode The postal code of the area in which the user resides

publicSshKey The user’s public key used for the SSH protocol

registrationSent If the user was created using the users/register endpoint, this will be the date and time at which the registration email was sent - otherwise it will be null

role The integral, unique identifier of the highest-privilege role assigned to this user

rolename The name of the highest-privilege role assigned to this user

stateOrProvince The name of the state or province where this user resides

tenant The name of the tenant to which this user belongs

tenantId The integral, unique identifier of the tenant to which this user belongs

uid A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user - now it is always null

username The user’s username

# 468: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: HQwu9FxPyinXSVFK5+wPeHsX60KbqXuokFbM23Oe0oerOoM5ZpWpglsH7zMrch8V8Aw0dzwJSzpPJivj
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 15:14:45 GMT
Content-Length: 382

{ "response": { (continues on next page) ```
"username": "admin",
"localUser": true,
"addressLine1": null,
"addressLine2": null,
"city": null,
"company": null,
"country": null,
"email": null,
"fullName": null,
"gid": null,
"id": 2,
"newUser": false,
"phoneNumber": null,
"postalCode": null,
"publicSshKey": null,
"role": 1,
"rolename": "admin",
"stateOrProvince": null,
"tenant": "root",
"tenantId": 1,
"uid": null,
"lastUpdated": "2018-12-12 16:26:32+00"
}}

**PUT**

Deprecated since version 1.4: Use the PUT method of the `users` instead.

**Warning:** Users that login via LDAP pass-back cannot be modified

Updates the date for the authenticated user.

**Auth. Required** Yes

**Roles Required** None

**Response Type** undefined

**Request Structure**

- **addressLine1** An optional field which should contain the user’s address - including street name and number
- **addressLine2** An optional field which should contain an additional address field for e.g. apartment number
**city** An optional field which should contain the name of the city wherein the user resides

**company** An optional field which should contain the name of the company for which the user works

**confirmLocalPasswd** The ‘confirm’ field in a new user’s password specification - must match **localPasswd**

**country** An optional field which should contain the name of the country wherein the user resides

**email** The user’s email address

Changed in version 1.4: Prior to version 1.4, the email was validated using the Email::Valid Perl package but is now validated (circuitously) by GitHub user asaskevich’s regular expression. Note that neither method can actually distinguish a valid, deliverable, email address but merely ensure the email is in a commonly-found format.

**fullName** The user’s full name, e.g. “John Quincy Adams”

**localPasswd** The user’s password

**newUser** An optional meta field with no apparent purpose - don’t use this

**phoneNumber** An optional field which should contain the user’s phone number

**postalCode** An optional field which should contain the user’s postal code

**publicSshKey** An optional field which should contain the user’s public encryption key used for the SSH protocol

**role** The number that corresponds to the highest permission role which will be permitted to the user

**stateOrProvince** An optional field which should contain the name of the state or province in which the user resides

**tenantId** The integral, unique identifier of the tenant to which the new user shall belong

**username** The user’s new username

---

**Note:** This field is optional if and only if tenancy is not enabled in Traffic Control

---

# 469: Request Example

```
PUT /api/1.4/user/current HTTP/1.1
Host: trafficsops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

(continues on next page)
Content-Length: 483
Content-Type: application/json

```
{
    "user": {
        "addressLine1": "not a real address",
        "addressLine2": "not a real address either",
        "city": "not a real city",
        "company": "not a real company",
        "country": "not a real country",
        "email": "not@real.email",
        "fullName": "Not a real fullName",
        "phoneNumber": "not a real phone number",
        "postalCode": "not a real postal code",
        "publicSshKey": "not a real ssh key",
        "stateOrProvince": "not a real state or province",
        "tenantId": 1,
        "role": 1,
        "username": "admin"
    }
}
```

Response Structure

# 470: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 21:05:49 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 14 Dec 2018 01:05:49 GMT;
PATH=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: sHFgZQ4Cv7IIWaIejoAvM2Fr/HSupcX3D16KU/+etjw+4jck9EME3Bq5ohLC+eQ52BDCKW2Ra+AC3TfFtworJww==
Content-Length: 79

```
{
    "alerts": [
        {
            "level": "success",
            "text": "User profile was successfully updated"
        }
    ]
}
```
user/current/jobs

GET

 Deprecated since version 1.1: Use the userId query parameter of a GET request to the jobs endpoint instead.

Retrieves the user’s list of running and pending content invalidation jobs.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

Request Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-word</td>
<td>no</td>
<td>Return only jobs that have this keyword - keyword corresponds to the operation or type of job (currently only “PURGE” is a valid keyword)</td>
</tr>
</tbody>
</table>

Deprecated since version 1.1: This query parameter has been deprecated because the only supported keyword is “PURGE”. Jobs used to be much more versatile, but such versatility is no longer required of them. This still “works”, but never has any effect on the output, except to make it an empty array if it is anything other than “PURGE”.

# 471: Request Example

```
GET /api/1.4/user/current/jobs?keyword=PURGE HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

**agent** The name of the agent or process responsible for running the job

Deprecated since version 1.1: This field is no longer used, but does still exist for legacy compatibility reasons. It will always be "dummy".

**assetUrl** A regular expression - matching URLs will be operated upon according to keyword

**assetType** The type of asset being revalidated e.g. “file”
Deprecated since version 1.1: This field still exists, but has no purpose as all assets are now treated as remote files; i.e. it will always be "file".

**createdBy**  
The username of the user who initiated the job

**deliveryService**  
The ‘xml_id’ that uniquely identifies the Delivery Service on which this job operates

**enteredTime**  
The date and time at which the job was created, in ISO format

**id**  
An integral, unique identifier for this job

**keyword**  
A keyword that represents the operation being performed by the job:

- **PURGE**  
  This job will prevent caching of URLs matching the assetURL until it is removed (or its Time to Live expires)

**objectName**  
A deprecated field of unknown use - it only still exists for legacy compatibility reasons, and will always be `null`

**objectType**  
A deprecated field of unknown use - it only still exists for legacy compatibility reasons, and will always be `null`

**parameters**  
A string containing key/value pairs representing parameters associated with the job - currently only uses Time to Live e.g. "TTL:48h"

**startTime**  
The date and time at which the job began or will begin, in ISO format

**status**  
A deprecated field of unknown use - it only still exists for legacy compatibility reasons, and appears to always be "PENDING"

**username**  
The username of the user who created this revalidation job

---

### 472: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 14:23:54 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 13 Dec 2018 18:23:54 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Ijr9pDZ4XwPIBX0Qnl+yihTYa8bK7TjJdPrDiV9VNq9k7OC9FSNQV4HSmX35KUAKMFpIHe/azutbvr0xZzQucg==
Content-Length: 301

{ "response": []
}
```
POST

Creates a new content revalidation job.

**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 HTTP response as appropriate.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”¹

**Response Type**  undefined

**Request Structure**

- **dsId**  The integral, unique identifier of the Delivery Service on which the revalidation job shall operate
- **regex**  This should be a PCRE-compatible regular expression for the path to match for forcing the revalidation

**Warning:** This is concatenated directly to the origin URL of the Delivery Service identified by dsId to make the full regular expression. Thus

¹ A role is only required if tenancy is not used; if tenancy is used by Traffic Control, then the user will be able to create the content revalidation job on Delivery Services scoped to his or her tenancy regardless of role. This means that even read-only users can create content invalidation jobs for Delivery Services scoped to their tenancy. This behavior is considered a bug, and it is tracked by GitHub Issue #3116.
it is not necessary to restate the URL but it should be noted that if the ori-
gin URL does not end with a backslash (/) then this should begin with
an escaped backslash to ensure proper behavior (otherwise it will match
against FQDNs, which leads to undefined behavior in Traffic Control).

**Note:** Be careful to only match on the content that must be removed -
revalidation is an expensive operation for many origins, and a simple \./.*
can cause an overload in requests to the origin.

**startTime** The time and date at which the revalidation rule will be made active, in
ISO format

**ttl** Specifies the Time To Live (TTL) - in hours - for which the revalidation rule
will remain active after **startTime**

**Note:** It usually makes sense to make this the same as the
Cache-Control header from the origin which 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.

**urgent** An optional boolean which, if present and true, marks the job as “ur-
gent”, which has no meaning to machines but is visible to humans for their
consideration

### # 473: Request Example

```plaintext
POST /api/1.4/user/current/jobs HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 79
Content-Type: application/json

{
    "dsId": 1,
    "regex": "\/.\.*\\.jpg",
    "startTime": "2018-12-13 13:55:09",
    "ttl": 1
}
```

**Response Structure**
# 474: Response Example

```http
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 13:56:35 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 13 Dec 2018 17:56:35 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Uyz2P6gkzsSu8xESEHSKQCG6+6Xw0o+wgjx30+UTBFNIZzFYlkjDKlwZdIUYUpdSbPcTRy5ZaxT1qF
Content-Length: 141

{ "alerts": [ { "level": "success", "text": "Invalidate content request submitted for demo1 [ http://origin.infra.ciab.test.*\.jpg - TTL:1h ]" } ]}
```

**user/current/update**

Deprecated since version 1.4: Use the **PUT** method of **users** instead.

**POST**

Updates the date for the authenticated user.

**Auth. Required** Yes

**Roles Required** None

**Response Type** undefined

**Request Structure**

- **addressLine1** An optional field which should contain the user’s address - including street name and number

- **addressLine2** An optional field which should contain an additional address field for e.g. apartment number
city  An optional field which should contain the name of the city wherein the user resides

country  An optional field which should contain the name of the country wherein the user resides

e-mail  The user’s e-mail address

Changed in version 1.4: Prior to version 1.4, the email was validated using the Email::Valid Perl package but is now validated (circuitously) by GitHub user asaskevich’s regular expression. Note that neither method can actually distinguish a valid, deliverable, email address but merely ensure the email is in a commonly-found format.

fullName  The user’s full name, e.g. “John Quincy Adams”

confirmLocalPasswd  The ‘confirm’ field in a new user’s password specification - must match localPasswd

localPasswd  The user’s password

newUser  An optional meta field with no apparent purpose - don’t use this

phoneNumber  An optional field which should contain the user’s phone number

postalCode  An optional field which should contain the user’s postal code

publicSshKey  An optional field which should contain the user’s public encryption key used for the SSH protocol

role  The number that corresponds to the highest permission role which will be permitted to the user

stateOrProvince  An optional field which should contain the name of the state or province in which the user resides

tenantId  The integral, unique identifier of the tenant to which the new user shall belong

Note:  This field is optional if and only if tenancy is not enabled in Traffic Control

username  The user’s new username

# 475: Request Example

```
POST  /api/1.4/user/current/update HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...  
```
Content-Length: 483
Content-Type: application/json

{
  "user": {
      "addressLine1": "not a real address",
      "addressLine2": "not a real address either",
      "city": "not a real city",
      "company": "not a real company",
      "country": "not a real country",
      "email": "not@real.email",
      "fullName": "Not a real fullName",
      "phoneNumber": "not a real phone number",
      "postalCode": "not a real postal code",
      "publicSshKey": "not a real ssh key",
      "stateOrProvince": "not a real state or province",
      "tenantId": 1,
      "role": 1,
      "username": "admin"
  }
}

Response Structure

# 476: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 21:04:36 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 14 Dec 2018 01:04:36 GMT;
  path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: sHFqZQ4Cv7IIWaIejoAvM2Fr/HSupcX3D16KU/etjw+4jCK9EME3Bq5ohLC+eQ52BDCKW2Ra+AC3TfFtworJww==
Content-Length: 79

{
  "alerts": [
    {
      "level": "success",
      "text": "User profile was successfully updated"
    }
  ]}
user/{{ID}}/deliveryservices/available

**GET**

Lists identifying information for all of the *Delivery Services* assigned to a user - not, as the name implies, the *Delivery Services available* to be assigned to that user.

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the users whose <em>Delivery Services</em> shall be retrieved</td>
</tr>
</tbody>
</table>

**# 477: Request Example**

GET /api/1.4/user/2/deliveryservices/available HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

**Response Structure**

- **displayName**  This *Delivery Service’s* name
- **id**  The integral, unique identifier of this *Delivery Service*
- **xmlId**  The ‘xml_id’ which (also) uniquely identifies this *Delivery Service*

**# 478: Response Example**

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: A1IUM2qkvJkviD0mcEADoCMiy76AWRO/Xnc70ur3Cr0lkySXwqxfrhLc3wK1I1926yW+QrtD3nQaVpbX7Rd9wQ==

(continues on next page)
user/login

POST

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

Auth. Required  No
Roles Required  None
Response Type  undefined

Request Structure

p  Password
u  Username

# 479: Request Example

POST /api/1.4/user/login HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 26
Content-Type: application/json

{  
  "u":  "admin",
  "p":  "twelve"
}
Response Structure

# 480: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Thu, 13 Dec 2018 21:21:33 GMT; HttpOnly
Whole-Content-Sha512: UdO6T3tMNc7uVsDxZrjVwwYOnD7jmnBzPEB9PvOt2bHajTv3SKTPiIZjDzvhU6EX4p+JoG4fA5w1h
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 15:21:33 GMT
Content-Length: 65

{
  "alerts": [
    {
      "text": "Successfully logged in."
    }
  ]
}
```

`user/login/token`

**Caution:** This page is a stub! Much of it may be missing or just downright wrong - it needs a lot of love from people with the domain knowledge required to update it.

**POST**

Authentication of a user using a token. Normally, the token is obtained via a call to either `user/reset_password` or `users/register`.

- **Auth. Required** No
- **Roles Required** None
- **Response Type** undefined

**Request Structure**

- `t` The login token
# 481: Request Example

```
POST /api/1.3/user/login/token HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 44
Content-Type: application/json

{
    "t": "18EE200C-FF24-11E8-BF01-870C776752A3"
}
```

Response Structure

# 482: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 22:16:25 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 14 Dec 2018 02:16:25 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: uDowfYsW7ADm2yfahD21A+KuDdycQ3a4ma5kbPO/9RXsvgL9bgNC00cp14QLxJN1FfeljroYoigcnj1K9KX/5Q==
Content-Length: 65

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

user/logout

**POST**

User logout. Invalidates the session cookie of the currently logged-in user.
**user/reset_password**

**POST**

Sends an email to reset a user’s password.

- **Auth. Required** No
- **Roles Required** None
- **Response Type** undefined

---

**Apache Traffic Control Documentation, Release 3.0.0**

**Auth. Required** Yes

**Roles Required** None

**Response Type** undefined

**Request Structure**

No parameters available

**Response Structure**

```
# 483: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Date: Thu, 13 Dec 2018 21:25:36 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 14 Dec 2018 01:25:36 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 6KEdr1ZC51Zk0103KwvQE0L7qrJ/+ek6ztymkYy9p8gdFUyYyzGEAJ/Ldb8GY0UBFYmgqeZ3yWHvTcEsOQMiw==
Content-Length: 61

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

**email**  The email address of the user to initiate password reset

### #484: Request Example

```plaintext
POST /api/1.3/user/reset_password HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 35
Content-Type: application/json

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

Response Structure

### #485: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 22:11:53 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 14 Dec 2018 02:11:53 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
    lKWwVYbgKk1k7b1jnQJZwVV5b1jIk+GkooD6SAc3CSSexVKvfL9dgL5iBc/
    BNMRk2pI5I/1GglcDLrXsF1ZA==
Content-Length: 109

{
    "alerts": [
        {
            "level": "success",
            "text": "Successfully sent password reset to email 'test@example.com'"
        }
    ]
}
```
users

GET

Retrieves all requested users.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 233: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the user identified by this integral, unique identifier</td>
</tr>
<tr>
<td>tenant</td>
<td>no</td>
<td>Return only users belonging to the tenant identified by this integral, unique identifier</td>
</tr>
<tr>
<td>username</td>
<td>no</td>
<td>Return only the user with this username</td>
</tr>
</tbody>
</table>

New in version 1.4: The id and username query parameters were added in the 1.4 API.

# 486: Request Example

```
GET /api/1.4/users?username=admin HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

addressLine1  The user’s address - including street name and number
addressLine2  An additional address field for e.g. apartment number
city          The name of the city wherein the user resides
country       The name of the country wherein the user resides
date
email         The user’s email address

1 While no roles are required, this endpoint does respect tenancy. A user will only be able to see, create, delete or modify other users belonging to the same tenant, or its descendants.
**fullName**  The user’s full name, e.g. “John Quincy Adams”

**gid**  A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user - now it is always null

**id**  An integral, unique identifier for this user

**lastUpdated**  The date and time at which the user was last modified, in ISO format

**newUser**  A meta field with no apparent purpose that is usually null unless explicitly set during creation or modification of a user via some API endpoint

**phoneNumber**  The user’s phone number

**postalCode**  The postal code of the area in which the user resides

**publicSshKey**  The user’s public key used for the SSH protocol

**registrationSent**  If the user was created using the `users/register` endpoint, this will be the date and time at which the registration email was sent - otherwise it will be null

**role**  The integral, unique identifier of the highest-privilege role assigned to this user

**rolename**  The name of the highest-privilege role assigned to this user

**stateOrProvince**  The name of the state or province where this user resides

**tenant**  The name of the tenant to which this user belongs

**tenantId**  The integral, unique identifier of the tenant to which this user belongs

**uid**  A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user - now it is always null

**username**  The user’s username

---

**# 487: Response Example**

```plaintext
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-\nType, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: ...
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 01:03:53 GMT
Content-Length: 391

{
  "response": [
  ]
}
```

(continues on next page)
POST

Creates a new user.

Auth. Required  Yes
Roles Required  “admin” or “operations”!
Response Type  Object

Request Structure

addressLine1 An optional field which should contain the user’s address - including street name and number
addressLine2 An optional field which should contain an additional address field for e.g. apartment number
city An optional field which should contain the name of the city wherein the user resides
company An optional field which should contain the name of the company for which the user works
confirmLocalPasswd  The ‘confirm’ field in a new user’s password specification
- must match localPasswd

country   An optional field which should contain the name of the country wherein
the user resides

eemail   The user’s email address
    Changed in version 1.4: Prior to version 1.4, the email was validated using
the Email::Valid Perl package but is now validated (circuitously) by GitHub
user asaskevich’s regular expression . Note that neither method can actually
distinguish a valid, deliverable, email address but merely ensure the email is
in a commonly-found format.

fullName   The user’s full name, e.g. “John Quincy Adams”

localPasswd   The user’s password

newUser   An optional meta field with no apparent purpose - don’t use this

phoneNumber   An optional field which should contain the user’s phone number

postalCode   An optional field which should contain the user’s postal code

publicSshKey   An optional field which should contain the user’s public encryption
key used for the SSH protocol

role   The number that corresponds to the highest permission role which will be
permitted to the user

stateOrProvince   An optional field which should contain the name of the state or
province in which the user resides

tenantId   The integral, unique identifier of the tenant to which the new user shall
belong

Note: This field is optional if and only if tenancy is not enabled in Traffic
Control

username   The new user’s username

# 488: Request Example

```json
POST /api/1.1/users HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 304
Content-Type: application/json

{
    "username": "mike",
```

(continues on next page)
Response Structure

- **addressLine1**: The user’s address - including street name and number
- **addressLine2**: An additional address field for e.g. apartment number
- **city**: The name of the city wherein the user resides
- **company**: The name of the company for which the user works
- **country**: The name of the country wherein the user resides
- **email**: The user’s email address
- **fullName**: The user’s full name, e.g. “John Quincy Adams”
- **gid**: A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user - now it is always `null`
- **id**: An integral, unique identifier for this user
- **lastUpdated**: The date and time at which the user was last modified, in ISO format
- **newUser**: A meta field with no apparent purpose that is usually `null` unless explicitly set during creation or modification of a user via some API endpoint
- **phoneNumber**: The user’s phone number
- **postalCode**: The postal code of the area in which the user resides
- **publicSshKey**: The user’s public key used for the SSH protocol
- **registrationSent**: If the user was created using the `users/register` endpoint, this will be the date and time at which the registration email was sent - otherwise it will be `null`
- **role**: The integral, unique identifier of the highest-privilege role assigned to this user
- **roleName**: The name of the highest-privilege role assigned to this user
- **stateOrProvince**: The name of the state or province where this user resides
tenant  The name of the tenant to which this user belongs

tenantId  The integral, unique identifier of the tenant to which this user belongs

uid  A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user - now it is always null

username  The user’s username

# 489: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 02:28:27 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Set-Cookie: mojolicious=...; expires=Thu, 13 Dec 2018 06:28:27 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: vDqbaMvggeoiDs1czqyIW1yDG8WLnCCJdF14Ub05nsE+oJOakkyeZ8odf4d0Zjtqpk01hoVo14H2tj
Content-Length: 520

```json
{
  "alerts": [
    {
      "level": "success",
      "text": "User creation was successful."
    }
  ],
  "response": {
    "registrationSent": null,
    "email": "mwazowski@minc.biz",
    "tenantId": 1,
    "city": "Monstropolis",
    "tenant": "root",
    "id": 8,
    "company": null,
    "roleName": "admin",
    "phoneNumber": null,
    "country": null,
    "fullName": "Mike Wazowski",
    "publicSshKey": null,
    "uid": null,
    "stateOrProvince": null,
    "lastUpdated": null,
    "username": "mike",
  }
}
```

(continues on next page)
"newUser": false,
"addressLine2": null,
"role": 1,
"addressLine1": "22 Mike Wazowski You've Got Your Life Back Lane",
"postalCode": null,
"gid": null
}}

users/{{ID}}

GET

Deprecated since version 1.4: Use the id query parameter of a GET request to the users endpoint instead.

Retrieves a specific user.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 234: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the user to be retrieved</td>
</tr>
</tbody>
</table>

# 490: Request Example

GET /api/1.4/users/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **addressLine1**  The user’s address - including street name and number
- **addressLine2**  An additional address field for e.g. apartment number
- **city**  The name of the city wherein the user resides
company  The name of the company for which the user works

country  The name of the country wherein the user resides

email  The user’s email address

fullName  The user’s full name, e.g. “John Quincy Adams”

gid  A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user - now it is always null

id  An integral, unique identifier for this user

lastUpdated  The date and time at which the user was last modified, in ISO format

newUser  A meta field with no apparent purpose that is usually null unless explicitly set during creation or modification of a user via some API endpoint

phoneNumber  The user’s phone number

postalCode  The postal code of the area in which the user resides

publicSshKey  The user’s public key used for the SSH protocol

registrationSent  If the user was created using the users/register endpoint, this will be the date and time at which the registration email was sent - otherwise it will be null

role  The integral, unique identifier of the highest-privilege role assigned to this user

tenant  The name of the tenant to which this user belongs

tenantId  The integral, unique identifier of the tenant to which this user belongs

uid  A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user - now it is always null

username  The user’s username

# 491: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 9vqUmt8fwruDb+9lQJ4sGbbF420a7uNyBNSWhyzAi3fBUZ5mGhd4Jx5IuS1EqiLZnYeV1JJL8mpRon
X-Server-Name: traffic_ops_golang/
```
PUT

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

Table 235: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the user to be modified</td>
</tr>
</tbody>
</table>

addressLine1 An optional field which should contain the user’s address - including street name and number
addressLine2  An optional field which should contain an additional address field for e.g. apartment number

city  An optional field which should contain the name of the city wherein the user resides

company  An optional field which should contain the name of the company for which the user works

confirmLocalPasswd  The ‘confirm’ field in a new user’s password specification - must match localPasswd

country  An optional field which should contain the name of the country wherein the user resides

eMail  The user’s email address

    Changed in version 1.4: Prior to version 1.4, the email was validated using the Email::Valid Perl package but is now validated (circuitously) by GitHub user asaskevich’s regular expression. Note that neither method can actually distinguish a valid, deliverable, email address but merely ensure the email is in a commonly-found format.

fullName  The user’s full name, e.g. “John Quincy Adams”

localPasswd  The user’s password

newUser  An optional meta field with no apparent purpose - don’t use this

phoneNumber  An optional field which should contain the user’s phone number

postalCode  An optional field which should contain the user’s postal code

publicSshKey  An optional field which should contain the user’s public encryption key used for the SSH protocol

role  The number that corresponds to the highest permission role which will be permitted to the user

stateOrProvince  An optional field which should contain the name of the state or province in which the user resides

tenantId  The integral, unique identifier of the tenant to which the new user shall belong

    Note: This field is optional if and only if tenancy is not enabled in Traffic Control

username  The new user’s username

# 492: Request Structure

PUT /api/1.1/users/2  HTTP/1.1
Host: trafficops.infra.ciab.test

(continues on next page)
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 458
Content-Type: application/json

```json
{
    "addressLine1": "not a real address",
    "addressLine2": "not a real address either",
    "city": "not a real city",
    "company": "not a real company",
    "country": "not a real country",
    "email": "not@real.email",
    "fullName": "Not a real fullName",
    "phoneNumber": "not a real phone number",
    "postalCode": "not a real postal code",
    "publicSshKey": "not a real ssh key",
    "stateOrProvince": "not a real state or province",
    "tenantId": 1,
    "role": 1,
    "username": "admin"
}
```

Response Structure

- **addressLine1** The user’s address - including street name and number
- **addressLine2** An additional address field for e.g. apartment number
- **city** The name of the city wherein the user resides
- **company** The name of the company for which the user works
- **country** The name of the country wherein the user resides
- **email** The user’s email address
- **fullName** The user’s full name, e.g. “John Quincy Adams”
- **gid** A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user - now it is always null
- **id** An integral, unique identifier for this user
- **lastUpdated** The date and time at which the user was last modified, in ISO format
- **newUser** A meta field with no apparent purpose that is usually null unless explicitly set during creation or modification of a user via some API endpoint
- **phoneNumber** The user’s phone number
- **postalCode** The postal code of the area in which the user resides
publicSshKey  The user’s public key used for the SSH protocol

registrationSent  If the user was created using the users/register endpoint, this will be the date and time at which the registration email was sent - otherwise it will be null

role  The integral, unique identifier of the highest-privilege role assigned to this user

roleName  The name of the highest-privilege role assigned to this user

stateOrProvince  The name of the state or province where this user resides

tenant  The name of the tenant to which this user belongs

tenantId  The integral, unique identifier of the tenant to which this user belongs

uid  A deprecated field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user - now it is always null

username  The user's username

# 493: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 17:24:23 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Set-Cookie: mojolicious=...; expires=Thu, 13 Dec 2018 21:24:23 GMT;

Vary: Accept-Encoding

Whole-Content-Sha512: QKvGSIwSdreMI/OdgWv9WQfI/C1JbXSoqGGospTGFcvuj32XWMhmREGzojWsi1W8os8b14TYeyMLUWunf2Ug==
Content-Length: 661

{
    "alerts": [
        {
            "level": "success",
            "text": "User update was successful."
        }
    ],

    "response": {
        "registrationSent": null,
        "email": "not@real.email",
        "tenantId": 1,
        "city": "not a real city",
        "tenant": "root",
    }
}
```
users/{{ID}}/deliveryservices

GET

Retrieves all Delivery Services assigned to the user.

Auth. Required Yes
Roles Required None¹
Response Type Array

Request Structure

Table 236: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The integral, unique identifier of the users whose Delivery Services shall be retrieved</td>
</tr>
</tbody>
</table>

# 494: Request Example

GET /api/1.4/users/2/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0

¹ Users with the roles “admin” and/or “operations” will be able to see all Delivery Services, whereas any other user will only see the Delivery Services their Tenant is allowed to see.
Response Structure

**active**  true if the *Delivery Service* is active, false otherwise

**anonymousBlockingEnabled**  true if *Anonymous Blocking* has been configured for the *Delivery Service*, false otherwise

**cacheurl**  A setting for a deprecated feature of now-unsupported Trafficserver versions

Deprecated since version ATCv3.0: This field has been deprecated in Traffic Control 3.x and is subject to removal in Traffic Control 4.x or later

**ccrDnsTtl**  The Time To Live (TTL) of the DNS response for A or AAAA record queries requesting the IP address of the Traffic Router - named “ccrDnsTtl” for legacy reasons

**cdnId**  The integral, unique identifier of the CDN to which the *Delivery Service* belongs

**cdnName**  Name of the CDN to which the *Delivery Service* belongs

**checkPath**  The path portion of the URL to check connections to this *Delivery Service’s* origin server

**displayName**  The display name of the *Delivery Service*

**dnsBypassCname**  Domain name to overflow requests for HTTP *Delivery Services* - bypass starts when the traffic on this *Delivery Service* exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the *Delivery Service*

**dnsBypassIp**  The IPv4 IP to use for bypass on a DNS *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the *Delivery Service*

**dnsBypassIp6**  The IPv6 IP to use for bypass on a DNS *Delivery Service* - bypass starts when the traffic on this *Delivery Service* exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the *Delivery Service*

**dnsBypassTtl**  The time for which a DNS bypass of this *Delivery Services* shall remain active

**dscp**  The Differentiated Services Code Point (DSCP) with which to mark traffic as it leaves the CDN and reaches clients

---

4 This only applies to DNS-routed *Delivery Services*
edgeHeaderRewrite  Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

fqPacingRate  The Fair-Queuing Pacing Rate in Bytes per second set on the all TCP connection sockets in the Delivery Service (see man tc-fc_codel for more information) - Linux only

geoLimit  The setting that determines how content is geographically limited - this is an integer on the interval [0-2] where the values have these meanings:

0  None - no limitations
1  Only route when the client’s IP is found in the Coverage Zone File (CZF)
2  Only route when the client’s IP is found in the CZF, or when the client can be determined to be from the United States of America

Warning:  This does not prevent access to content or make content secure; it merely prevents routing to the content through Traffic Router

g eoLimitCountries  A string containing a comma-separated list of country codes (e.g. “US,AU”) which are allowed to request content through this Delivery Service

g eoLimitRedirectUrl  A URL to which clients blocked by Regional Geographic Blocking or the geoLimit settings will be re-directed

g eoProvider  An integer that represents the provider of a database for mapping IPs to geographic locations; currently only the following values are supported:

0  The “Maxmind” GeoIP2 database (default)
1  Neustar

globalMaxMbps  The maximum global bandwidth allowed on this Delivery Service. If exceeded, traffic will be routed to dnsBypassIp (or dnsBypassIp6 for IPv6 traffic) for DNS Delivery Services and to httpBypassFqdn for HTTP Delivery Services

globalMaxTps  The maximum global transactions per second allowed on this Delivery Service. When this is exceeded traffic will be sent to the dnsBypassIp (and/or dnsBypassIp6) for DNS Delivery Services and to the httpBypassFqdn for HTTP Delivery Services

httpBypassFqdn  The HTTP destination to use for bypass on an HTTP Delivery Service - bypass starts when the traffic on this Delivery Service exceeds globalMaxMbps, or when more than globalMaxTps is being exceeded within the Delivery Service

id  An integral, unique identifier for this Delivery Service

infoUrl  This is a string which is expected to contain at least one URL pointing to more information about the Delivery Service. Historically, this has been
used to link relevant JIRA tickets

**initialDispersion** The number of caches between which traffic requesting the same object will be randomly split - meaning that if 4 clients all request the same object (one after another), then if this is above 4 there is a possibility that all 4 are cache misses. For most use-cases, this should be 1

**ipv6RoutingEnabled** If true, clients that connect to Traffic Router using IPv6 will be given the IPv6 address of a suitable Edge-tier cache; if false all addresses will be IPv4, regardless of the client connection

**lastUpdated** The date and time at which this Delivery Service was last updated, in a ctime-like format

**logsEnabled** If true, logging is enabled for this Delivery Service, otherwise it is disabled

**longDesc** A description of the Delivery Service

**longDesc1** A field used when more detailed information that that provided by longDesc is desired

**longDesc2** A field used when even more detailed information that that provided by either longDesc or longDesc1 is desired

**matchList** An array of methods used by Traffic Router to determine whether or not a request can be serviced by this Delivery Service

- **pattern** A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)
- **setNumber** An integral, unique identifier for the set of types to which the type field belongs
- **type** The type of match performed using pattern to determine whether or not to use this Delivery Service

**HOST_REGEXP** Use the Delivery Service if pattern matches the Host: HTTP header of an HTTP request

**HEADER_REGEXP** Use the Delivery Service if pattern matches an HTTP header (both the name and value) in an HTTP request

**PATH_REGEXP** Use the Delivery Service if pattern matches the request path of this Delivery Service’s URL

**STEERING_REGEXP** Use the Delivery Service if pattern matches the xml_id of one of this Delivery Service’s “Steering” target Delivery Services

**maxDnsAnswers** The maximum number of IPs to put in responses to A/AAAA DNS record requests (0 means all available)

---

2 This only applies to HTTP-routed Delivery Services
midHeaderRewrite  Rewrite operations to be performed on TCP headers at the Edge-tier cache level - used by the Header Rewrite Apache Trafficserver plugin

missLat  The latitude to use when the client cannot be found in the CZF or a geographic IP lookup

missLong  The longitude to use when the client cannot be found in the CZF or a geographic IP lookup

multiSiteOrigin  true if the Multi Site Origin feature is enabled for this Delivery Service, false otherwise

orgServerFqdn  The URL of the Delivery Service’s origin server for use in retrieving content from the origin server

Note:  Despite the field name, this must truly be a full URL - including the protocol (e.g. http:// or https://) - NOT merely the server’s Fully Qualified Domain Name (FQDN)

originShield  An “origin shield” is a forward proxy that sits between Mid-tier caches and the origin and performs further caching beyond what’s offered by a standard CDN. This field is a string of FQDNs to use as origin shields, delimited by |

profileDescription  The description of the Traffic Router Profile with which this Delivery Service is associated

profileId  The integral, unique identifier for the Traffic Router profile with which this Delivery Service is associated

profileName  The name of the Traffic Router Profile with which this Delivery Service is associated

protocol  The protocol which clients will use to communicate with Edge-tier cache servers - this is an integer on the interval [0-2] where the values have these meanings:

0  HTTP
1  HTTPS
2  Both HTTP and HTTPS

qstringIgnore  Tells caches whether or not to consider URLs with different query parameter strings to be distinct - this is an integer on the interval [0-2] where the values have these meanings:

0  URLs with different query parameter strings will be considered distinct for caching purposes, and query strings will be passed upstream to the origin

3  See Use Multi-Site Origin Feature
URLs with different query parameter strings will be considered identical for caching purposes, and query strings will be passed upstream to the origin.

Query strings are stripped out by Edge-tier caches, and thus are neither taken into consideration for caching purposes, nor passed upstream in requests to the origin.

**rangeRequestHandling** Tells caches how to handle range requests - this is an integer on the interval [0,2] where the values have these meanings:

0  Range requests will not be cached, but range requests that request ranges of content already cached will be served from the cache

1  Use the `background_fetch` plugin to service the range request while caching the whole object

2  Use the experimental `cache_range_requests` plugin to treat unique ranges as unique objects

**regexRemap** A regular expression remap rule to apply to this Delivery Service at the Edge tier

**See also:**

The Apache Trafficserver documentation for the Regex Remap plugin

**regionalGeoBlocking** `true` if Regional Geo Blocking is in use within this Delivery Service, `false` otherwise - see Configure Regional Geo-blocking (RGB) for more information

**remapText** Additional, raw text to add to the remap line for caches

**See also:**

The Apache Trafficserver documentation for the Regex Remap plugin

**signed** `true` if token-based authentication is enabled for this Delivery Service, `false` otherwise

**signingAlgorithm** Type of URL signing method to sign the URLs, basically comes down to one of two plugins or `null`:

**null** Token-based authentication is not enabled for this Delivery Service

**url_sig**: URL Signing token-based authentication is enabled for this Delivery Service

**uri_signing** URI Signing token-based authentication is enabled for this Delivery Service

**See also:**

The Apache Trafficserver documentation for the `url_sig` plugin and the draft RFC for `uri_signing` - note, however that the current implementation of `uri_signing` uses Draft 12 of that RFC document, NOT the latest.
sslKeyVersion This integer indicates the generation of keys in use by the Delivery Service - if any - and is incremented by the Traffic Portal client whenever new keys are generated.

**Warning:** This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

tenantId The integral, unique identifier of the tenant who owns this Delivery Service.

trRequestHeaders If defined, this takes the form of a string of HTTP headers to be included in Traffic Router access logs for requests - it's a template where \_\_RETURN\_\_ translates to a carriage return and line feed (\r\n).

trResponseHeaders If defined, this takes the form of a string of HTTP headers to be included in Traffic Router responses - it's a template where \_\_RETURN\_\_ translates to a carriage return and line feed (\r\n).

type The name of the routing type of this Delivery Service e.g. “HTTP”.

typeId The integral, unique identifier of the routing type of this Delivery Service.

xmlId A unique string that describes this Delivery Service - exists for legacy reasons.

# 495: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 19:29:06 GMT
Content-Length: 1194

{
    "response": [
        {
            "active": true,
            "anonymousBlockingEnabled": false,
            "cacheurl": null,
            "ccrDnsTtl": null,
            "cdnId": 2,
            "cdnName": "CDN-in-a-Box",
            "checkPath": null,
            "displayName": "Demo 1",
```
"dnsBypassCname": null,
"dnsBypassIp": null,
"dnsBypassIp6": null,
"dnsBypassTtl": null,
"dscp": 0,
"edgeHeaderRewrite": null,
"geoLimit": 0,
"geoLimitCountries": null,
"geoLimitRedirectURL": null,
"geoProvider": 0,
"globalMaxMbps": null,
"globalMaxTps": null,
"httpBypassFqdn": null,
"id": 1,
"infoUrl": null,
"initialDispersion": 1,
"ipv6RoutingEnabled": true,
"lastUpdated": "2018-12-12 16:26:44+00",
"logsEnabled": true,
"longDesc": "Apachecon North America 2018",
"longDesc1": null,
"longDesc2": null,
"matchList": null,
"maxDnsAnswers": null,
"midHeaderRewrite": null,
"missLat": 42,
"missLong": -88,
"multiSiteOrigin": false,
"originShield": null,
"orgServerFqdn": "http://origin.infra.ciab.test",
"profileDescription": null,
"profileId": null,
"profileName": null,
"protocol": 0,
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regexRemap": null,
"regionalGeoBlocking": false,
"remapText": null,
"routingName": "video",
"signed": false,
"sslKeyVersion": null,
"tenantId": 1,
"type": "HTTP",
"typeId": 1,
"xmlId": "demo1",
"exampleURLs": null,
"deepCachingType": "NEVER",
"signingAlgorithm": null,
users/register

POST

Register a user and send registration email.

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type undefined

Request Structure

email Email address of the new user

Changed in version 1.4: Prior to version 1.4, the email was validated using the Email::Valid Perl package but is now validated (circuitously) by GitHub user asaskevich’s regular expression. Note that neither method can actually distinguish a valid, deliverable, email address but merely ensure the email is in a commonly-found format.

role The integral, unique identifier of the highest permissions role which will be afforded to the new user

tenantId An optional field containing the integral, unique identifier of the tenant to which the new user will belong

Note: This field is optional if and only if tenancy is not enabled in Traffic Ops.

# 496: Request Example

```
POST /api/1.3/users/register HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 59
Content-Type: application/json

{
    "email": "test@example.com",
```
"role": 3,
"tenantId": 1
}

Response Structure

# 497: Response Example

```http
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: application/json
Date: Thu, 13 Dec 2018 22:03:22 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 14 Dec 2018 02:03:22 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: yvf++Oqxvu3uOIAybWLUGJKxZ4T60Mi5H9eGTxrKLxnRhsHw0PdIrThbTnWtATBkak4yU/dPHLLXKW85LUTEWg==
Content-Length: 160

{
    "alerts": [
        {
            "level": "success",
            "text": "Sent user registration to test@example.com with the following permissions [ role: read-only | tenant: root ]"
        }
    ]
}
```

5.1.2 How to Read this Documentation

Each endpoint is on its own page, titled with the request path. The request paths shown on each endpoint’s page are - unless otherwise noted - only usable by being appended to the request path prefix `api/<version>/` where `<version>` is the API version being requested. The API versions officially supported as of the time of this writing are 1.1, 1.2, 1.3, 1.4. All endpoints are documented as though they were being used in version 1.4. If an endpoint or request method of an endpoint is only available after a specific version, that will be noted next to the method or endpoint name. If changes were made to the structure of an endpoint’s input or output, the version number and nature of the change will be noted.

Every endpoint is documented with a section for each method, containing the subsections “Re-
quest Structure” and “Response Structure” which identify all properties and structure of the Request to and Response from the endpoint. Before these subsections, three key pieces of information will be provided:

**Auth. Required** This will either be ‘Yes’ to indicate that a user must be authenticated (or “logged-in”) via e.g. *user/login* to use this method of the endpoint, or ‘No’ to indicate that this is not required.

**Roles Required** Any permissions roles that are allowed to use this method of the endpoint will be listed here. Users with roles not listed here will be unable to properly use these endpoints.

**Response Type** Unless otherwise noted, all responses are JSON objects. See *Response Structure* for more information.

The methods of endpoints that require/accept data payloads - unless otherwise noted - always interpret the content of the payload as a JSON object, regardless of the request’s Content-Type header. Because of this, all payloads are - unless otherwise noted - JSON objects. The Request Structure and Response Structure subsections will contain explanations of the fields before any examples like e.g.

- **foo** A constant field that always contains “foo”
- **bar** An array of objects that each represent a “bar” object
  - **name** The bar’s name
  - **value** The bar’s value (an integer)

All fields are mandatory in a request payload, or always present in a response payload unless otherwise noted in the field description.

In most cases, JSON objects have been “pretty-printed” by inserting line breaks and indentation. This means that the Content-Length HTTP header does not, in general, accurately portray the length of the content displayed in Request Examples and Response Examples. Also, the Traffic Ops endpoints will ignore any content negotiation, meaning that the Content-Type header of a request is totally meaningless. A utility may choose to pass the data as e.g. *application/x-www-form-urlencoded* (cURL’s default Content-Type) when constructing a Request Example, but the example itself will most often show *application/json* in order for syntax highlighting to properly work.

**Response Structure**

Unless otherwise noted, all response payloads come as JSON objects.

```json
# 498: Response Structure
{
    "response": "<JSON object with main response>",
}
```

To make the documentation easier to read, only the `<JSON object with main response>` is documented, even though the response endpoints may return other top-level
objects (most commonly the "alerts" object). The field definitions listed in the Response Structure subsection of an endpoint method are the elements of this object. Sometimes the response object is a string, sometimes it’s an object that maps keys to values, sometimes it’s an array that contains many arbitrary objects, and sometimes it isn’t present at all. For ease of reading, the field lists delegate the distinction to be made by the Response Type field directly under the request method heading.

**Response Type Meanings**

**Array** The fields in the field list refer to the keys of the objects in the response array.

**Object** The fields in the field list refer to the keys of the response object.

**undefined** No response object is present in the response payload. Unless the format is otherwise noted, this means that there should be no field list in the “Response Structure” subsection.

### 5.1.3 Using API Endpoints

1. Authenticate with valid Traffic Control user account credentials (the same used by Traffic Portal).

2. Upon successful user authentication, note the Mojolicious cookie value in the response headers1.

   **Note:** Many tools have methods for doing this without manual intervention - a web browser for instance will automatically remember and properly handle cookies. Another common tool, cURL, has command line switches that will also accomplish this. Most high-level programming language libraries will implement a cookie-handling method as well.

3. Pass the Mojolicious cookie value, along with any subsequent calls to an authenticated API endpoint.

   **Note:** Many endpoints support a .json suffix. This should be avoided at all costs, because there’s no real consistency regarding when it may be used, and the output of API endpoints, in general, are not capable of representing POSIX-compliant files (as a ‘file extension’ might imply).

---

1 A cookie obtained by logging in through Traffic Portal can be used to access API endpoints under the Traffic Portal domain name - since it will proxy such requests back to Traffic Ops. This is not recommended in actual deployments, however, because it will involve an extra network connection which could be avoided by simply using the Traffic Ops domain itself.
Example Session

A user makes a request to the /api/1.1/asns endpoint.

```plaintext
GET /api/1.1/asns HTTP/1.1
Accept: application/json
Host: trafficops.infra.ciab.test
User-Agent: example
```

The response JSON indicates an authentication error.

```plaintext
HTTP/1.1 401 UNAUTHORIZED
Content-Length: 68
Content-Type: application/json
Date: Tue, 02 Oct 2018 13:12:30 GMT

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

To authenticate, the user sends a POST request containing their login information to the /api/1.1/user/login endpoint.

```plaintext
POST /api/1.1/user/login HTTP/1.1
User-Agent: example
Host: trafficops.infra.ciab.test
Accept: application/json
Content-Length: 32
Content-Type: application/x-www-form-urlencoded

Traffic Ops responds with a Mojolicious cookie to be used for future requests, and a message indicating the success or failure (in this case success) of the login operation.

```plaintext
HTTP/1.1 200 OK
Connection: keep-alive
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Set-Cookie: mojolicious=...; Path=/; Expires=Tue, 02 Oct 2018 19:03:18 GMT; HttpOnly
Content-Type: application/json
Date: Tue, 02 Oct 2018 12:53:32 GMT
Access-Control-Allow-Credentials: true
Content-Length: 81
X-Server-Name: traffic_ops_golang/

{
  "alerts": [
```

(continues on next page)
Using this cookie, the user can now access their original target - the `/api/1.1/asns` end-point...

```
GET /api/1.1/asns HTTP/1.1
Accept: application/json
Cookie: mojolicious=...;
Host: trafficops.infra.ciab.test
User-Agent: Example

... and the Traffic Ops server will now happily service this request.
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Encoding: gzip
Content-Length: 48
Content-Type: application/json
Date: Tue, 02 Oct 2018 12:55:57 GMT
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-SHA512: u+Q5X7z/DMTc/VzRGaFlJBa8btA8EC...dnA85HCYTm8vVwsQCvle+uC1nA==
X-Server-Name: traffic_ops_golang/

{ "response": {
    "asns": [
    {
        "lastUpdated": "2012-09-17 21:41:22",
        "id": 27,
        "asn": 7015,
        "cachegroup": "us-ma-woburn",
        "cachegroupId": 2
    },
    {
        "lastUpdated": "2012-09-17 21:41:22",
        "id": 28,
        "asn": 7016,
        "cachegroup": "us-pa-pittsburgh",
        "cachegroupId": 3
    }
    ]
}}
```

(continues on next page)
5.1.4 API Errors

If an API endpoint has something to say besides the actual response (usually an error message), it will add a top-level object to the response JSON with the key "alerts". This will be an array of objects that represent messages from the server, each with the following string fields:

- **level**  "success", "info", "warning" or "error" as appropriate
- **text**  The alert’s actual message

The most common errors returned by Traffic Ops are:

**401 Unauthorized**  When a “mojolicious” cookie is supplied that is invalid or expired, or the login credentials are incorrect the server responds with a 401 UNAUTHORIZED response code.

```plaintext
# 499: Example of a Response to a Login Request with Bad Credentials

HTTP/1.1 401 Unauthorized
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Whole-Content-Sha512: xRKu2Q7Yj07UA6A6SyxMNmcBpuBcW2/¬bzuK05eTz2y4V27rXfP/5bSkNpesomJbi0O+xSmiybDsH1cL3P+pzpg==
X-Server-Name: traffic_ops_golang/
Date: Tue, 02 Oct 2018 13:28:30 GMT
Content-Length: 69

{ "alerts": [  
  {  
    "text": "Invalid username or password.",  
    "level": "error"
  }
]}
```

**404 Not Found**  When the requested resource (path) doesn’t exist, Traffic Ops returns a 404 NOT FOUND response code.
# 500: Example Response to GET /not/an/api/path
HTTP/1.1 with Proper Cookies

HTTP/1.1 404 Not Found
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: text/html;charset=UTF-8
Date: Tue, 02 Oct 2018 13:58:56 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Tue, 02 Oct 2018 17:58:56 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Ff5hO8ZUNUMbwCW0mBuUlsvrSmm/-Giijq7O3uLlvLz6V0u66Gom4Jag6UqIbbDBbnP6AG7l1Szd74TT6NidA==
Transfer-Encoding: chunked

The content of this response will be the Legacy UI login page (which is omitted because it’s huge)

500 Internal Server Error When a server-side error occurs, the Perl API will return a 500 INTERNAL SERVER ERROR response (the below example request will result in a 400 BAD REQUEST response if using the v1.3 API instead - as this will use the Go server’s API)

# 501: Example Response to GET /api/1.1/servers/hostname/jj/details.json (‘jj’ doesn’t exist)

HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 93
Content-Type: application/json
Date: Tue, 02 Oct 2018 17:29:42 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=..; expires=Sun, 19 Apr 2015 00:45:06 GMT; path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: gFa4NYFmofCbV7Yqgw+yFrzKk90+KNgo2u6p2Nz98J4gy7/-2j55tYknvk53WXuMdkKrgYMop4uiYO1al1k1ozQQ==

{ "alerts": [ 

](continues on next page)
The rest of the API documentation will only document the `200 OK` case, where no errors have occurred.

### 5.1.5 TrafficOps Native Client Libraries

TrafficOps client libraries are available in Java, Go and Python. You can read (very little) more about them in the client README.
Tools

A living list of tools for testing, interacting with, and developing for the Traffic Control CDN

6.1 Tools

This is a living list of tools used to interact with, test, and develop for the Traffic Control CDN.

6.1.1 The Compare Tool

The compare tool is used to compare the output of a set of Traffic Ops API endpoints between two running instances of Traffic Ops. The idea is that two different versions of Traffic Ops with the same data will have differences in the output of their API endpoints if and only if either the change was intentional, or a new bug was introduced in the newer version. Typically, this isn’t really true, due to rapidly changing data structures like timestamps in the API outputs, but this should offer a good starting point for identifying bugs in changes made to the Traffic Ops API.

Location and Dependencies

The compare tool is written in Go, and can be found from within the Traffic Control repository at traffic_ops/testing/compare/. The main file of interest is compare.go, which contains the main routine and logic for checking endpoints. To build the executable, run go build . from within the traffic_ops/testing/compare/ directory. Alternatively, run the file without storing a built binary by using go run <path to compare.go>. In order to build/run the compare tool, the following dependencies should be satisfied, probably via go get:
The directory containing the compare tool also contains an executable Python 3 script named genConfigRoutes.py. This script can be used to scrape the two Traffic Ops instances for API routes that resolve to generated configuration files for mid-tier and edge-tier cache servers, which can then be fed directly into the compare tool via a file or pipe. While the script itself has no actual dependencies, it must be run from within the full Traffic Control repository, as it imports the Python client for Traffic Ops (located in traffic_control/clients/python/trafficops inside the repository). The client itself has its own documented dependencies.

**Usage**

```
traffic_ops/testing/compare/compare.go
```

```
```

`--ref_passwd PASSWD`

The password for logging into the reference Traffic Ops instance. This option overrides the TO_PASSWORD environment variable, and is required if and only if TO_PASSWORD is not set.

`--ref_url URL`

The URL that points to the reference Traffic Ops instance. This option overrides the TO_URL environment variable, and is required if and only if TO_URL is not set.

`--ref_user USER`

The username for logging into the reference Traffic Ops instance. This option overrides the TO_USER environment variable, and is required if and only if TO_USER is not set.

`--test_passwd PASSWD`

The password for logging into the testing Traffic Ops instance. This option overrides the TEST_PASSWORD environment variable. Additionally, if this option is not specified and TEST_PASSWORD is not set, the value for TO_PASSWORD (or --ref_passwd if overridden) will be used.

`--test_url URL`

The URL for the testing Traffic Ops instance. This option overrides the TEST_URL environment variable. Additionally, if this option is not specified and TEST_URL is not set, the value for TO_URL (or --ref_url if overridden) will be used.

`--test_user USER`

The username for logging into the testing Traffic Ops instance. This option overrides

---

1 Theoretically, if you downloaded the Traffic Control repository properly (into $GOPATH/src/github.com/apache/trafficcontrol), this will already be satisfied.
the TEST_USER environment variable. Additionally, if this option is not specified and TEST_USER is not set, the value for TO_USER (or --ref_user if overridden) will be used.

-f FILE, --file FILE
This optional flag specifies a file from which to list API paths to test. If this option is not given, compare will read from STDIN.

-h, --help
Print usage information and exit

-r PATH, --results_path PATH
This optional flag specifies an output directory where results will be written. Default: ./results

-V, --version
Print version information and exit

Changed in version 3.0.0: Removed the -s command line switch to compare CDN Snapshots - this is now the responsibility of the genConfigRoutes.py script.

traffic_ops/testing/compare/genConfigRoutes.py

Note: This script uses the Apache-TrafficControl Package, and so that must be installed to use it.


A simple script to generate API routes to server configuration files for a given pair of Traffic Ops instances. This, for the purpose of using the compare tool

-h, --help
Show usage information and exit

--refURL URL
The full URL of the reference Traffic Ops instance. This option overrides the TO_URL environment variable, and is required if and only if TO_URL is not set.

--testURL URL
The full URL of the testing Traffic Ops instance. This option overrides the TEST_URL environment variable. Additionally, if this option is not specified and TEST_URL is not set, the value for TO_URL (or --refURL if overridden) will be used.

--refUser USER
A username for logging into the reference Traffic Ops instance. This option overrides the TO_USER environment variable, and is required if and only if TO_USER is not set.

--refPasswd PASSWD
A password for logging into the reference Traffic Ops instance. This option overrides the
The `TO_PASSWORD` environment variable, and is required if and only if `TO_PASSWORD` is not set.

`--testUser USER`
A username for logging into the testing Traffic Ops instance. This option overrides the `TEST_USER` environment variable. Additionally, if this option is not specified and `TEST_USER` is not set, the value for `TO_USER` (or `--refUser` if overridden) will be used.

`--testPasswd PASSWD`
A password for logging into the testing Traffic Ops instance. This option overrides the `TEST_PASSWORD` environment variable. Additionally, if this option is not specified and `TEST_PASSWORD` is not set, the value for `TO_PASSWORD` (or `--refPasswd` if overridden) will be used.

`-k, --insecure`
Do not verify SSL certificate signatures against either Traffic Ops instance (default: False)

`-v, --version`
Print version information and exit

`-l LOG_LEVEL, --log_level LOG_LEVEL`
Sets the Python log level, one of
- DEBUG
- INFO
- WARN
- ERROR
- CRITICAL
(default: INFO)

..option:: -q, –quiet
Suppresses all logging output - even for critical errors (default: False)

`-s, --snapshot`
Produce CDN Snapshot routes in the output (CRConfig.json, snapshot/new etc.) (default: False)

`-C, --no-server-configs`
Do not generate routes for server configuration files (default: False)

**Tip:** If you're using a CDN-in-a-Box environment for testing, it's likely that you'll need the `-k/--insecure` option if you're outside the Docker network

**Environment Variables**

Both `compare` and `genConfigRoutes.py` require connection and authentication methods for two Traffic Ops instances. For ease of use, these can be provided by environment variables.
Both programs are capable of using the same environment variables, so that they only need to be defined once each.

**TO_URL**

The URL of the reference Traffic Ops instance. Overridden by `genConfigRoutes.py --refURL` and `compare --ref_url`.

**TO_USER**

The username to authenticate with the reference Traffic Ops instance. Overridden by `genConfigRoutes.py --refUser` and `compare --ref_user`.

**TO_PASSWORD**

The password to authenticate with the reference Traffic Ops instance. Overridden by `genConfigRoutes.py --refPasswd` and `compare --ref_passwd`.

**TEST_URL**

The URL of the testing Traffic Ops instance. Overridden by `genConfigRoutes.py --testURL` and `compare --test_url`.

**TEST_USER**

The username to authenticate with the testing Traffic Ops instance. Overridden by `genConfigRoutes.py --testUser` and `compare --test_user`.

**TEST_PASSWORD**

The password to authenticate with the testing Traffic Ops instance. Overridden by `genConfigRoutes.py --testPasswd` and `compare --test_passwd`.

### Usage in a Pipeline

The `genConfigRoutes.py` script will output list of unique API routes (relative to the desired Traffic Ops URL) that point to generated configuration files for a sample set of servers common to both Traffic Ops instances. The results are printed to STDOUT, making the output perfect for piping directly into `compare` like so:

```
# 502: Example Pipeline from genConfigRoutes.py into compare
./genConfigRoutes.py https://trafficopsA.example.test https://trafficopsB.example.test username:password | ./compare
```

**Note:** This is assuming the proper *Environment Variables* have been set for `compare`.

### Usage with Docker

A Dockerfile is provided to run tests on a pair of instances given the configuration environment variables necessary. This will generate configuration file routes using `genConfigRoutes.py`, and add them to whatever is already contained in `traffic_ops/testing/compare/testroutes.txt`, then run the `compare` tool on the final API route list. Build artifacts (i.e. 6.1. Tools 755
Apache Traffic Control Documentation, Release 3.0.0

anything output files created by the compare tool) are placed in the /artifacts/ directory
on the container. To retrieve these results, the use of a volume is recommended. The build
context must be at the root of the Traffic Control repository, as the tools have dependencies on
the Traffic Control clients.
Arguments can be passed to the genConfigRoutes.py script by defining the build-time
argument MODE. By default it expands to -s to allow the generation of CDN Snapshot routes.
It is not necessary to pass -k/--insecure, as the Dockerfile will do that implicitly.
In order to use the container, the following environment variables must be defined for the
container at runtime:
TO_URL The URL of the reference Traffic Ops instance
TO_USER The username to authenticate with the reference Traffic Ops instance
TO_PASSWORD The password to authenticate with the reference Traffic Ops instance
TEST_URL The URL of the testing Traffic Ops instance
TEST_USER The username to authenticate with the testing Traffic Ops instance
TEST_PASSWORD The password to authenticate with the testing Traffic Ops instance
# 503: Sample Script to Build and Run
sudo docker build . -f traffic_ops/testing/compare/Dockerfile -t
˓→compare:latest
sudo docker run -v $PWD/artifacts:/artifacts -e TO_URL="$TO_URL" -e
˓→TEST_URL="$TEST_URL" -e TO_USER="admin" -e TO_PASSWORD="twelve" ˓→e TEST_USER="admin" -e TEST_PASSWORD="twelve" compare:latest

Note:
The above code example assumes that the environment variables TO_URL and
TEST_URL refer to the URL of the reference Traffic Ops instance and the URL of the test
Traffic Ops instance, respectively (including port numbers). It also uses credentials suitable for
logging into a stock CDN in a Box instance.

Note: Unlike using the genConfigRoutes.py script and/or the compare on their own,
all of the variables must be defined, even if they are duplicates.

6.1.2 Apache-TrafficControl Package
Package Contents
class trafficops.TOSession(host_ip, host_port=443, api_version=’1.3’,
ssl=True, headers=None, verify_cert=True)
Bases: trafficops.restapi.RestApiSession

756

Chapter 6. Tools


Traffic Ops Session Class Once you login to the Traffic Ops API via `login()`, you can call one or more of the methods to retrieve, POST, PUT, DELETE, etc. data to the API. If you are not logged in, an exception will be thrown if you try to call any of the endpoint methods. This API client is simplistic and lightly structured on purpose but adding support for new endpoints routinely takes seconds. Another nice bit of convenience that result data is, by default, wrapped in `munch.Munch` objects, which provide attribute access to the returned dictionaries/hashe - e.g. `a_dict['a_key']` with `munch` becomes `a_dict.a_key` or `a_dict['a_key']`. Also, the lack of rigid structure (loose coupling) means many changes to the Traffic Ops API, as it evolves, will probably go un-noticed (usually additions), which means fewer future problems to potentially fix in user applications.

An area of improvement for later is defining classes to represent request data instead of loading up dictionaries for request data.

Please see the *API documentation* for the details of the API endpoints.

Adding end-point methods

# 504: Endpoint with no URL parameters and no query parameters

```python
@api_request(u'get', u'cdns', (u'1.1', u'1.2',))
def get_cdns(self):
    pass
```

# 505: End-point with URL parameters and no query parameters

```python
@api_request(u'get', u'cdns/{cdn_id:d}', (u'1.1', u'1.2',))
def get_cdn_by_id(self, cdn_id=None):
    pass
```

# 506: End-point with no URL parameters but with query parameters

```python
@api_request(u'get', u'deliveryservices', (u'1.1', u'1.2',))
def get_deliveryservices(self, query_params=None):
    pass
```

# 507: End-point with URL parameters and query parameters

```python
@api_request(u'get', u'deliveryservices/xmlId/{xml_id}/sslkeys',
             (u'1.1', u'1.2',))
def get_deliveryservice_ssl_keys_by_xml_id(self, xml_id=None, 
                                         query_params=None):
    pass
```
# 508: End-point with request data

```python
@api_request('post', 'cdns', (u'1.1', u'1.2',))
def create_cdn(self, data=None):
    pass
```

# 509: End-point with URL parameters and request data

```python
@api_request('put', 'cdns', (u'1.1', u'1.2',))
def update_cdn_by_id(self, cdn_id=None, data=None):
    pass
```

Calling end-point methods

**get_cdns()** calls endpoint cdns e.g. `t.get_cdns()`  
**get_types()** calls endpoint types, optionally with query parameters e.g. `get_foo_data(id=45, query_params={'sort': 'asc'})` calls endpoint `GET api/1.x/foo/45?sort=asc` (presumably)  
**cdns_queue_update()** calls endpoint `cdns/[[ID]]/queue_update`, with an ID path parameter and a JSON payload e.g. `cdns_queue_update(id=1, data={'action': 'queue'})`

**Note:** Only a small subset of the API endpoints are implemented. More can be implemented as needed.

**add_ssl_keys_to_deliveryservice** *(data=None)*  
Add SSL keys to a Delivery Service. deliveryservices/sslkeys/add  
:param data: The parameter data to use for adding SSL keys to a Delivery Service.  
:type data: Dict[str, Any]  
:type: Tuple[Dict[str, Any], requests.Response]  
:raises: Union[LoginError, OperationError]

**assign_delivery_services_to_federations** *(federation_id=None, data=None)*  
Create one or more federation / delivery service assignments. federations/[[ID]]/deliveryservices  
:param federation_id: The federation id  
:type federation_id: int  
:param data: The update action. QueueUpdateRequest() can be used for this argument also.  
:type data: Dict[str, Any]  
:type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

**assign_deliveryservice_servers_by_ids** *(data=None)*  
Assign servers by id to a Delivery Service. (New Method) deliveryserviceserver  
:param data: The required data to create server associations to a delivery service  
:type data: Dict[str, Any]  
:type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

**assign_deliveryservice_servers_by_names** *(xml_id=None, data=None)*
Assign servers by name to a Delivery Service by xmlId. 

```python
def assign_servers_by_name(xml_id, data):
    # Assign servers by name to a Delivery Service by xmlId.
```

**assign_federation_resolver_to_federations**

```python
assign_federation_resolver_to_federations(federation_id=None, data=None)
```

Create one or more federation / federation resolver assignments.

```python
federations/{{ID}}/federation_resolvers
```

**assign_parameter_to_profile_ids**

```python
assign_parameter_to_profile_ids(data=None)
```

Create one or more parameter / profile assignments.

```python
profileparameter
```

**assign_profile_to_parameter_ids**

```python
assign_profile_to_parameter_ids(data=None)
```

Create one or more profile / parameter assignments.

```python
profileparameter
```

**associate_parameter_to_profile**

```python
associate_parameter_to_profile(data=None)
```

Associate parameter to profile.

```python
profileparameters
```

**associate_parameters_by_profile_id**

```python
associate_parameters_by_profile_id(profile_id=None, data=None)
```

Associate Parameters to a Profile by Id.

```python
profiles/{{ID}}/parameters
```

**associate_parameters_by_profile_name**

```python
associate_parameters_by_profile_name(profile_name=None, data=None)
```

Associate Parameters to a Profile by Name.

```python
profiles/name/{{name}}/parameters
```

**base_url**

Returns the base url. (read-only)

```python
Returns
```

```
The base url should match ‘^[w+.:]+://[w+.:]+(:d+)?$’ e.g

https://to.somedomain.net/api/1.2/
```
Return type  str

cachegroups_queue_update (cache_group_id=None, data=None)
Queue Updates by Cache Group ID cachegroups/{{ID}}/queue_update :param
  cache_group_id: The Cache Group Id :type cache_group_id: int :param data:
The update action. QueueUpdateRequest() can be used for this argument also.
  Union[LoginError, OperationError]

cdns_queue_update (cdn_id=None, data=None)
Queue Updates by CDN Id. cdns/{{ID}}/queue_update :param cdn_id: The CDN
  Id :type cdn_id: int :param data: The update action. QueueUpdateRequest() can
  be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Dict[str,
  Any], requests.Response] :raises: Union[LoginError, OperationError]

copy_profile (new_profile_name=None, copy_profile_name=None,
  data=None)
Copy profile to a new profile. The new profile name must not exist
  profiles/name/{{name}}/copy/{{copy}} :param new_profile_name: The name of pro-
  file to copy to :type new_profile_name: String :param copy_profile_name: The
  name of profile copy from :type copy_profile_name: String :param data: The up-
  date action. QueueUpdateRequest() can be used for this argument also. :type
  Union[LoginError, OperationError]

create_asn (data=None)
Create ASN asns :param data: The parameter data to use for cachegroup creation.
  Union[LoginError, OperationError]

create_cache_group_fallbacks (data=None)
Creates fallback configuration for the cache group. New fallbacks can be added
  only via POST. cachegroup_fallbacks :param data: The update action. Queue-
  UpdateRequest() can be used for this argument also. :type data: Dict[str, Any]
  :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, Opera-
  tionError]

create_cachegroups (data=None)
Create a Cache Group cachegroups :param data: The parameter data to use for
  cachegroup creation. :type data: Dict[str, Any] :rtype: Tuple[Dict[str, Any],

create_cdn (data=None)
Create a new CDN. cdns :param data: The parameter data to use for cdn creation.
  Union[LoginError, OperationError]

create_cdn_dns_sec_keys (data=None)
Generates ZSK and KSK keypairs for a CDN and all associated Delivery Services
deliveryservices/dnssseckeys/generate :param data: The parameter data to use for
  cachegroup creation. :type data: Dict[str, Any] :rtype: Tuple[Dict[str, Any], re-
create_coordinates(data=None)

create_delivery_service_user_link(data=None)
Create one or more user / delivery service assignments. deliveryservice_user :param data: The parameter data to use for Delivery Service SSL key generation. :type data: Dict[str, Any] :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

create_deliveryservice(data=None)

create_deliveryservice_regexes(delivery_service_id=None, data=None)

create_division(data=None)
Create a division divisions :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_federation(data=None)
Allows a user to add federations for their delivery service(s). federations :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_federation_in_cdn(cdn_name=None, data=None)

create_federation_resolver(data=None)
Create a federation resolver. :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_federation_user(federation_id=None, data=None)
Create one or more federation / user assignments. federations/{{ID}}/users :param

create_invalidation_job (data=None)
Invalidate content on the CDN is sometimes necessary when the origin was misconfigured and something is cached in the CDN 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 inaccessible using the regex_revalidate ATS plugin. This forces a revalidation of the content, rather than a new get.

create_origins (data=None)
Creates origins associated with a delivery service origins :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_parameter (data=None)

create_physical_location (region_name=None, query_params=None)
Create physical location regions/:region_name/phys_locations :param region_name: the name of the region to create physical location into :type region_name: String :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_profile (data=None)
Create a profile profiles :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_region (division_name=None, data=None)
Create a region divisions/{{name}}/regions :param division_name: The Division name in which region will reside :type division_name: String :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_server (data=None)
create_servercheck (data=None)

create_staticdnsentries (data=None)
Create static DNS entries associated with the delivery service staticdnsentries :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_tenant (data=None)
Create a tenant tenants :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_to_extension (data=None)

create_user (data=None)
Create a user. users :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

create_user_with_registration (data=None)
Register a user and send registration email users/register :param data: The update action. QueueUpdateRequest() can be used for this argument also. :type data: Dict[str, Any] :type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

delete_asn (asn_id=None)

delete_cache_group_fallbacks (query_params=None)
Deletes an existing fallback related configurations for a cache group cachegroup_fallbacks

Parameters query_params (Dict[str, int]) – Either cacheGroupId or fallbackId must be used or can be used simultaneously

Return type Tuple[Dict[str, Any], requests.Response]

6.1. Tools 763
**Raises** Union[LoginError, OperationError]

```python
delete_cache_group_parameters(cache_group_id=None, parameter_id=None)
delete a cache group parameter association cachegroupparameters/{{ID}}/[[parameterID]]
:param cache_group_id: The cache group id in which the parameter will be deleted
:type cache_group_id: int
:param parameter_id: The parameter id which will be disassociated
:type parameter_id: int
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_cachegroups(cache_group_id=None)
delete a cache group cachegroups/{{ID}}
:param cache_group_id: The cache group id to update
:type cache_group_id: Integer
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_cdn_by_id(cdn_id=None)
delete a CDN by Id. cdns/{{ID}}
:param cdn_id: The CDN id
:type cdn_id: int
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_cdn_dns_sec_keys(cdn_name=None)
delete dnssec keys for a cdn and all associated delivery services cdns/name/{{name}}/dnsseckeys/delete
:param cdn_name: The CDN name to delete dnsseckeys info for
:type cdn_name: String
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_coordinates(query_params=None)
delete coordinates coordinates
:param query_params: The optional url query parameters for the call
:type query_params: Dict[str, Any]
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_delivery_service_user_link(delivery_service_id=None, user_id=None)
deletes a delivery service from a user.
deliveryservice_user/{{dsID}}/[[userID]]
:param delivery_service_id: The delivery service id to disassociate the user
:type delivery_service_id: int
:param user_id: The user id to disassociate
:type user_id: int
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_deliveryservice_by_id(delivery_service_id=None)
Allows user to delete a delivery service. deliveryservices/{{ID}}
:param delivery_service_id: The delivery service Id
:type delivery_service_id: int
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]
```

```python
delete_deliveryservice_regex_by_regex_id(delivery_service_id=None, delivery_service_regex_id=None)
delete a RegEx by Id for a Delivery Service by Id.
deliveryservices/{{ID}}/regexes/{{rID}}
:param delivery_service_id: The delivery service Id
:type delivery_service_id: int
:param delivery_service_regex_id: The delivery ser-

 delete_deliveryservice_servers_by_id(delivery_service_id=None, server_id=None)

Removes a server (cache) from a delivery service.  

 delete_deliveryservice_ssl_keys_by_xml_id(xml_id=None, query_params=None)


 delete_division(dvision_id=None, query_params=None)


 delete_federation_in_cdn(cdn_name=None, federation_id=None)


 delete_federation_resolver(federation_resolver_id=None)


 delete_federation_user(federation_id=None, user_id=None)


 delete_origins(query_params=None)

```python
def delete_parameter(parameter_id=None):
    Delete Parameter `parameters/{{ID}}`
    :param parameter_id: The parameter id to delete
    :type parameter_id: int
    :rtype: Tuple[Dict[str, Any], requests.Response]
    :raises: Union[LoginError, OperationError]

def delete_physical_location(physical_location_id=None, query_params=None):
    Delete Physical Location by id `phys_locations/{{ID}}`
    :param physical_location_id: The id to delete
    :type physical_location_id: int
    :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    :raises: Union[LoginError, OperationError]

def delete_profile_by_id(profile_id=None):
    Delete Profile by Id. `profiles/{{ID}}`
    :param profile_id: The profile Id
    :type profile_id: int
    :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    :raises: Union[LoginError, OperationError]

def delete_profile_parameter_association_by_id(profile_id=None, parameter_id=None):
    Delete Parameter association by Id for a Profile by Id. `profileparameters/{{profileID}}/{{parameterID}}`
    :param profile_id: The profile id
    :type profile_id: int
    :param parameter_id: The parameter id
    :type parameter_id: int
    :rtype: Tuple[Dict[str, Any], requests.Response]
    :raises: Union[LoginError, OperationError]

def delete_server_by_id(server_id=None):
    Delete a Server by Id. `servers/{{ID}}`
    :param server_id: The server Id
    :type server_id: int
    :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    :raises: Union[LoginError, OperationError]

def delete_staticdnsentries(query_params=None):
    Delete static DNS entries associated with the delivery service `staticdnsentries`
    :param query_params: The optional url query parameters for the call
    :type query_params: Dict[str, Any]
    :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    :raises: Union[LoginError, OperationError]

def delete_to_extension(extension_id=None):
    Deletes a Traffic Ops extension. `to_extensions/{{ID}}/delete`
    :param extension_id: The extension id to delete
    :type extension_id: int
    :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    :raises: Union[LoginError, OperationError]

def generate_deliveryservice_ssl_keys(data=None):
    Generate an SSL certificate. (self-signed) `deliveryservices/sslkeys/generate`
    :param data: The parameter data to use for Delivery Service SSL key generation
    :type data: Dict[str, Any]
    :rtype: Tuple[Dict[str, Any], requests.Response]
    :raises: Union[LoginError, OperationError]

def generate_deliveryservice_url_signature_keys(xml_id=None):
    Generate URL Signature Keys for a Delivery Service by xmlId. `deliveryservices/xmlId/{{xml_id}}/urlkeys/generate`
    :param xml_id: The Delivery Service xml-

generate_iso (data=None)

get_all_cachegroup_parameters ()

get_all_deliveryservice_servers (*args, **kwargs)

get_api_capabilities (query_params=None)

get_api_capabilities_by_id (id=None)

get_asn_by_id (asn_id=None)

get_asns (query_params=None)

get_associated_profiles_by_parameter_id (parameter_id=None)

get_authenticated_user ()

get_authenticated_user_jobs ()

get_cache_group_fallbacks (query_params=None)
Retrieve fallback related configurations for a cache group `cachegroup_fallbacks`

**Parameters**

`query_params (Dict[str, int])` – Either `cacheGroupId` or `fallbackId` must be used or can be used simultaneously

**Return type**

`Tuple[Dict[str, Any], requests.Response]`

**Raises**

`Union[LoginError, OperationError]`

get_cache_stats(`query_params=None`)


get_cachegroup_by_id(`cache_group_id=None`)


get_cachegroup_parameters(`cache_group_id=None`)


get_cachegroup_parameters_by_id(`parameter_id=None`)


get_cachegroup_unassigned_parameters(`cache_group_id=None`)


get_cachegroups(`query_params=None`)


get_cdn_by_id(`cdn_id=None`)


get_cdn_by_name(`cdn_name=None`)


get_cdn_dns_sec_keys(`cdn_name=None`)

Gets a list of dnsseckeys for a CDN and all associated Delivery Services `cdns/name/{{name}}/dnsseckeys` :param `cdn_name`: The CDN name to find

Chapter 6. Tools
get_cdn_health_by_name(cdn_name=None)
Retrieves the health of all locations (cache groups) for a given CDN

get_cdn_monitoring_info(cdn_name=None)
Retrieves CDN monitoring information

get_cdn_routing_info(cdn_name=None)
Retrieves CDN routing information

get_cdn_specific_config_file(cdn_name=None, config_file=None, query_params=None)
Get the configuration files for a given cdn name and config file

get_cdn_ssl_keys(cdn_name=None)
Returns ssl certificates for all Delivery Services that are a part of the CDN.

get_cdns()
Get all CDNs.

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

get_cdns_domains()
Retrieves the different CDN domains

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

```
```

`get_cdns_usage()`
Retrieves the high-level CDN usage metrics.  

```
```

`get_change_logs()`

`get_change_logs_for_days (days=None)`
Retrieve all change logs from Traffic Ops logs/{{days}}/days :param days: The number of days to retrieve change logs :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

`get_change_logs_newcount ()`

`get_coordinates (query_params=None)`

`get_current_snapshot_crconfig (cdn_name=None)`
Retrieves the CURRENT snapshot for a CDN which doesn’t necessarily represent
the current state of the CDN. The contents of this snapshot are currently
used by Traffic Monitor and Traffic Router.  

```
```

`get_delivery_service_capacity (delivery_service_id=None)`
Retrieves the capacity percentages of a delivery service. Delivery service
must be assigned to user if user is not admin or operations.  

```
```

`get_delivery_service_failover_state (delivery_service_id=None)`
Retrieves the failover state for a delivery service. Delivery service must be
assigned to user if user is not admin or operations.  

```
```

`get_delivery_service_health (delivery_service_id=None)`
Retrieves the health of all locations (cache groups) for a delivery service. Delivery
service must be assigned to user if user is not admin or operations.  

```
```
get_delivery_service_routing (delivery_service_id=None)
Retrieves the routing method percentages of a delivery service. Delivery service must be assigned to user if user is not admin or operations. deliveryservices/{{ID}}/routing :param delivery_service_id: The delivery service Id :type delivery_service_id: int :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

get_delivery_service_stats (query_params=None)

get_deliveryservice_by_id (delivery_service_id=None)

get_deliveryservice_ineligible_servers (delivery_service_id=None)

get_deliveryservice_regexes_by_id (delivery_service_id=None)

get_deliveryservice_regexes_by_regex_id (delivery_service_id=None, regex_id=None)

get_deliveryservice_servers (delivery_service_id=None)

get_deliveryservice_ssl_keys_by_xml_id (xml_id=None, query_params=None)
Get SSL keys for a Delivery Service by xmlId. deliveryservices/xmlId/{{XMLID}}/sslkeys :param xml_id: The Delivery Service XML id
get_deliveryservice_unassigned_servers(delivery_service_id=None)
Retrieves properties of CDN EDGE or ORG servers not assigned to a delivery service. (Currently call does not work) deliveryservices/{{ID}}/unassigned_servers

get_deliveryservices(query_params=None)
Retrieves all delivery services (if admin or ops) or all delivery services assigned to user. deliveryservices :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

get_deliveryservices_regexes()

get_deliveryserviceserver(query_params=None)

get_division_by_id(dvision_id=None)

get_divisions()

get_federation_delivery_services(federation_id=None)

get_federation_for_cdn_by_id(cdn_name=None, federation_id=None)

get_federation_resolvers(query_params=None)
Get federation resolvers. federation_resolvers :rtype: Tuple[Union[Dict[str,
get_federation_resolvers_by_id(federation_id=None)

Retrieves federation resolvers assigned to a federation.

:param federation_id: The federation id
:type federation_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_federation_users(federation_id=None)

Retrieves users assigned to a federation.

:param federation_id: The federation id
:type federation_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_federations()

Retrieves a list of federation mappings (aka federation resolvers) for the current user.

:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_federations_for_cdn(cdn_name=None)

Retrieves a list of federations for a CDN.

:param cdn_name: The CDN name to find federation
:type cdn_name: String
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_hwinfo()

Get hwinfo for servers.

:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_job_by_id(job_id=None)

Get a job by ID (currently limited to invalidate content (PURGE) jobs).

:param job_id: The job id to retrieve
:type job_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_jobs(query_params=None)

Get all jobs (currently limited to invalidate content (PURGE) jobs) sorted by start time (descending).

:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_origins(query_params=None)

Get origins associated with the delivery service.

:param query_params: The optional url query parameters for the call
:type query_params: Dict[str, Any]
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_osversions()

Get all OS versions for ISO generation and the directory where the kickstarter files are found. The values are retrieved from osversions.cfg found in either /var/www/files or in the location defined by the kickstart.files.location parameter (if defined).

:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]
Apache Traffic Control Documentation, Release 3.0.0

get_parameter_by_id (parameter_id=None)
Get a Parameter by Id.  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

get_parameters ()
Get all Profile Parameters.  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

get_parameters_by_profile_id (profile_id=None)
Get all Parameters associated with a Profile by Id.  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

get_parameters_by_profile_name (profile_name=None)
Get all Parameters associated with a Profile by Name.  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

get_pending_snapshot_crconfig (cdn_name=None)
Retrieves a PENDING snapshot for a CDN which represents the current state of the CDN. The contents of this snapshot are NOT currently used by Traffic Monitor and Traffic Router. Once a snapshot is performed, this snapshot will become the CURRENT snapshot and will be used by Traffic Monitor and Traffic Router.  
:returns: Tuple[Dict[str, Any], requests.Response]  
:raises: Union[LoginError, OperationError]

get_physical_location_by_id (physical_location_id=None)
Get Physical Location by id  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

get_physical_locations (query_params=None)
Get Physical Locations.  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]

get_profile_by_id (profile_id=None)
Get Profile by Id.  
:returns: Tuple[Dict[str, Any], requests.Response]  
:raises: Union[LoginError, OperationError]

get_profile_specific_config_files (profile_name=None, config_file=None, query_params=None)
Get the configuration files for a given profile name and config file  
:returns: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]  
:raises: Union[LoginError, OperationError]
get_profiles(query_params=None)

get_region_by_id(region_id=None)

get_regions()

get_roles()

get_server_by_id(server_id=None)

get_server_config_files(host_name=None, query_params=None)
    Get the configuration files for a given host name servers/{{server}}/configfiles/ats :param host_name: The host name to get config files for :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_server_delivery_services(server_id=None)

get_server_details(name=None)

get_server_specific_config_file(host_name=None, config_file=None, query_params=None)
    Get the configuration files for a given host name and config file servers/{{server}}/configfiles/ats/{{filename}} :param host_name: The host name to get config files for :param config_file: The configuration file name to retrieve for host :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_server_status_count()
Retrieves a count of CDN servers by status `servers/status`

```python
def get_server_type_count()
    Retrieves a count of CDN servers by type `servers/totals`
```

```python
def get_server_update_status(server_name=None)
    Gets the current update status of a server named `server_name`
```

```python
def get_servers(query_params=None)
    Get Servers.
```

```python
def get_static_dns_entries()
    Get Static DNS Entries.
```

```python
def get_staticdnsentries(query_params=None)
    Get static DNS entries associated with the delivery service `staticdnsentries`
```

```python
def get_statuses()
    Retrieves a list of the server status codes available.
```

```python
def get_statuses_by_id(status_id=None)
    Retrieves a server status by ID.
```

```python
def get_system_info()
    Get information on the traffic ops system.
```

```python
def get_tenant_by_id(tenant_id=None)
    Get a tenant by ID.
```

```python
def get_tenants()
    ```
Get all tenants. 

```python
get_tenant():
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_to_extensions()**
Retrieves the list of extensions.

```python
to_extensions():
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_traffic_monitor_cache_stats()**
Retrieves cache stats from Traffic Monitor. Also includes rows for aggregates.

```python
get_traffic_monitor_cache_stats():
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_truncated_physical_locations()**
Get Physical Locations with name only.

```python
get_truncated_physical_locations():
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_truncated_profiles()**
Get Profiles with names only.

```python
get_truncated_profiles():
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_type_by_id(type_id=None)**
Get Data Type with the given type id.

```python
get_type_by_id(type_id=None):
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_types(query_params=None)**
Get Data Types.

```python
get_types(query_params=None):
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_types_only_names()**
Get Data Types with only the Names.

```python
get_types_only_names():
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_unassigned_profiles_by_parameter_id(parameter_id=None)**
Retrieves all profiles NOT assigned to the parameter.

```python
get_unassigned_profiles_by_parameter_id(parameter_id=None):
    rtype: Tuple[Dict[str, Any], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_unassigned_parameters_by_profile_id(profile_id=None)**
Get all Parameters associated with a Profile by Id.

```python
get_unassigned_parameters_by_profile_id(profile_id=None):
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```

**get_user_by_id(user_id=None)**
Retrieves user by ID.

```python
get_user_by_id(user_id=None):
    rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
    raises: Union[LoginError, OperationError]
```
get_user_delivery_services (user_id=None)
Retrieves all delivery services assigned to the user.

users/{{ID}}/deliveryservices
:param user_id: The user to retrieve
:type user_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_users ()
Retrieves all users.

users
:type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

logged_in
Read-only property of to determine if user is logged in to Traffic Ops.

:return: True if connected and logged in, False otherwise
:rtype: bool

login (username, password)
Login to the Traffic Ops API.

:param username: Traffic Ops User Name
:type username: str
:param password: Traffic Ops User Password
:type password: str
:return: None
:rtype: None
:raises: LoginError

servers_queue_update (server_id=None, data=None)
Queue Updates by Server Id.

servers/{{ID}}/queue_update
:param server_id: The server Id
:type server_id: int
:param data: The update action. QueueUpdateRequest() can be used for this argument also.
:type data: Dict[str, Any]
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

snapshot_crconfig (cdn_name=None)
Snapshot CRConfig by CDN Name.

snapshot/{{name}}
:param cdn_name: The CDN name
:type cdn_name: str
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

to_url
The URL without the api portion. (read-only)

Returns The URL should match ‘[w+-.]+://[w+-.]+(:d+)?’ e.g https://to.somedomain.net or https://to.somedomain.net:443

Return type str

update_asn (asn_id=None, query_params=None)
Update ASN

asns/{{id}}
:param asn_id: The ID of the ASN to update
:type asn_id: int
:type: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

update_cache_group_fallbacks (data=None)
Updates an existing fallback configuration for the cache group.

 cachegroup_fallbacks
:param data: The update action. QueueUpdateRequest() can be used for this argument also.
:type data: Dict[str, Any]
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

update_cachegroups (cache_group_id=None, data=None)
Update a cache group

cachegroups/{{ID}}
:param cache_group_id: The cache group id to update
:type cache_group_id: Integer
:param data: The param-


update_cdn_by_id(cdn_id=None, data=None)

Update a CDN by Id.  

\[ \text{cdns/{{ID}}} \]

- :param cdn_id: The CDN id
- :type cdn_id: int
- :param data: The parameter data to use for cdn update.
- :type data: Dict[str, Any]
- :rtype: Tuple[Dict[str, Any], requests.Response]
- :raises: Union[LoginError, OperationError]

update_coordinates(query_params=None, data=None)

Update coordinates:

- :param query_params: The optional url query parameters for the call
- :type query_params: Dict[str, Any]
- :param data: The update action.
- :type data: Dict[str, Any]
- :rtype: Tuple[Dict[str, Any], requests.Response]
- :raises: Union[LoginError, OperationError]

update_deliveryservice_by_id(delivery_service_id=None, data=None)

Update a Delivery Service by Id.

- :param delivery_service_id: The delivery service Id
- :type delivery_service_id: int
- :param data: The request data structure for the API request
- :type data: Dict[str, Any]
- :rtype: Tuple[Dict[str, Any], requests.Response]
- :raises: Union[LoginError, OperationError]

update_deliveryservice_regexes(delivery_service_id=None, regex_id=None, query_params=None)

Update a regex for a delivery service.

- :param delivery_service_id: The delivery service Id
- :type delivery_service_id: int
- :param regex_id: The delivery service regex id
- :type regex_id: int
- :param query_params: The required data to update delivery service regexes
- :type query_params: Dict[str, Any]
- :rtype: Tuple[Dict[str, Any], requests.Response]
- :raises: Union[LoginError, OperationError]

update_deliveryservice_safe(delivery_service_id=None, data=None)

Allows a user to edit limited fields of an assigned delivery service.

- :param delivery_service_id: The delivery service Id
- :type delivery_service_id: int
- :param data: The request data structure for the API request
- :type data: Dict[str, Any]
- :rtype: Tuple[Dict[str, Any], requests.Response]
- :raises: Union[LoginError, OperationError]

update_division(division_id=None, query_params=None)

Update a division by division id

- :param division_id: The division id to update
- :type division_id: int
- :param query_params: The required data to update delivery service regexes
- :type query_params: Dict[str, Any]
- :rtype: Tuple[Dict[str, Any], requests.Response]
- :raises: Union[LoginError, OperationError]

update_federation_in_cdn(cdn_name=None, federation_id=None, query_params=None)


Update a federation. 

```plaintext
cdns/{{name}}/federations/{{ID}}
```

- **param cdn_name**: The CDN name to find federation
- **type cdn_name**: String
- **param federation_id**: The federation id
- **type federation_id**: int
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_origins**(query_params=None)

Updates origins associated with a delivery service.

- **param data**: The update action. QueueUpdateRequest() can be used for this argument also.
- **type data**: Dict[str, Any]
- **param query_params**: The optional url query parameters for the call
- **type query_params**: Dict[str, Any]
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_parameter**(parameter_id=None, query_params=None)

Update Parameter.

- **param parameter_id**: The parameter id to update
- **type parameter_id**: int
- **result**: Tuple[Dict[str, Any], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_physical_location**(physical_location_id=None, query_params=None)

Update Physical Location by id.

- **param physical_location_id**: The id to update
- **type physical_location_id**: int
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_profile_by_id**(profile_id=None, data=None)

Update Profile by Id.

- **param profile_id**: The profile Id
- **type profile_id**: int
- **param data**: The parameter data to edit
- **type data**: Dict[str, Any]
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_region**(region_id=None)

Update a region.

- **param region_id**: The region to update
- **type region_id**: int
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_server_by_id**(server_id=None, data=None)

Update a Server by Id.

- **param server_id**: The server Id
- **type server_id**: int
- **param data**: The parameter data to edit
- **type data**: Dict[str, Any]
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_server_status_by_id**(server_id=None, data=None)

Update server status by Id.

- **param server_id**: The server Id
- **type server_id**: int
- **param data**: The server status
- **type data**: https://traffic-control-cdn.readthedocs.io/en/latest/api/server.html
- **result**: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
- **raises**: Union[LoginError, OperationError]

**update_staticdnsentries**(data=None, query_params=None)

Update static DNS entries associated with the delivery service.

- **param data**: The update action. QueueUpdateRequest() can be used for this

update_tenant (tenant_id=None)

validate_parameter_exists (data=None)

exception trafficops.InvalidJSONError (*args, resp=None)
Bases: ValueError
An error that occurs when an invalid JSON payload is passed to an endpoint.

resp = None
Contains the response object that generated the error

exception trafficops.LoginError (*args)
Bases: OSError
This represents an error that occurred during server login.

exception trafficops.OperationError (*args, resp=None)
Bases: OSError
This class represents a generic error, indicating something went wrong with the request or on the server.

resp = None
Contains the response object that generated the error

trafficops.api_request (method_name, api_path, supported_versions)
This wrapper returns a decorator that routes the calls to the appropriate utility function that generates the RESTful API endpoint, performs the appropriate call to the endpoint and returns the data to the user.

Parameters

- method_name (str) – A method name defined on the Class, this decorator is decorating, that will be called to perform the operation. E.g. ‘GET’, ‘POST’, ‘PUT’, ‘DELETE’, etc. The method_name chosen must have the signature of <method>(self, api_path, **kwargs) e.g. def get (self, api_path, **kwargs):

- api_path (str) – The path to the API end-point that you want to call which does not include the base url e.g. user/login, servers, etc. This string can contain substitution parameters as denoted by a valid field_name replacement field speci-
• supported_versions (Tuple[\text{str}]) – A tuple of API versions that this route supports

Returns  \text{rtype int}: A new function that replaces the original function with a boilerplate execution process.

Return type  \text{Callable[\text{str}, \text{Dict[\text{str, Any}}]]}

class trafficops.RestApiSession (host_ip, api_version=None, api_base_path='api', host_port=443, ssl=True, headers=None, verify_cert=True, create_session=False, max_retries=5)

Bases: object

This class represents a login session with a generic REST API server. It provides base functionality inherited by TOSession.

api_base_url

Returns the base URL. (read-only)

Returns  The base URL should match \([w+.]+://[w+.]+(:d+)?\) e.g. ‘https://to.somedomain.net/api/0.1/’

Return type  \text{str}

api_version

Returns the api version. (read-only)

Returns  The api version from which this instance will request end-points.

Return type  \text{str}

close()

Close and cleanup the requests Session object.

Returns  \text{None}

Return type  \text{NoneType}

create()

Create the requests.Session to communicate with the RESTful API.

Returns  \text{None}

Return type  \text{NoneType}

delete (api_path, *args, **kwargs)

Perform HTTP DELETE requests

Parameters

• \text{api_path (str)} – The path to the API end-point that you want to call which does not include the base URL e.g.
user/login, servers, etc. This string can contain substitution parameters as denoted by a valid field_name replacement field specification as per str.format() e.g. cachegroups/{id} or cachegroups/{id:d}

- **kwargs (Dict[str, Any])** – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter data with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use query_params to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** Union[LoginError, OperationError]

### get (api_path, *args, **kwargs)

Perform http get requests

#### Parameters

- **api_path (str)** – The path to the API end-point that you want to call which does not include the base url e.g. user/login, servers, etc. This string can contain substitution parameters as denoted by a valid field_name replacement field specification as per str.format() e.g. cachegroups/{id} or cachegroups/{id:d}

- **kwargs (Dict[str, Any])** – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter data with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use query_params to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** Union[LoginError, OperationError]

### head (api_path, *args, **kwargs)

Perform HTTP HEAD requests :param api_path: The path to the API end-point that you want to call which does not include the base URL e.g. user/login, servers, etc. This string can contain substitution parameters as denoted by a valid
field_name replacement field specification as per \texttt{\texttt{str.format()}}
e.g. \texttt{cachegroups/{id}} or \texttt{cachegroups/{id:d}}

**Parameters** `kwargs` (**Dict[str, Any]**) – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter `data` with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use `query_params` to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** `Union[LoginError, OperationError]`

**is_open**

Is the session open to the RESTful API? (Read-only Property)

**Returns** `True` if yes, otherwise, `False`

**Return type** `bool`

**options** (`api_path`, *args, **kwargs)

Perform HTTP OPTIONS requests :param api_path: The path to the API end-point that you want to call which does not include the base URL e.g. `user/login`, `servers`, etc. This string can contain substitution parameters as denoted by a valid field_name replacement field specification as per \texttt{\texttt{str.format()}}
e.g. \texttt{cachegroups/{id}} or \texttt{cachegroups/{id:d}}

**Parameters** `kwargs` (**Dict[str, Any]**) – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter `data` with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use `query_params` to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** `Union[LoginError, OperationError]`

**patch** (`api_path`, *args, **kwargs)

Perform HTTP PATCH requests :param api_path: The path to the API end-point that you want to call which does not include
the base URL e.g. user/login, servers, etc. This string can contain substitution parameters as denoted by a valid field_name replacement field specification as per str.format() e.g. cachegroups/{id} or cachegroups/{id:d}

**Parameters**

- **kwargs** (**Dict[str, Any]**) – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter data with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use query_params to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** Union[LoginError, OperationError]

### post (api_path, *args, **kwargs)

Perform http post requests

**Parameters**

- **api_path** (**str**) – The path to the API end-point that you want to call which does not include the base URL e.g. user/login, servers, etc. This string can contain substitution parameters as denoted by a valid field_name replacement field specification as per str.format() e.g. cachegroups/{id} or cachegroups/{id:d}

- **kwargs** (**Dict[str, Any]**) – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter data with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use query_params to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** Union[LoginError, OperationError]

### put (api_path, *args, **kwargs)

Perform http put requests

**Parameters**

- **api_path** (**str**) – The path to the API end-point that you want to call which does not include the base URL e.g.
This string can contain substitution parameters as denoted by a valid field_name re-
placement field specification as per str.format() e.g. cachegroups/{id} or cachegroups/{id:d}

- **kwargs** (Dict[str, Any]) – Passed Keyword Parameters. If you need to send JSON data to the endpoint pass the keyword parameter data with the Python data structure. This method will convert it to JSON before sending it to the API endpoint. Use query_params to pass a dictionary of query parameters

**Returns** Python data structure distilled from JSON from the API request.


**Raises** Union[LoginError, OperationError]

**server_url**

The URL without the api portion. (read-only)

**Returns** The URL should match ‘[w+-.]++//[w+-.]+:(d+)’ e.g. ‘https://to.somedomain.net’ or ‘https://to.somedomain.net:443’

**Return type** str

**session**

The RESTful API session (Read-only Property)

**Returns** The requests session

**Return type** requests.Session

**utils**

Useful utility methods

trafficops.utils.log_with_debug_info (logging_level=20, msg=", parent=False, separator=‘.’)

Uses inspect module(reflection) to gather debugging information for the source file name, function name, and line number of the calling function/method.

**Parameters**

- **logging_level** (int) – The logging level from the logging module constants E.g. logging.INFO, logging.DEBUG, etc.

- **msg** (Text) – The message to log.

- **parent** (bool) – If True, use the caller’s parent information instead of the caller’s information in the message.

- **separator** (Text) – The string to use for the component sepa-
Returns `<file name>:<function name>:<line number>: <msg>` e.g. `tosession.py:_build_endpoint:199: This is a message to log.`

Return type Text

Versioning

The `trafficops.__version__` module contains only the `__version__` “constant” which gives the version of this Apache-TrafficControl package and not the version of Apache Traffic Control for which it was made. The two are versioned separately, to allow the client to grow in a version-controlled manner without being tied to the release cadence of Apache Traffic Control as a whole.

Version 1.0 is supported for use with Apache Traffic Control version 3.0 (release pending at the time of this writing). New functionality will be added as the Traffic Ops API evolves, but changes to this client will remain non-breaking for existing code using it until the next major version is released.

Deprecated since version 1.0: The v1.0 release of this client deprecates support of Python2. Versions 2.0 and onward will only support Python3 (v3.4+). Note that this release is expected either by the time Python2 reaches its end-of-life at the end of 2019, or with the release of Apache Traffic Control v4.0, should that happen first. Users and developers are encouraged to switch to Python3 as soon as possible.

6.1.3 Traffic Vault Util

The `traffic_vault_util` tool - located at `tools/traffic_vault_util.go` in the Apache Traffic Control repository - is used to view and modify the contents of a Traffic Vault (i.e. Riak) cluster. The tool contains basic operations to display the buckets, keys and values stored within Traffic Vault.

`traffic_vault_util` also has a small converter utility to perform a one-off conversion of key formats within the SSL bucket. This conversion is useful when moving from an older version of Traffic Ops to the current version. In the older version, SSL records were indexed by Delivery Service database ID. Currently, SSL records are indexed by Delivery Service xml_id.

Usage

```
traffic_vault_util [--dry_run] --vault_ip IP --vault_action ACTION [--vault_user USER] [--vault_password PASSWD] [--vault_port PORT]
```

`--dry_run`

An optional flag which, if given, will cause `traffic_vault_util` to not write changes, but merely print what would be done in a real run.

`--vault_action ACTION`

Defines the action to be performed. Available actions are:
**list_buckets** Lists the “buckets” in the Riak cluster used by Traffic Vault
**list_keys** Lists all the keys in all the buckets in the Riak cluster used by Traffic Vault
**list_values** Lists all the values of all the keys in all the buckets in the Riak cluster used by Traffic Vault

**convert_ssl_to_xmlid** Changes the key of all records in all buckets that start with “ds” into the xml_id of the *Delivery Service* for which we assume the record was created.

**--vault_ip** IP
Either the IP address or FQDN of the Traffic Vault instance with which *traffic_vault_util* will interact.

| Warning: | If this IP address or FQDN does not point to a real Riak cluster, *traffic_vault_util* will print an error message to STDOUT, but will not terminate. Instead, it will try forever to query the server to which it failed to connect, consuming large amounts of CPU usage all the while.

**--vault_password** PASSWD
An optional flag used to specify the password of the user defined by **--vault_user** when authenticating with Traffic Vault’s Riak cluster.

| Warning: | Although this flag is optional, the utility will not work without it. It will try, but it will fail.

**--vault_port** PORT
An optional flag which, if given, sets the port to which *traffic_vault_util* will try to connect to Riak. Default: 8087

**--vault_user** USER
An optional flag which, if given, specifies the name of the user as whom to connect to Riak.

| Warning: | Although this flag is optional, the utility will not work without it. It will try, but it will fail.

### 6.1.4 toaccess

**toaccess**

This module provides a set of functions meant to provide ease-of-use functionality for interacting with the Traffic Ops API. It provides scripts named *tomethod* where *method* is the

---

1 These problems are all tracked by GitHub Issue #3261.
name of an HTTP method (in lowercase). Collectively they are referred to as toaccess. Implemented methods thus far are:

- delete
- head
- get
- options
- patch
- post
- put

Arguments and Flags

PATH
This is the request path. By default, whatever is passed is considered to be relative to /api/api-version/ where api-version is --api-version. This behavior can be disabled by using --raw-path.

DATA
An optional positional argument that is a data payload to pass to the Traffic Ops server in the request body. If this is the absolute or relative path to a file, the contents of the file will instead be read and used as the request payload.

-h, --help
Print usage information and exit

-a API_VERSION, --api-version API_VERSION
Specifies the version of the Traffic Ops API that will be used for the request. Has no effect if --raw-path is used. (Default: 1.3)

-f, --full
Output the full HTTP exchange including request method line, request headers, request body (if any), response status line, and response headers (as well as the response body, if any). This is equivalent to using --request-headers, --request-payload, and --response-headers at the same time, and those options will have no effect if given. (Default: false)

-k, --insecure
Do not verify SSL certificates - typically useful for making requests to development or testing servers as they frequently have self-signed certificates. (Default: false)

-p, --pretty
Pretty-print any payloads that are output as formatted JSON. Has no effect on plaintext payloads. Uses tab characters for indentation. (Default: false)

-r, --raw-path
Request exactly PATH; do not preface the request path with /api/api_version. This effectively means that --api-version will have no effect. (Default: false)
--request-headers
Output the request method line and any and all request headers. (Default: false)

--request-payload
Output the request body if any was sent. Will attempt to pretty-print the body as JSON if
--pretty is used. (Default: false)

--response-headers
Output the response status line and any and all response headers. (Default: false)

--to-url URL
The FQDN and optionally the port and scheme of the Traffic Ops server. This will
override TO_URL. The format is the same as for TO_URL. (Default: uses the value of
TO_URL)

--to-password PASSWORD
The password to use when authenticating to Traffic Ops. Overrides TO_PASSWORD.
(Default: uses the value of TO_PASSWORD)

--to-user USERNAME
The username to use when connecting to Traffic Ops. Overrides TO_USER. (Default:
uses the value of TO_USER)

Environment Variables

If defined, toaccess scripts will use these environment variables to define their connection
to and authentication with the Traffic Ops server. Typically, setting these is easier than using
the long options --to-url, --to-user, and --to-password on every invocation.

TO_PASSWORD
Will be used to authenticate the user defined by either --to-user or TO_USER.

TO_URL
The FQDN of the Traffic Ops server to which the script will connect. The format of this
should be [http or https://]hostname[:port]. Note that this may option-
ally start with http:// or https:// (case insensitive), but typically this is unneces-
sary. Also notice that the port number may be specified, though again this isn’t usually
required. All toaccess scripts will assume that port 443 should be used unless other-
wise specified. They will further assume that the protocol is HTTPS unless TO_URL (or
--to-url) starts with http://, in which case the default port will also be set to 80
unless otherwise specified in the URL.

TO_USER
The name of the user as whom to connect to the Traffic Ops server. Overriden by
--to-user.

Exit Codes

The exit code of a toaccess script can sometimes be used by the caller to determine what
the result of calling the script was without needing to parse the output. The exit codes used are:
0 The command executed successfully, and the result is on STDOUT.

1 Typically this exit code means that an error was encountered when parsing positional command line arguments. However, this is also the exit code used by most Python interpreters to signal an unhandled exception.

2 Signifies a runtime error that caused the request to fail - this is not generally indicative of an HTTP client or server error, but rather an underlying issue connecting to or authenticating with Traffic Ops. This is distinct from an exit code of 32 in that the format of the arguments was correct, but there was some problem with the value. For example, passing https://test: to --to-url will cause an exit code of 2, not 32.

4 An HTTP client error occurred. The HTTP stack will be printed to stdout as indicated by other options - meaning by default it will only print the response payload if one was given, but will respect options like e.g. --request-payload as well as -pl--pretty.

5 An HTTP server error occurred. The HTTP stack will be printed to stdout as indicated by other options - meaning by default it will only print the response payload if one was given, but will respect options like e.g. --request-payload as well as -pl--pretty.

32 This is the error code emitted by Python’s argparse module when the passed arguments could not be parsed successfully.

Note: The way exit codes 4 and 5 are implemented is by returning the status code of the HTTP request divided by 100 whenever it is at least 400. This means that if the Traffic Ops server ever started returning e.g. 700 status codes, the exit code of the script would be 7.

Module Reference

to_access.delete()
   Entry point for todelete
   Returns The program’s exit code

to_access.get()
   Entry point for toget
   Returns The program’s exit code

to_access.head()
   Entry point for tohead
   Returns The program’s exit code

to_access.options()
   Entry point for tooptions
   Returns The program’s exit code

to_access.output(r, pretty, request_header, response_header, request_payload, indent='\n')
   Prints the passed response object in a format consistent with the other parameters.
   Parameters
   • r – The requests response object being printed
- **pretty** – If True, attempt to pretty-print payloads as JSON
- **request_header** – If True, print request line and request headers
- **response_header** – If True, print response line and response headers
- **request_payload** – If True, print the request payload
- **indent** – An optional number of spaces for pretty-printing indentation (default is the tab character)

```python
to_access.parse_arguments (program)
```
A common-use function that parses the command line arguments.

**Parameters** program – The name of the program being run - used for usage informational output

**Returns** The Traffic Ops HTTP session object, the requested path, any data to be sent, an output format specification, whether or not the path is raw, and whether or not output should be prettified

```python
to_access.patch ()
```
Entry point for `topatch`

**Returns** The program’s exit code

```python
to_access.post ()
```
Entry point for `topost`

**Returns** The program’s exit code

```python
to_access.put ()
```
Entry point for `toput`

**Returns** The program’s exit code

```python
to_access.request (method)
```
All of the scripts wind up calling this function to handle their common functionality.

**Parameters** method – The name of the request method to use (case-insensitive)

**Returns** The program’s exit code
7.1 FAQ

7.1.1 Who is using Traffic Control?

- Comcast Cable is the original developer of Traffic Control and is using it for all its IP video delivery, delivering images and software to its X1 platform, and for delivering third party content to its footprint.
- Cox Communications
- Cisco has a product called Open Media Distribution that is based on Traffic Control.
- Concurrent has a product that uses Traffic Control, see their github page for more info.
- Augere Pakistan / QUBEE
- Qwilt has a product called Open Edge CDN that is based on Traffic Control.

7.1.2 How do I get help with Traffic Control?

Join us on our Slack or send your questions to our mailing list. Slack is usually the best forum for quick Q&A type issues – like when you are getting a Traffic Control CDN up and running or if you have a question about configuration. Any discussions that could potentially lead to decisions being made about the project – like a new feature – should happen on the developer mailing list-.
7.1.3 How do I get involved with the development of Traffic Control?

See the CONTRIBUTING.md file.

7.1.4 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.

7.1.5 What is the CCR?

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.

7.1.6 What is Twelve Monkeys?

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

7.1.7 What license is Traffic Control released under?

See the LICENSE file.
8.1 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 Groups** A group of caching HTTP proxy servers that together create a combined larger cache using consistent hashing. 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 general area. A *Cache Group* has one single set of geographical coordinates even if the *cache servers* that make up the *Cache Group* are actually in *Physical Locations*. The *cache servers* in a *Cache Group* are not aware of the other *cache servers* in the group - there is no clustering software or communications between *cache servers* in a *Cache Group*.

There are two basic types of *Cache Groups*: EDGE_LOC and MID_LOC (“LOC” being short for “location” - a holdover from when *Cache Groups* were called “Cache Locations). Traffic Control is a two-tiered system, where the clients get directed to the Edge-tier (EDGE_LOC) *Cache Group*. On cache miss, the *cache server* in the Edge-tier *Cache Group* obtains content from a Mid-tier (MID_LOC) *Cache Group*, rather than the origin, which is shared with multiple Edge-tier *Cache Groups*. Edge-tier *Cache Groups* are configured to have a single “parent” *Cache Group*, but in general Mid-tier *Cache Groups* have many “children”.

**Note:** Often the Edge-tier to Mid-tier relationship is based on network distance, and
does not necessarily match the geographic distance.

See also:

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* for details.

Consider the example CDN in Fig. 36. Here some country/province/region has been divided into quarters: Northeast, Southeast, Northwest, and Southwest. The arrows in the diagram indicate the flow of requests. If a client in the Northwest, for example, were to make a request to the *Delivery Service*, it would first be directed to some *cache server* in the “Northwest” Edge-tier *Cache Group*. Should the requested content not be in cache, the Edge-tier server will select a parent from the “West” *Cache Group* and pass the request up, caching the result for future use. All Mid-tier *Cache Groups* (usually) answer to a single *origin* that provides canonical content. If requested content is not in the Mid-tier cache, then the request will be passed up to the *origin* and the result cached.

![Fig. 36: An example CDN that shows the hierarchy between four Edge-tier Cache Groups, two Mid-tier Cache Groups, and one Origin](image)

**cache server**

*cache servers* 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:

- **Reverse Proxy**: Used by Traffic Control for Edge-tier *cache servers*.
- **Forward Proxy**: Used by Traffic Control for Mid-tier *cache servers*.
- Transparent Proxy: These are not used by Traffic Control. If you are interested you can learn more about transparent proxies on wikipedia.

**consistent hashing** See the [Wikipedia article](http://example.com); 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 File**
Coverage Zone Map  The CZM (Coverage Zone Map) or CZF is a file that maps network prefixes to Cache Groups. Traffic Router uses the CZM to determine what Cache Group is closest to the client. If the client IP address is not in this CZM, it falls back to geographic mapping, using a MaxMind GeoIP2 database to find the client’s location, and the geographic coordinates from Traffic Ops for the Cache Group. Traffic Router is inserted into the HTTP retrieval process by making it the authoritative DNS server 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 a routing name, which is configurable per-Delivery Service, e.g. http://foo.mydeliveryservice.cdn.com/fun/example.html with the chosen routing name foo.

Deep Coverage Zone Map  The DCZF or DCZM (Deep Coverage Zone Map) maps network prefixes to “locations” - almost like the Coverage Zone File. Location names must be unique, and within the file are simply used to group Edge-tier cache servers. When a mapping is performed by Traffic Router, it will only look in the DCZF if the Delivery Service to which a client is being directed makes use of Deep Caching. If the client’s IP address cannot be matched by entries in this file, Traffic Router will first fall back to the regular Coverage Zone File. Then, failing that, it will perform geographic mapping using a database provided by the Delivery Service’s Geolocation Provider.

Delivery Services  Delivery Services are often referred to as a reverse proxy “remap rule” that exists on Edge-tier cache servers. In most cases, a Delivery Service is a one-to-one mapping to an FQDN that is used as a hostname to deliver the content. Many options and settings regarding how to optimize the content delivery exist, which are 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.
- Limits on transactions per second and bandwidth.
- Protocol (HTTP or HTTPS).
- Token-based authentication settings.
- Header rewrite rules.

Since Traffic Control version 2.1, Delivery Services can optionally be linked to a Profile, and have Parameters associated with them. One example of a feature that uses Delivery Service Parameters is the Use Multi-Site Origin Feature configuration. Delivery Services are also for use in allowing multiple Tenants to coexist in a Traffic Control CDN without interfering with each other, and to keep information about their content separated.

See also:

See Delivery Services for a more in-depth explanation of Delivery Services.

Division  A group of Regions.
Edge

Edge-tier

Edge-tier cache

Edge-tier caches

Edge-tier cache server

**Edge-tier cache servers** 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*.

Federation

**Federations** *Federations* allow for other (“federated”) CDNs (e.g. at a different ISP (Internet Service Provider)) to add a list of DNS resolvers and an FQDN to be used in a DNS CNAME record for a *Delivery Service*. When a request is made from one of the federated CDN’s clients, Traffic Router will return the CNAME record configured from the federation mapping. This allows the federated CDN to serve the content without the content provider changing the URL, or having to manage multiple URLs. For example, if the external CDN was actually another ATC-managed CDN, then a federation mapping to direct clients toward it should use the FQDN of a *Delivery Service* on the external CDN.

Federations only have meaning to DNS-routed *Delivery Services* - HTTP-routed Delivery services should instead treat the external FQDN as an *origin* to achieve the same effect.

See also:

Federations are currently only manageable by directly using the *Traffic Ops API*. The endpoints related to federations are *federations*, *federation_resolvers*, *federation_resolvers/***/ID*/, *deliveryservices*, *deliveryservices/***/ID*/, *deliveryservices/***/dsID*/, *federation_resolvers*, *federations/***/ID*/users, and *federations/***/ID*/users/***/userID*/.

**forward proxy** A forward proxy acts on behalf of the client such that the *origin server* is (potentially) unaware of the proxy’s existence. All Mid-tier *cache servers* in a Traffic Control based CDN are *forward proxies*. In a *forward proxy* scenario, the client is explicitly configured to use the the proxy’s IP address and port as a *forward proxy*. 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 (potentially) unaware of the proxy in the middle.

See also:

*ATS documentation on forward proxy.*

If a client uses a *forward proxy* to request the URL `http://www.origin.com/foo/bar/fun.html` the resulting chain of events follows.

1. To retrieve `http://www.origin.com/foo/bar/fun.html`, the client sends an HTTP request to the *forward proxy*. 

---

Chapter 8. Indices and Tables
# 510: Client Requests Content from its Forward Proxy

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

Note: In this case, the client requests the entire URL instead of just the path as is the case when using a reverse proxy or when requesting content directly from the origin server.

2. The proxy verifies whether the response for `http://www-origin-cache.cdn.com/foo/bar/fun.html` is already in the cache. If it is not in the cache:

1. The proxy sends the HTTP request to the origin.

# 511: The Forward Proxy Requests Content from the Origin Server

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

2. The origin server responds with the requested content.

# 512: The Origin Server's Response

```
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

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

3. The proxy sends this on to the client, optionally adding a Via: header to indicate that the request was serviced by proxy.

# 513: The Forward Proxy's Response to the Client

```
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
```

(continues on next page)
If, however, the requested content was in the cache the proxy responds to the client with the previously retrieved result

# 514: The Forward Proxy Sends the Cached Response

```
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

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

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

Mid

Mid-tier

Mid-tier cache

Mid-tier caches

Mid-tier cache server

Mid-tier cache servers  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.

origin
origins

origin server

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

ORT The “Operational Readiness Test” script that stitches the configuration configured in Traffic Portal and generated by Traffic Ops into the cache servers. See Configuring Traffic Server for more information.

Parameter

Parameters Typically refers to a line in a configuration file, but in practice can represent any arbitrary configuration option

parent

parents The parent(s) of a cache server is/are the cache server(s) belonging to either the “parent” or “secondary parent” Cache Group(s) of the Cache Group to which the cache server belongs. For example, in general it is true that an Edge-tier cache server has one or more parents which are Mid-tier cache servers.

Physical Location A pair of geographic coordinates (latitude and longitude) that is used by Cache Groups to define their location. This information is used by Traffic Router to route client traffic to the geographically nearest Cache Group.

Profile A Profile is, most generally, a group of Parameters that will be applied to a server. Profiles are typically re-used by all Edge-tier cache servers within a CDN or Cache Group. A Profile will, in addition to configuration Parameters, define the CDN to which a server belongs and the “Type” of the profile - which determines some behaviors of Traffic Control components. The allowed “Types” of Profiles are not the same as Types, and are maintained as a PostgreSQL “Enum” in traffic_ops/app/db/migrations/20170205101432_cdn_table_domain_name.go. The only allowed values are:

UNK_PROFILE A catch-all type that can be assigned to anything without imbuing it with any special meaning or behavior

TR_PROFILE A Traffic Router Profile.

Warning: For legacy reasons, the names of Profiles of this type must begin with CCR__ or TR__. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

TM_PROFILE A Traffic Monitor Profile.

Warning: For legacy reasons, the names of Profiles of this type must begin with RASCAL__ or TM__. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!
**TS_PROFILE**  A Traffic Stats Profile

**Warning:** For legacy reasons, the names of Profiles of this type must begin with TRAFFIC_STATS. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**TP_PROFILE**  A Traffic Portal Profile

**Warning:** For legacy reasons, the names of Profiles of this type must begin with TRAFFIC_PORTAL. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**INFLUXDB_PROFILE**  A Profile used with InfluxDB, which is used by Traffic Stats.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with INFLUXDB. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**RIAK_PROFILE**  A Profile used for each Riak server in a Traffic Stats cluster.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with RIAK. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**SPLUNK_PROFILE**

A Profile meant to be used with Splunk servers.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with SPLUNK. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**ORG_PROFILE**  Origin Profile.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with MSO, or contain either ORG or ORIGIN anywhere in the name. This is not
enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**KAFKA_PROFILE** A Profile for Kafka servers.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with KAFKA. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**LOGSTASH_PROFILE** A Profile for Logstash servers.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with LOGSTASH_. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**ES_PROFILE** A Profile for ElasticSearch servers.

**Warning:** For legacy reasons, the names of Profiles of this type must begin with ELASTICSEARCH. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**ATS_PROFILE** A Profile that can be used with either an Edge-tier or Mid-tier cache server’ (but not both, in general).

**Warning:** For legacy reasons, the names of Profiles of this type must begin with EDGE or MID. This is not enforced by the Traffic Ops API or Traffic Portal, but certain Traffic Control operations/components expect this and will fail to work otherwise!

**Tip:** A Profile of the wrong type assigned to a Traffic Control component will (in general) cause it to function incorrectly, regardless of the Parameters assigned to it.

**Danger:** Nearly all of these Profile types have strict naming requirements, and it may be noted that some of said requirements are prefixes ending with __, while others are either not prefixes or do not end with __. This is exactly true; some requirements need that __ and some may or may not have it. It is our suggestion, therefore, that for
the time being all prefixes use the _ notation to separate words, so as to avoid causing headaches remembering when that matters and when it does not.

Region  A group of Physical Locations.

**reverse proxy**  A reverse proxy acts on behalf of the origin server such that the client is (potentially) unaware it is not communicating directly with the origin. All Edge-tier cache servers 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 (RFC 2616), the client sends a Host: header to the reverse proxy that matches the hostname in the URL. The proxy looks up this hostname in a list of mappings to find the origin hostname; if the hostname of the Host: header is not found in the list, the proxy will send an error (usually either 404 Not Found or 503 Service Unavailable as appropriate) to the client. If the supplied hostname is found in this list of mappings, the proxy checks its cache, and when the content is not already present, connects to the origin to which the requested Host: maps requests the path of the original URL, providing the origin hostname in the Host header. The proxy then stores the URL in its 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 its cache - provided Cache Control Headers and Revalidation are satisfied - thereby reducing latency and network traffic.

**See also:**

The Apache Traffic Server documentation on reverse proxy.

To insert a reverse proxy into a typical HTTP 1.1 request and response flow, the reverse proxy needs to be told where the origin server can be reached (and which origin to use for a given request when it’s configured to proxy requests for multiple origins). In ATS this is handled by adding rules to the remap.config configuration file. 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 example, clients might be instructed to request content from http://www-origin-cache.cdn.com which points to a reverse proxy for the actual origin located at http://www.origin.com.

Now, if the client requests /foo/bar/fun.html from the reverse proxy the sequence of events is as follows. 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 DNS request to the LDNS to resolve the name www-origin-cache.cdn.com to an IP address.
2. The LDNS finds an IP address for www-origin-cache.cdn.com e.g. 55.44.33.22.
3. The client sends an HTTP request for /foo/bar/fun.html to the IP address.
4. The reverse proxy finds out the URL of the true origin - in the case of ATS this is done by looking up www-origin-cache.cdn.com in its remap rules - and finds that it is www.origin.com.

5. The proxy checks its cache to see if the response for GET /foo/bar/fun.html HTTP/1.1 from www.origin.com is already in the cache.

6. If the response is not in the cache:
   1. The proxy sends the request to the actual origin

   # 516: Reverse Proxy Requests Content from the Origin Server
   GET /foo/bar/fun.html HTTP/1.1
   Host: www.origin.com

2. The origin server responds with the requested content

   # 517: Response from the Origin Server
   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

   <!DOCTYPE html>
   <html>
   <body>
   This is a fun file</body>
   </html>

3. The proxy sends the response on to the client, optionally adding a Via: header to indicate that the request was serviced by proxy.

   # 518: Resulting Response from the Reverse Proxy to the Client
   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

   (continues on next page)
If, however, the response was already in the cache - and still valid according to the Cache Control Headers and Revalidation - the proxy responds to the client with the previously retrieved result.

### # 519: The Reverse Proxy Provides a Cached Response

```http
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 cCi p sS])
Server: ATS/4.2.1

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

**Role** Permissions *Roles* define the operations a user is allowed to perform, and are currently an ordered list of permission levels.

**Snapshot** Previously called a “CRConfig” or “CRConfig.json” (and still called such in many places), this is a rather large set of routing information generated from a CDN’s configuration and topology.

**Status** A *Status* represents the current operating state of a server. The default *Statuses* made available on initial startup of Traffic Ops are related to the *Health Protocol* and are explained in that section.

**Tenant**

**Tenants** Users are grouped into *Tenants* (or Tenancies) to segregate ownership of and permissions over *Delivery Services* and their resources. To be clear, the notion of Tenancy only applies within the context of *Delivery Services* and does not apply permissions restrictions to any other aspect of Traffic Control.
Type

**Types**  A *Type* defines a type of some kind of object configured in Traffic Ops. Unfortunately, that is exactly as specific as this definition can be.
t

- to_access, 788
- traffic_ops Ort, 184
- traffic_ops Ort.configuration, 191
- traffic_ops Ort.main_routines, 193
- traffic_ops Ort.packaging, 195
- traffic_ops Ort.to_api, 196
- traffic_ops Ort.utils, 199
- trafficops, 756
- trafficops.__version__, 787
- trafficops.utils, 786
### Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>database database_name</code></td>
<td>-database database_name</td>
</tr>
<tr>
<td><code>sync_ts_databases command</code></td>
<td>index.html line option, 123</td>
</tr>
<tr>
<td><code>days N</code></td>
<td>index.html line option, 123</td>
</tr>
<tr>
<td><code>dir directory</code></td>
<td>command line option, 144</td>
</tr>
<tr>
<td><code>dispersion DISP</code></td>
<td>traffic_ops_ort command line option, 185</td>
</tr>
<tr>
<td><code>dry_run</code></td>
<td>traffic_vault_util command line option, 787</td>
</tr>
<tr>
<td><code>env ENVIRONMENT</code></td>
<td>admin command line option, 180</td>
</tr>
<tr>
<td><code>help</code></td>
<td>create_ts_databases command line option, 122</td>
</tr>
<tr>
<td><code>http port</code></td>
<td>sync_ts_databases command line option, 123</td>
</tr>
<tr>
<td><code>login Dispersion DISP</code></td>
<td>traffic_ops_ort command line option, 185</td>
</tr>
<tr>
<td><code>password password</code></td>
<td>create_ts_databases command line option, 122</td>
</tr>
<tr>
<td><code>refPasswd PASSWD</code></td>
<td>genConfigRoutes.py command line option, 753</td>
</tr>
<tr>
<td><code>refURL URL</code></td>
<td>sync_ts_databases command line option, 123</td>
</tr>
<tr>
<td><code>loginDispersion DISP</code></td>
<td>traffic_ops_ort command line option, 185</td>
</tr>
<tr>
<td><code>request-headers</code></td>
<td>toaccess command line option, 789</td>
</tr>
<tr>
<td><code>request-payload</code></td>
<td>toaccess command line option, 790</td>
</tr>
<tr>
<td><code>response-headers</code></td>
<td>toaccess command line option, 790</td>
</tr>
<tr>
<td><code>retries RETRIES</code></td>
<td>traffic_ops_ort command line option, 185</td>
</tr>
<tr>
<td><code>rev_prox_disable</code></td>
<td>traffic_ops_ort command line option, 185</td>
</tr>
<tr>
<td><code>source-password password</code></td>
<td>sync_ts_databases command line option, 123</td>
</tr>
<tr>
<td><code>source-url URL</code></td>
<td>sync_ts_databases command line option, 123</td>
</tr>
</tbody>
</table>
sync_ts_databases command line option, 123
-user username
create_ts_databases command line option, 123
-vault_action ACTION
traffic_vault_util command line option, 787
-vault_ip IP
traffic_vault_util command line option, 788
-vault_password PASSWD
traffic_vault_util command line option, 788
-vault_port PORT
traffic_vault_util command line option, 788
-vault_user USER
traffic_vault_util command line option, 788
-wait_for_parents INT
traffic_ops_ort command line option, 185
-C, -no-server-configs
genConfigRoutes.py command line option, 754
-V, -version
compare command line option, 753
-a API_VERSION, -api-version
API_VERSION
toaccess command line option, 789
-f FILE, -file FILE
compare command line option, 753
-f, -full
toaccess command line option, 789
-h, -help
compare command line option, 185
genConfigRoutes.py command line option, 753
toaccess command line option, 185
-k, -insecure
genConfigRoutes.py command line option, 754
toaccess command line option, 789
traffic_ops_ort command line option, 185
-l LOG_LEVEL, -log_level
genConfigRoutes.py command line option, 754
-p PASSWORD, -to_password
traffic_ops_ort command line option, 187
-p, -pretty
toaccess command line option, 789
-r PATH, -results_path PATH
compare command line option, 753
-r, -raw-path
toaccess command line option, 789
-s, -snapshot
traffic_ops_ort command line option, 754
-u USER, -to_user USER
traffic_ops_ort command line option, 187
-v, -version
traffic_ops_ort command line option, 754
traffic_ops_ort command line option, 185
302 content routing, 795

A

add_ssl_keys_to_deliveryservice()
(trafficops.TOSession method), 758
admin command line option
-environment ENVIRONMENT, 180
command, 180
ALL (traffic_ops_ort.configuration.LogLevels attribute), 192
API (class in traffic_ops_ort.to_api), 196
api (traffic_ops_ort.configuration.Configuration attribute), 191
api_base_url (trafficops.RestApiSession attribute), 782
API_LOGGERS (in module traffic_ops_ort.to_api), 197
api_request () (in module trafficops), 781
api_version (trafficops.RestApiSession attribute), 782
assign_delivery_services_to_federations()
(trafficops.TOSession method), 758
assign_deliveryservice_servers_by_ids()
(trafficops.TOSession method), 758
assign_deliveryservice_servers_by_names()
(trafficops.TOSession method), 758
assign_federation_resolver_to_federations()
(trafficops.TOSession method), 759
assign_parameter_to_profile_ids()
(trafficops.TOSession method), 759
assign_profile_to_parameter_ids()
(trafficops.TOSession method), 759
associateParameter_to_profile()
(trafficops.TOSession method), 759
associateParameters_by_profile_id()
(trafficops.TOSession method), 759
associateParameters_by_profile_name()
(trafficops.TOSession method), 759
astats (stats_over_http), 795
B
BADDASS (traffic_ops_ort.configuration.Configuration.Modes attribute), 191
base_url (trafficops.TOSession attribute), 759
C
Cache Group, 795
Cache Groups, 795
cache server, 796
cache servers, 796
Cache Updates, 73
CACHED_STATUSES (in module traffic_ops_ort.to_api), 198
cachegroups_queue_update () (trafficops.TOSession method), 758
cdnId (traffic_ops_ort.to_api.ServerInfo attribute), 198
cdnName (traffic_ops_ort.to_api.ServerInfo attribute), 198
cdns_queue_update () (trafficops.TOSession method), 760
Change Log, 64
checkInstallList() (traffic_ops_ort.packaging.Package method), 195
close() (trafficops.RestApiSession method), 782
command
  admin command line option, 180
  command line option
    -dir directory, 144
    -http port, 144
    -started_filename, 144
compare command line option
  -ref_passwd PASSWD, 752
  -ref_url URL, 752
  -ref_user USER, 752
  -test_passwd PASSWD, 752
  -test_url URL, 752
  -test_user USER, 752
  -V, -version, 753
  -f FILE, -file FILE, 753
  -h, -help, 753
  -r PATH, -results_path PATH, 753
Configuration (class in traffic_ops_ort.configuration), 191
Configuration.Modes (class in traffic_ops_ort.configuration), 191
consistent hashing, 796
content routing, 796
copy_profile() (trafficops.TOSession method), 760
Coverage Zone File, 796
Coverage Zone Map, 797
create() (trafficops.RestApiSession method), 782
create_asn() (trafficops.TOSession method), 760
create_cache_group_fallbacks() (trafficops.TOSession method), 760
create_cachegroups() (trafficops.TOSession method), 760
create_cdn() (trafficops.TOSession method), 760
create_cdn_dns_sec_keys() (trafficops.TOSession method), 760
create_coordinates() (trafficops.TOSession method), 761
create_delivery_service_user_link() (trafficops.TOSession method), 761
create_deliveryservice() (trafficops.TOSession method), 761
create_deliveryservice_regexes() (trafficops.TOSession method), 761
create_division() (trafficops.TOSession method), 761
create_federation() (trafficops.TOSession method), 761
create_federation_in_cdn() (trafficops.TOSession method), 761
create_federation_resolver() (trafficops.TOSession method), 761
create_federation_user() (trafficops.TOSession method), 761
create_invalidation_job() (trafficops.TOSession method), 762
create_origins() (trafficops.TOSession method), 762
create_parameter() (trafficops.TOSession method), 762
create_physical_location() (trafficops.TOSession method), 762
create_profile() (trafficops.TOSession method), 762
create_region() (trafficops.TOSession method), 762
create_server() (trafficops.TOSession method), 762
create_servercheck() (trafficops.TOSession method), 763
create_staticdnssentries() (trafficops.TOSession method), 763
create_tests() (trafficops.TOSession method), 763
create_tests() (trafficops.TOSession method), 763
create_tenant() (trafficops.TOSession method), 763
create_to_extension() (trafficops.TOSession method), 763
create_ts_databases command line option
  -help, 122
  -password password, 122
  -replication N, 123
  -url URL, 123
  -user username, 123
create_user() (trafficops.TOSession method), 763
create_user_with_registration() (trafficops.TOSession method), 763

CRITICAL (traffic_ops_ort.configuration.LogLevels attribute), 192

D

DATA
toaccess command line option, 789

DEBUG (traffic_ops_ort.configuration.LogLevels attribute), 192

Deep Coverage Zone File, 797
Deep Coverage Zone Map, 797
delete() (in module to_access), 791
delete() (trafficops.RestApiSession method), 782

delete_asn() (trafficops.TOSession method), 763

delete_cache_group_fallbacks() (trafficops.TOSession method), 763
delete_cache_group_parameters() (trafficops.TOSession method), 764
delete_cachegroups() (trafficops.TOSession method), 764
delete_cdn_by_id() (trafficops.TOSession method), 764
delete_cdn_dns_sec_keys() (trafficops.TOSession method), 764
delete_coordinates() (trafficops.TOSession method), 764
delete_delivery_service_user_link() (trafficops.TOSession method), 764
delete_deliveryservice_by_id() (trafficops.TOSession method), 764
delete_deliveryservice_regex_by_regex_id() (trafficops.TOSession method), 764
delete_deliveryservice_servers_by_id() (trafficops.TOSession method), 765
delete_deliveryservice_ssl_keys_by_xml_id() (trafficops.TOSession method), 765
delete_division() (trafficops.TOSession method), 765
delete_federation_in_cdn() (trafficops.TOSession method), 765
delete_federation_user() (trafficops.TOSession method), 765
delete_origins() (trafficops.TOSession method), 765
delete_parameter() (trafficops.TOSession method), 765
delete_physical_location() (trafficops.TOSession method), 766
delete_profile_by_id() (trafficops.TOSession method), 766
delete_profile_parameter_association_by_id() (trafficops.TOSession method), 766
delete_server_by_id() (trafficops.TOSession method), 766
delete_staticdnssentries() (trafficops.TOSession method), 766
delete_to_extension() (trafficops.TOSession method), 766
deleteOldStatusFiles() (in module traffic_ops_ort.main_routines), 193

Delivery Service, 797
Delivery Services, 797
DISTRO (in module traffic_ops_ort.configuration), 192
Division, 797
doMain() (in module traffic_ops_ort), 190

E

Edge, 798
Edge Health, 65
Edge-tier, 798
Edge-tier cache, 798
Edge-tier cache server, 798
Edge-tier cache servers, 798
Edge-tier caches, 798
environment variable
MOJO_MODE, 180, 182
TEST_PASSWORD, 752, 754, 755
TEST_URL, 752, 753, 755
TEST_USER, 753–755
TO_PASSWORD, 185, 187, 188, 752, 754, 755, 790
TO_URL, 185, 187, 188, 752, 753, 755, 790
TO_USER, 185, 187, 188, 752–755, 790
ERROR (traffic_ops_ort.configuration.LogLevels attribute), 192

Index 815
F
FATAL (traffic_ops_or.configuration.LogLevels attribute), 192
Federation, 798
Federations, 798
forward proxy, 798
fullHostname (traffic_ops_or.configuration.Configuration attribute), 191

G
genConfigRoutes.py command line option
-refPasswd PASSWD, 753
-refUser USER, 753
-testPasswd PASSWD, 754
-testUser USER, 754
-C, -no-server-configs, 754
-h, -help, 753
-k, -insecure, 754
-l LOG_LEVEL, -log_level LOG_LEVEL, 754
-s, -snapshot, 754
-v, -version, 754
Generate ISO, 70
generate_deliveryservice_ssl_keys() (trafficops.TOSession method), 766
generate_deliveryservice_url_signature_keys() (trafficops.TOSession method), 766
generate_iso() (trafficops.TOSession method), 767
geo localization or geo routing, 800
get() (in module to_access), 791
get() (trafficops.RestApiSession method), 783
get_all_cachegroup_parameters() (trafficops.TOSession method), 767
get_all_deliveryservice_servers() (trafficops.TOSession method), 767
get_api_capabilities() (trafficops.TOSession method), 767
get_api_capabilities_by_id() (trafficops.TOSession method), 767
get_cdns_routing() (trafficops.TOSession method), 769
get_cdns_usage() (trafficops.TOSession method), 770
get_change_logs() (trafficops.TOSession method), 770
get_change_logs_for_days() (trafficops.TOSession method), 770
get_change_logs_newcount() (trafficops.TOSession method), 770
get_coordinates() (trafficops.TOSession method), 770
get_current_snapshot_crconfig() (trafficops.TOSession method), 770
get_delivery_service_capacity() (trafficops.TOSession method), 770
get_delivery_service_failover_state() (trafficops.TOSession method), 770
get_delivery_service_health() (trafficops.TOSession method), 770
get_delivery_service_routing() (trafficops.TOSession method), 771
get_delivery_service_stats() (trafficops.TOSession method), 771
get_deliveryservice_by_id() (trafficops.TOSession method), 771
get_deliveryservice_ineligible_servers() (trafficops.TOSession method), 771
get_deliveryservice_regexes_by_id() (trafficops.TOSession method), 771
get_deliveryservice_regexes_by_regex_id() (trafficops.TOSession method), 771
get_deliveryservice_servers() (trafficops.TOSession method), 771
get_deliveryservice_unassigned_servers() (trafficops.TOSession method), 771
get_deliveryservices() (trafficops.TOSession method), 772
get_deliveryservices_regexes() (trafficops.TOSession method), 772
get_deliveryserviceserver() (trafficops.TOSession method), 772
get_division_by_id() (trafficops.TOSession method), 772
get_divisions() (trafficops.TOSession method), 772
get_division_by_id() (trafficops.TOSession method), 772
get_divisions() (trafficops.TOSession method), 772
get_federation_delivery_services() (trafficops.TOSession method), 772
get_federation_for_cdn_by_id() (trafficops.TOSession method), 772
get_federation_resolvers() (trafficops.TOSession method), 772
get_federation_resolvers_by_id() (trafficops.TOSession method), 773
get_federation_users() (trafficops.TOSession method), 773
get_federations() (trafficops.TOSession method), 773
get_federations_for_cdn() (trafficops.TOSession method), 773
get_federation_resolvers_by_id() (trafficops.TOSession method), 773
get_federation_users() (trafficops.TOSession method), 773
get_federations() (trafficops.TOSession method), 773
get_federations() (trafficops.TOSession method), 773
get_federations() (trafficops.TOSession method), 773
get_fhinfo() (trafficops.TOSession method), 773
get_hjobs() (trafficops.TOSession method), 773
get_hosversions() (trafficops.TOSession method), 773
get_jobs() (trafficops.TOSession method), 773
get_jobs() (trafficops.TOSession method), 773
get_job_by_id() (trafficops.TOSession method), 773
get_job_by_id() (trafficops.TOSession method), 773
get_osversions() (trafficops.TOSession method), 773
get_osversions() (trafficops.TOSession method), 773
get_pending_snapshot_crconfig() (trafficops.TOSession method), 774
get_parameters() (trafficops.TOSession method), 774
get_parameters() (trafficops.TOSession method), 774
get_parameters() (trafficops.TOSession method), 774
get_parameters() (trafficops.TOSession method), 774
get_parameters_by_profile_id() (trafficops.TOSession method), 774
get_parameters_by_profile_name() (trafficops.TOSession method), 774
get_physical_location_by_id() (trafficops.TOSession method), 774
get_physical_locations() (trafficops.TOSession method), 774
get_physical_locations() (trafficops.TOSession method), 774
get_profile_by_id() (trafficops.TOSession method), 774
get_profile_specific_config_files() (trafficops.TOSession method), 774
get_profiles() (trafficops.TOSession method), 774
get_profiles() (trafficops.TOSession method), 774
get_regregion_by_id() (trafficops.TOSession method), 774
get_regregion_by_id() (trafficops.TOSession method), 774
get_regions() (trafficops.TOSession method), 775
get_regions() (trafficops.TOSession method), 775
get_roles()  (trafficops.TOSession method), 775
get_server_by_id()  (trafficops.TOSession method), 775
get_server_config_files()  (trafficops.TOSession method), 775
get_server_delivery_services()  (trafficops.TOSession method), 775
get_server_details()  (trafficops.TOSession method), 775
get_server_specific_config_file()  (trafficops.TOSession method), 775
get_server_status_count()  (trafficops.TOSession method), 775
get_server_type_count()  (trafficops.TOSession method), 776
get_server_update_status()  (trafficops.TOSession method), 776
get_servers()  (trafficops.TOSession method), 776
get_static_dns_entries()  (trafficops.TOSession method), 776
get_staticdnsentries()  (trafficops.TOSession method), 776
get_statuses()  (trafficops.TOSession method), 776
get_statuses_by_id()  (trafficops.TOSession method), 776
get_system_info()  (trafficops.TOSession method), 776
get_tenant_by_id()  (trafficops.TOSession method), 776
get_tenants()  (trafficops.TOSession method), 776
get_to_extensions()  (trafficops.TOSession method), 777
get_traffic_monitor_cache_stats()  (trafficops.TOSession method), 777
get_trimmed_physical_locations()  (trafficops.TOSession method), 777
get_trimmed_profiles()  (trafficops.TOSession method), 777
get_type_by_id()  (trafficops.TOSession method), 777
get_types()  (trafficops.TOSession method), 777
get_types_only_names()  (trafficops.TOSession method), 777
get_unassigned_profiles_by_parameter_id()  (trafficops.TOSession method), 777
get_unassigned_parameters_by_profile_id()  (trafficops.TOSession method), 777
get_user_by_id()  (trafficops.TOSession method), 777
get_user_delivery_services()  (trafficops.TOSession method), 777
get_users()  (trafficops.TOSession method), 778
getJSONResponse()  (in module traffic_ops_ort.utils), 199
getMyChkconfig()  (traffic_ops_ort.to_api.API method), 196
getMyConfigFiles()  (traffic_ops_ort.to_api.API method), 196
getMyPackages()  (traffic_ops_ort.to_api.API method), 197
getMyStatus()  (traffic_ops_ort.to_api.API method), 197
getMyUpdateStatus()  (traffic_ops_ort.to_api.API method), 197
getRaw()  (traffic_ops_ort.to_api.API method), 197
getTextResponse()  (in module traffic_ops_ort.utils), 199
getYesNoResponse()  (in module traffic_ops_ort.utils), 199

H
head()  (in module to_access), 791
head()  (trafficops.RestApiSession method), 783
Health, 65
Health Protocol, 800
Health Tab, 62

I
INFO  (traffic_ops_ort.configuration.LogLevels attribute), 192
install()  (traffic_ops_ort.packaging.Package method), 195
installArgs
    (traffic_ops_ort.packaging.Package attribute), 195
INTERACTIVE
    (traffic_ops_ort.configuration.Configuration.Modes attribute), 191
Invalidate Content, 74
InvalidJSONError, 781
is_open (trafficops.RestApiSession attribute), 784
isInstalled()
    (traffic_ops_ort.packaging.Package method), 195
ISO, 70
L
localization, 800
LOG_FORMAT (in module traffic_ops_ort.configuration), 192
LOG_LEVEL, -l LOG_LEVEL,
        -log_level LOG_LEVEL
traffic_ops_ort command line option, 186
log_with_debug_info() (in module trafficops.utils), 786
logged_in (trafficops.TOSession attribute), 778
LOGIN
    traffic_ops_ort command line option, 187
login() (trafficops.TOSession method), 778
LoginError, 781
LogLevels (class in traffic_ops_ort.configuration), 192
M
main() (in module traffic_ops_ort), 190
Mid, 800
Mid-tier, 800
Mid-tier cache, 800
Mid-tier cache server, 800
Mid-tier cache servers, 800
Mid-tier caches, 800
MODE
    traffic_ops_ort command line option, 186
MOJO_MODE, 182
N
name (traffic_ops_ort.packaging.Package attribute), 196
    (traffic_ops_ort.configuration.LogLevels attribute), 192
O
OperationError, 781
options() (in module to_access), 791
options() (trafficops.RestApiSession method), 784
origin, 800
origin server, 801
origin servers, 801
origins, 801
ORT, 801
ORTException, 193
output() (in module to_access), 791
P
Package (class in traffic_ops_ort.packaging), 195
Parameter, 801
Parameters, 801
parent, 801
parents, 801
parse_arguments() (in module to_access), 792
parseTOURL() (in module traffic_ops_ort.configuration), 192
parseTSRoot() (in module traffic_ops_ort.configuration), 193
patch() (in module to_access), 792
patch() (trafficops.RestApiSession method), 784
PATH
toaccess command line option, 789
Physical Location, 801
post() (in module to_access), 792
post() (trafficops.RestApiSession method), 785
processConfigurationFiles() (in module traffic_ops_ort.main_routines), 194
processPackages() (in module traffic_ops_ort.main_routines), 194
processServices() (in module traffic_ops_ort.main_routines), 194
Profile, 801
profileId (traffic_ops_ort.to_api.ServerInfo attribute), 198
profileName (traffic_ops_ort.to_api.ServerInfo attribute), 198
Purge, 74
put() (in module to_access), 792
put() (trafficops.RestApiSession method), 785
Q
Queue Updates, 73
R
Region, 804
REPORT (traffic_ops_ort.configuration.ConfigurationModes attribute), 191
request() (in module to_access), 792
resp (trafficops.InvalidJSONError attribute), 781
resp (trafficops.OperationError attribute), 781
RestApiSession (class in trafficops), 782
REVALIDATE (traffic_ops_ort.configuration.ConfigurationModes attribute), 191
revalidateState() (in module traffic_ops_ort.main_routines), 194
reverse proxy, 804
RFC
RFC 7516, 480–482
RFC 7518, 480–482
RFC
RFC 1035, 32, 348, 353, 363, 368
RFC 2616, 4, 804
RFC 2616#section-14.9, 5
RFC 4033, 103
RFC 4044, 103
RFC 4045, 103
RFC 6781, 103
RFC 6781#section-4.1.1.1, 104
RFC 7129, 103
Role, 806
run() (in module traffic_ops_ort.main_routines), 194
S
sanitize() (traffic_ops_ort.to_api.ServerInfo method), 198
server_url (trafficops.RestApiSession attribute), 786
serverId (traffic_ops_ort.to_api.ServerInfo attribute), 198
ServerInfo (class in traffic_ops_ort.to_api), 198
ServerInfo (traffic_ops_ort.configuration.Configuration attribute), 191
serverIpv4 (traffic_ops_ort.to_api.ServerInfo attribute), 198
serverName (traffic_ops_ort.to_api.ServerInfo attribute), 198
servers_queue_update() (trafficops.TOSession method), 778
serverTcpPort (traffic_ops_ort.to_api.ServerInfo attribute), 198
session (trafficops.RestApiSession attribute), 786
setLogLevel() (in module traffic_ops_ort.configuration), 193
setStatusFile() (in module traffic_ops_ort.main_routines), 195
shortHostname (traffic_ops_ort.configuration.Configuration attribute), 192
Snapshot, 806
Snapshot CRConfig, 73
snapshot_crconfig() (trafficops.TOSession method), 778
Status, 806
STATUS_FILE_DIR (in module traffic_ops_ort.main_routines), 193
sync_ts_databases command line option
-database database_name, 123
-days N, 123
-help, 123
-source-password password, 123
-source-url URL, 123
-source-user username, 123
-target-password password, 123
-target-url URL, 123
-target-user username, 124

SYNCDS

syncDSState() (in module traffic_ops_ort.main_routines), 195

T

Tenant, 806
Tenants, 806
TEST_PASSWORD, 752, 754
TEST_URL, 752, 753
TEST_USER, 753, 754
to_access (module), 788
TO_PASSWORD, 185, 188, 752, 754, 790
TO_URL, 185, 187, 188, 752, 753, 790
to_url (traffic.ops.TOSession attribute), 778
TO_URL, -to_url TO_URL

traffic_ops_ort command line option

TO_USER, 185, 187, 752–754, 790
toaccess command line option

-h, -help, 185
-k, -insecure, 185
-p PASSWORD, -to_password PASSWORD, 187
-u USER, -to_user USER, 187
-v, -version, 185
LOG_LEVEL, -l LOG_LEVEL,
-MODE, 186

traffic_ops_ort (module), 184
traffic_ops_ort command line option

-dispersion DISP, 185
-login Dispersion DISP, 185
-retries RETRIES, 185
-rev_prox_disable, 185
-ts_root PATH, 185
-wait_for_parents INT, 185

traffic_ops_ort (class in trafficops), 756

TOSession (traffic_ops_ort.configuration.Configuration attribute), 191
toUrl (traffic_ops_ort.to_api.ServerInfo attribute), 198

TRACE

traffic_ops_ort.configuration.LogLevels attribute, 192

traffic_ops_ort (module), 184

traffic_ops_ort.to_api.ServerInfo attribute), 198
toRevProxyUrl

toRevProxyUrl (traffic_ops_ort.to_api.ServerInfo attribute), 198
trafficops.__version__ (module), 787
trafficops.utils (module), 786
Type, 807
Types, 807

U
uninstall() (traffic_ops_ort.packaging.Package, 196
uninstallArgs (traffic_ops_ort.packaging.Package, 196
update_asn() (trafficops.TOSession method), 778
update_cache_group_fallbacks() (trafficops.TOSession method), 778
update_cachegroups() (trafficops.TOSession method), 778
update_cdn_by_id() (trafficops.TOSession method), 779
update_coordinates() (trafficops.TOSession method), 779
update_deliveryservice_by_id() (trafficops.TOSession method), 779
update_deliveryservice_regexes() (trafficops.TOSession method), 779
update_deliveryservice_safe() (trafficops.TOSession method), 779
update_division() (trafficops.TOSession method), 779
update_federation_in_cdn() (trafficops.TOSession method), 779
update_origins() (trafficops.TOSession method), 780
update_parameter() (trafficops.TOSession method), 780
update_physical_location() (trafficops.TOSession method), 780
update_profile_by_id() (trafficops.TOSession method), 780
update_region() (trafficops.TOSession method), 780
update_server_by_id() (trafficops.TOSession method), 780
update_server_status_by_id() (trafficops.TOSession method), 780
update_staticdnsentries() (trafficops.TOSession method), 780
updateTrafficOps() (traffic_ops_ort.to_api.API method), 197
validate_parameter_exists() (trafficops.TOSession method), 781
version (traffic_ops_ort.packaging.Package attribute), 196
VERSION (traffic_ops_ort.to_api.API attribute), 196
WARN (traffic_ops_ort.configuration.LogLevels attribute), 192